國立臺灣師範大學科技與工程學院電機工程學系

博士論文

Department of Electrical Engineering

College of Technology and Engineering

National Taiwan Normal University

Doctoral Dissertation

Sim2Real 在高動態環境下人形機器人的平衡控制

Sim2Real for Balancing Control of Humanoid Robot in Highly Dynamic Environment

鄭在植

Jaesik Jeong

指導教授: 包傑奇 博士

Advisor: Baltes, Jacky Hansjoerg, Ph.D.

中華民國 112 年 7 月

July, 2023

Acknowledgements

I started my Ph.D. at a relatively late age. However, with the support and help of my loved people, I was able to finish my Ph.D. well.

First, I would like to express my gratitude to my beloved wife, Jeehyun, who supported my Ph.D. journey and studied with me while experiencing joys and sorrows abroad. And I sincerely thank the parents of both families who always supported me and Jeehyun.

I would like to thank my advisor Prof.Jacky Baltes. He was the main reason why I started my Ph.D. at NTNU. His support and guidance helped set the direction for my research as a roboticist and researcher and allowed me to grow.

I would also like to thank all my defense committee members: Prof.Wei-Yen Wang, Prof.You-Shan Su, Prof.Kuo-Yang Tu, Prof.Chih-Cheng Liu, and Prof.Chen-Chien James Hsu. I could complete my thesis with their thoughtful feedback and direction.

And I really thank our former and current ERC members who immensely helped me and Jeehyun. Especially, I want to thank Christmann, Eko, Hanjaya, Joe, Greene, Sun, Song, Leo, Ingram, and Ugo.

I also want to thank my Korean and international friends.

This thesis is dedicated to all of my loved people.

Abstract

This thesis presents comprehensive research into the dynamic balance control of a humanoid robot, namely the Robinion2S. The research initiates with detail of humanoid robot platforms with mechatronic systems, walking gait algorithms, and perception systems. Special focus is given to Robinion2S, the latest version of the Robinion series, which forms the backbone of the study. The experiments show the limitations of traditional PID-based balance control methods when deployed in a complex, dynamic environment such as a balance board. Despite optimization efforts using a high-throughput random search algorithm within Nvidia's Isaac Gym simulation environment, the PID-based approach fails to ensure consistent balance. This result leads to the need for more robust control strategies. The research focuses on reinforcement learning techniques to balance control to overcome the result. Despite the challenges of traditional control theory, reinforcement learning techniques show potential as a viable solution to the intricacies of balance control. The reinforcement learning models demonstrate their adaptability and robustness in maintaining balance, hinting at their potential to solve more complex control problems. Extending the study into real-world applications, the Sim2Real approach is developed. The Sim2Real approach implements the trained reinforcement learning models into a dynamic, physical environment. Despite not achieving ideal results, the approach demonstrates the potential for trained models to transfer control policies effectively from simulation to the real environment. This thesis provides potential methods in the field of balance control in humanoid robots, motivating a shift from traditional control methods to more robust reinforcement learning techniques. Despite not being ideal, the obtained results show significant potential for future research and advancements in the robotics field. This research provides a foundation for understanding the balance control in the humanoid robot and potential strategies to optimize the performance in a real-world environment.

Keyword： Humanoid Robots, Balance Control, Reinforcement Learning, Sim2Real

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

Humanoid robots have emerged as versatile and valuable platforms for a wide range of applications, including last mile, entertainment, and industrial automation. Achieving balance control is a fundamental challenge in humanoid robot development, as it directly impacts their stability, mobility, and ability to perform complicated tasks [1, 2]. The ability to maintain balance is required for robots to navigate uneven and dynamic environments, interact with objects, and ensure safe and efficient operation [3, 4]. However, achieving robust balance control in humanoid robots poses significant challenges. Humanoid robots possess complex joints and non-linear dynamics, making it challenging to design control strategies to handle the uncertainties and disturbances encountered in the real world [5]. External forces, such as environmental disturbances or interactions with objects, can destabilize the robot and lead to falls or loss of balance [6]. Moreover, the inherent limitations of the physical hardware, such as joint flexibility, sensor noise, and actuator constraints, further complicate the control problem.

The problem of balance control becomes even more challenging when considering highly dynamic tasks and environments [7, 8, 9]. Humanoid robots often need to traverse uneven terrains, recognize obstacles, or perform agile movements, requiring precise and adaptive balance control strategies [10]. Additionally, the need for real-time responsiveness further adds complexity, as balance control algorithms operate at high frequencies to ensure stability and maintain desired poses [11]. Existing control approaches for balance control in humanoid robots range from classical methods, such as Proportional-Integral-Derivative (PID) control, to more advanced techniques, including model-based control, optimization-based control, and learning-based approaches [12, 13, 14]. While these methods have made significant progress, they often have limitations regarding adaptability, robustness, and scalability. Designing control strategies that effectively handle complex dynamics, external disturbances, and varying task requirements remains an ongoing research challenge.

In other words, as our exploration into humanoid robotics deepens, the challenge of maintaining a dynamic balance in such robots has emerged as a pressing concern [15]. This is the main problem that our thesis solves. To further elucidate this problem, discussing specificities

that make balance control in humanoid robots such a complex task is necessary.

Inherent Complexity and High-Dimensionality of Humanoid Robots: Unlike other robotic platforms, humanoid robots possess a large number of degrees of freedom. Managing these numerous degrees of freedom simultaneously and ensuring their harmonious functioning is an intricate challenge. The high-dimensional configuration space, combined with the non-linear and non-convex nature of balance-related problems, contributes to the computational complexity of this problem.

Unpredictability of Real-World Environments: Humanoid robots are operated in real-world environments where conditions can be highly unpredictable and dynamic. These conditions, ranging from varying terrains to unexpected disturbances, make the task of maintaining balance even more complex. Traditional control methods often fail to adapt to these changes, leading to a loss of balance.

Limitations of Traditional Control Methods: Traditional control strategies, such as PID-based methods, have shown limitations in handling the complex interactions involved in balance control, especially in dynamic environments. These methods, while capable of simple task handling, often fail when it comes to intricate and adaptive balance control.

Sim2Real Gap: One of the significant challenges in implementing learning-based methods is the Sim2Real gap. Although simulations provide a safe and efficient environment for training and testing control policies, discrepancies often occur when transferring these learned policies to real robots. Bridging this gap without overfitting to the simulated environment is a difficult task.

The main objective of this thesis is to solve the problem of balance control in humanoid robots, specifically focusing on the Robinion2S platform. The goal is to develop and implement effective control algorithms that enable the robot to maintain balance and perform complex movements on a balance board. By achieving robust and adaptive balance control, the thesis aims to enhance the stability, mobility, and functionality of humanoid robots, applying real-world applications. In order to solve the problem, the thesis presents various control approaches, including PID-based control, reinforcement learning-based control, and the Sim2Real

framework. It analyzes the strengths and limitations of each approach and aims to develop novel algorithms that can handle the challenges of balance control in dynamic and unpredictable environments. By leveraging advances in control theory, machine learning, and simulation, the thesis pushes the boundaries of humanoid robot balance control and contributes to developing more capable and versatile robotic systems.

1.1 Key Contributions

This thesis makes several key contributions with novel approaches to balance control for the humanoid robot in a highly dynamic environment.

- Comprehensive analysis of humanoid robot platforms with advanced capabilities

  It provides a detailed analysis of humanoid robot platforms, the Robinion series, with useful results in mechatronic systems, walking gait algorithms, and perception systems. The emphasis is laid on advanced capabilities and suitability for real-world applications.

- Implementation of traditional control approach for the highly dynamic environment

  This research identifies the limitations of a traditional PID-based balance control method, showing an imperfection when dealing with highly dynamic environments. This establishes the need for more robust and adaptable control strategies.

- Analysis of reinforcement learning for balance control

  The introduction and analysis of reinforcement learning as a method for achieving balance control demonstrate its potential to solve the challenging problem of maintaining balance on a balance board.

- Deployment of trained models on the real-world robot platform

  The training models were deployed from the reinforcement learning approach on the real-world robot platform Robinion2S. This work validates the feasibility of using reinforcement learning for real-world balance control and provides a practical evaluation of the Sim2Real.

1.2 Outline

This thesis is composed of seven chapters, with the core of the scientific contributions and results being presented from Chapter 2 to Chapter 6, which are organized thematically in two Parts: Humanoid Robot Platforms and Balance Control. Each of these chapters initiates with the problem statement, followed by an in-depth exposition of the methodology employed in the study. Each chapter finalizes with the evaluation of the proposed methods and a discussion of the advantages, limitations, and future work. Structuring the thesis in this form ensures a logical and seamless flow of the experiments and results.

Chapter 2, *Humanoid Robot Platforms*, serves as an underpinning to this research. This chapter offers a thorough review of humanoid robot platforms, with a specific focus on the Robinion series. The chapter dissects the major components contributing to the functionalities, such as the mechatronic system, walking gait algorithm, and perception system. The robustness and cost-effectiveness of the Robinion series, as well as their ability to maintain stability, are demonstrated through experiments.

Chapter 3, *Literature Review for Balance Control*, presents a comprehensive review of literature on balance control in humanoid robots. It examines control methods employed in the thesis, analyzing their strengths, limitations, and areas for improvement. This review is a foundation for the subsequent chapters, providing an analysis of the traditional control method and state-of-the-art techniques. The literature review establishes the theoretical background and motivates the development of novel balance control algorithms in the following chapters.

Chapter 4, *Balance Control with PID*, dives into the application of a PD-based balance control approach, tested on Robinion2S on different balance boards. It discusses the challenges encountered in tuning the PID gains and proposes a random search algorithm to optimize the control parameters. The chapter not only shows how this traditional method works in controlling balance but also uncovers its limitations, particularly when dealing with dynamic environments.

Chapter 5, *Balance Control with Reinforcement Learning*, introduces the application of reinforcement learning as a means to overcome the limitations identified in the previous chapter. This chapter presents a shift from traditional balance control methods to a more dynamic and

adaptive reinforcement learning technique. The chapter analyzes various control modes and their effectiveness in handling the complex balancing task.

Chapter 6, *Sim2Real*, draws on the preceding chapters to propose and test a Sim2Real approach for balance control. Here, the capabilities of reinforcement learning in a real-world application are validated, as the trained models are implemented on the real robot, Robinion2S. The chapter demonstrates both the potential of the Sim2Real framework and its limitations, setting a foundation for future research.

Chapter 2. Humanoid Robot Platforms

This chapter presents intelligent humanoid robot platforms, which are developed about hardware and software ourselves, namely the Robinion series, for various research purposes. There are five types of humanoid robots that Chapter 2 is introduced: Robinion, Robinion Senior(Sr.), Robinion2P (Robinion 2nd version with Parallel kinematic leg), and Robinion2S (Robinion 2nd version with Standard kinematic leg) [16, 17]. In this dissertation, experiments are carried out using the latest humanoid robot platform, Robinion2S, which has a standard kinematic leg design. Robinion2S is developed based on the research results of Robinion, Robinion Sr., and Robinion2P, which were developed in advance. The effectiveness of the research and development process for Robinion2S, therefore, is justified by presenting the performance of four different humanoid robots. Robustness mechanical design is essential for humanoid robots to move in a precise position. Furthermore, well-scheduled and optimized firmware and software are crucial to the movement of the robot. This chapter aims to introduce improvements and advancements of humanoid robot platforms in electrical and mechanical structures, kinematics analysis, gait generation, and software design for the basics of balance control.

2.1 Robinion

In this section, the hardware of an intelligent humanoid robot, Robinion version 1 (Robinion), is described, including its mechanical and electrical structures. Robinion is an autonomous humanoid robot platform serving as a research and competition for omnidirectional walking gaits, motion algorithms, and participation in intelligent humanoid robot competitions such as the FIRA Hurocup, Robocup Humanoid League, and Humanoid Robot Application Challenge - Robot Magic (HAC) (see Figure 2.1) [18, 19, 20, 21, 22, 23]. For the mechanical structure of Robinion, previous humanoid robots developed by the ZSTT from Korea, which has been the robotics team since 2010, have been influenced in the design of the robot. The humanoid robot is the first model of a middle-sized robot above 80 cm in height from team ZSTT. The first Robinion was created in 2013, and in terms of mechanics and electronics, Robinion has continued to evolve since its inception. Since the initial Robinion was developed to focus on gait analysis for stable walking, the upper body and arms were designed to have the slightest degree

Figure 2.1: Robinion Version 1 in the competitions (FIRA Hurocup & HAC)

of freedom. The humanoid robot has continuously added mechanical structure, such as a torso joint, five degrees of freedom for each arm, and ten degrees of freedom for a gripper because it required high degrees of freedom for accurate manipulation to perform various magic tricks in HAC and carry out events such as weightlifting, basketball, and archery in FIRA HuroCup. Moreover, as technology advances, the performance of its electronic devices has been improved for various algorithms, such as high-spec image processing, deep learning, and localization. The mechatronic system is described in detail in Sections 2.1.1 and 2.1.2.

2.1.1 Mechanics

Detailed information about the mechanical structure of Robinion version 1 is presented in this section. Robinion is the first humanoid robot with parallel kinematic legs designed by the ZSTT team. As an alternative to standard kinematics, parallel kinematics has several advantages, such as reducing weight and cost by creating legs with one fewer actuator and simplifying inverse kinematics calculations. As shown in Table 1, Robinion has the following mechanical specifications.

The humanoid robot is 85cm tall and weighs around 7kg. The robot has 41 degrees of

Table 2.1: Robinion mechanical specification

Items	Description
Height	85 cm
Weight	7 kg
Kinematic structure	Parallel kinematics in the legs, standard kinematics in the arms and head
Degrees of Freedom	41 (10 in legs, 1 in torso, 8 in arms, 2 in the head, 20 in grippers)
Actuators	MX-106 x 12, MX-64 x 6, MX-28 x 2, AX-12A x 2
Gripper	RH2D
Fream	Aluminium alloy 5052
Surface treatment	Anodising & Sanding
Bearing	ball(flanged type)

freedom (DoF): five in each leg, one in the torso, four in each arm, ten in each hand, and two in the head. It is equipped with an intelligent gripper for each hand, namely RH2D[1]. The innovative gripper has nine DoF for three fingers and one for the wrist. The RH2D is a tendon-based, under-actuated unit with tendon tension equalization in a compact and lightweight form. Objects can be grasped with force control by measuring the current of a motor inside the hand. With the RH2D, finger joints are protected against impact by a magnetic finger detachment system. To achieve maximum reliability, the gripper's tendons are made from Dyneema (a Kevlar fiber) [24]. The parallel mechanism is adapted for humanoid robot legs to reduce weight and cost. Aluminum flat sheets (1.5T, 2T, and 3T) are used in the fabrication of the frame for our humanoid robot. The CNC machine cuts the flat aluminum sheets, and the milling machine makes side-tapping holes to build precise frames. In addition, the frames are anodized and sanded to improve durability and be corrosion-resistant. Dynamixel, such as MX series[2] and AX-12A[3], is equipped for all joints of Robinion. The humanoid robot was fitted with MX-106 on joints requiring high torque, such as the pitch and roll joints of each hip and ankle, the pitch joint of each shoulder, and the pitch joint of each elbow. The MX series is used for joints other than the head, MX-64 for the yaw joint of each hip and the roll joint of each shoulder, MX-28 for the yaw joint of each elbow, and AX-12A for the roll and pitch joint of the head.

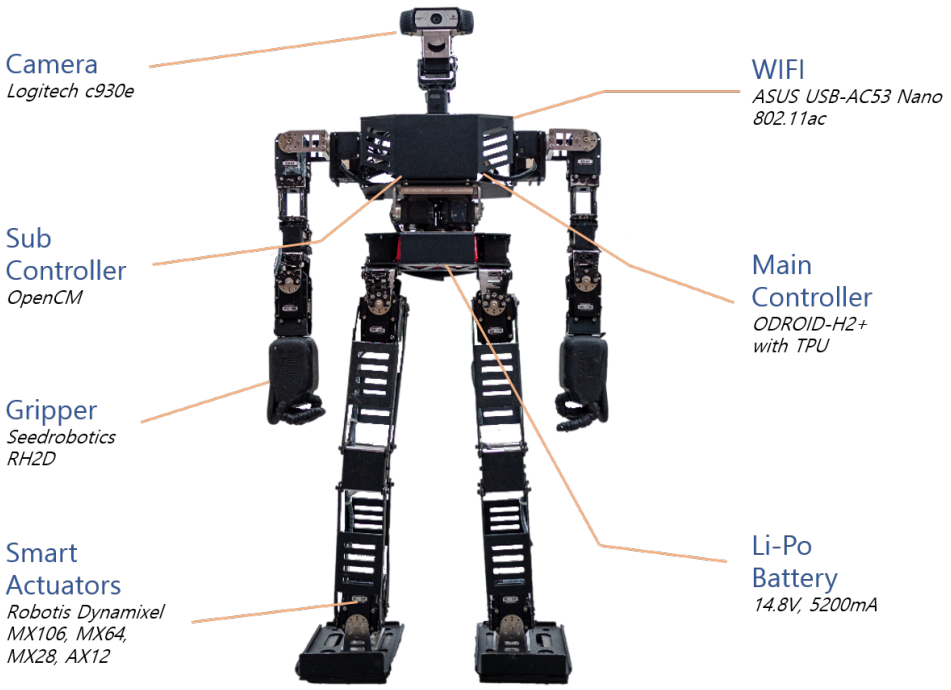


Figure 2.2: Electrical components on Robinion

2.1.2 Electronics

Figure 2.2 shows the electrical components' position of Robinion. There are two controllers that work together: the main controller, which schedules tasks and recognizes the environments of Robinion, and the sub-controllers, which operate smart actuators and read the orientation from IMU. As the main controller, it is capable of reading frames from a webcam for detecting objects with image processing and deep learning in various research such as SLAM and navigation and entertainment tasks. It is equipped with a 4-cell Li-Po battery as the main power source, allowing it to run for up to 1 hour. Table 2.2 presents the electrical specifications of Robinion, including the power source, controllers, and sensor.

It is powered by an ODROID-H2+ onboard processor with a quad-core Intel J4115 processor with 4GB RAM and 128GB SSD M.2. The Logitech C930e webcam allows Robinion to check the environment, and image processing enables it to recognize shapes and colors. Object recognition is based on deep learning, and the humanoid robot is equipped with an ASIC chip which Google produces a lightweight Tensor Processing Unit (Edge-TPU) [25, 26]. Using Tiny-YOLO V3, a deep-learning algorithm that recognizes objects to enable autonomous humanoid

[1]https://www.seedrobotics.com/rh2d-advanced-manipulator
[2]https://emanual.robotis.com/docs/en/dxl/mx/
[3]https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

Table 2.2: Robinion electrical specification

Items	Description
Power Source	LiPo battery (14.8V, 5200mAh x 2) Up to 60 min
Main Controller	ODROID-H2+ (Intel J4115) & Coral USB Accelerator
Motion Controller	OpenCM9.04 & EXP board (ARM Cortex-M3, 128KB Flash, 20KB SRAM)
Sensor Controller	OpenCM9.04 (ARM Cortex-M3, 128KB Flash, 20KB SRAM)
Sensors	IMU(MPU6050), Camera(Logitec C930e)

robot athletics and magic shows [27]. The control of forward and inverse kinematics is used for open-loop walking gaits and manipulation tasks. Additionally, the humanoid robot is capable of performing speech acts such as recognizing the voice, using TTS (Text-To-Speech), and playing MP3 for HRI(Human-Robot Interaction) and entertainment tasks.

2.2 Robinion Sr.

This section introduces Robinion Sr., a lightweight humanoid robot platform that stands 135cm tall and weighs approximately 12kg (See Figure 2.3). The main focus of this research is to develop a low-cost and robust humanoid robot, and this section describes the hardware

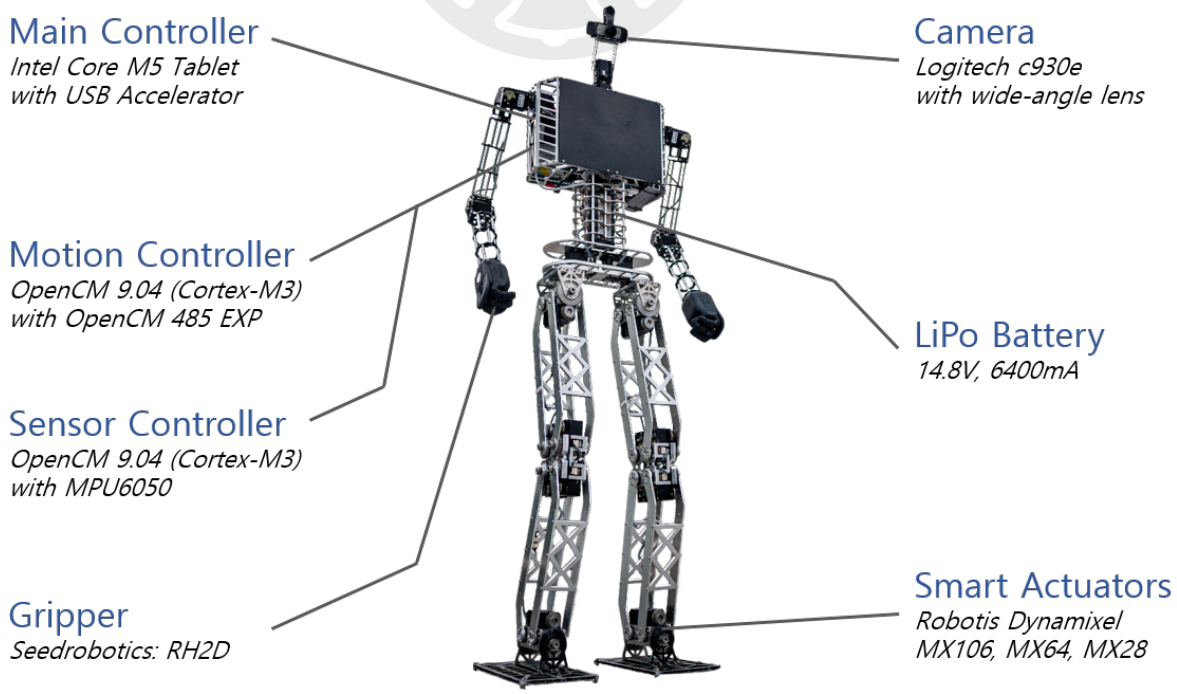


Figure 2.3: Electrical components on Robinion Sr.

systems used to achieve this goal. The humanoid robot uses a low-cost tablet computer as the main computer, which does not have a GPU. Nonetheless, the perception system is still able to deploy state-of-the-art deep learning systems using a modified tiny YOLO-V3 model and the Google Coral USB accelerator[4]. In addition, it can achieve real-time inference speed by compiling the processed models into a particular format([28]).

The development of humanoid robots involves considerable costs and time investments, making it imperative to clearly define the objectives and scalability of such robots before undertaking their manufacture [29]. This research aims to optimize the mechanical, electronic, and software components to achieve maximum robot performance while minimizing manufacturing expenses. In this section, we present a comprehensive overview of the mechanical structure, electrical configuration, and GUI-based software that comprise our lightweight and robust humanoid robot. Notably, the cost of developing our humanoid robot is lower than that of the commercially available small humanoid robot, such as the Darwin-OP3[5] of ROBOTIS[6] from ROBOTIS. Our robot's manufacturing cost is approximately one-third of the NimbRo-OP2(X) cost from the University of Bonn, establishing it as the most cost-effective humanoid robot in a similar size range [30]. Table 2.3 provides a detailed comparison of the humanoid robot platforms, showcasing the specifications of Robinion Sr., Nimbro-OP2(X), and Darwin-OP3. Through this analysis, we show the distinct advantages and unique capabilities of Robinion Sr., supporting its position as a competitive and cost-efficient humanoid robot solution.

Table 2.3: Comparison of specification of the humanoid robot platforms

Specification	Robinion Sr.	Nimbro OP2(X)	Darwin-OP3
Height(cm)	134	135	51
Weight(kg)	12	19	3.5
Kinematic structure(Leg)	Parallel	Parallel	Standard
Degrees of freedom(DoF)	38	18	20
Frame	Aluminium alloy	Polyamide 12(PA12)	Aluminium alloy
Gripper	RH2D(10 DoF)	-	-
Cost(USD)	9,500	26,500	11,000

[4]https://coral.ai/products/accelerator/
[5]https://emanual.robotis.com/docs/en/platform/op3/introduction/
[6]http://en.robotis.com/

In comparison to other humanoid robots of similar size, Robinion Sr. stands out with its remarkable degree of freedom, facilitated by the inclusion of a gripper. While conventional humanoid robots typically feature six joints on each leg as part of their standard structure, Robinion Sr. uses a parallel-kinematics design, eliminating the need for an actuator in each knee. This innovative approach reduces the total number of actuators in the humanoid robot's legs, resulting in a lighter and more cost-effective robot. Despite the significant enhancements in its mechanical and electrical structure, Robinion Sr. is a lower manufacturing price than the commercially available robot Darwin-OP3. This cost advantage, combined with improved performance, renders the manufacturing of a high-performance humanoid robot like Robinion Sr. beneficial in terms of cost efficiency, maintenance, and scalability. Since the initial development of Robinion Sr. in 2017, continuous efforts have been made to enhance its mechatronic system and overall performance. Detailed descriptions of these advancements can be found in Sections 2.2.1 and 2.2.2.

2.2.1 Mechanics

The efficient movement and performance of robots are heavily influenced by the robustness of their mechanical design, prompting robotic designers to meticulously consider numerous factors such as size, weight, frame, actuators, torque, gears, bearings, and cost. Given the historically high manufacturing costs associated with humanoid robots, prioritizing reusability and scalability in the robot design becomes important. Originating in 2017, Robinion Sr. has undergone a series of continuous partial upgrades to enhance its mechanical structure continually. Developed primarily for robot gait research and participation in intelligent robot competitions like Robocup and FIRA, Robinion Sr. places particular emphasis on its leg design to achieve both high efficiency and cost-effectiveness. The mechanical specifications of this lightweight humanoid robot are detailed in Table 2.4, optimizing for superior performance, reliability, and adaptability.

Robinion Sr. is a height of 134cm and weighs approximately 12kg, showcasing a lightweight humanoid robot. In contrast to the standard humanoid robot kinematic structure, where each leg typically comprises six joints containing roll and pitch joints in the ankles, pitch joints in the knees, and yaw, pitch, and roll joints in the hips, Robinion Sr. adopts a parallel mechanism with

Table 2.4: Robinion Sr. mechanical specification

Items	Description
Height	134 cm
Weight	12 kg
Kinematic structure	Parallel kinematics in the legs, standard kinematics in the arms and head
Degrees of Freedom	38 (10 in legs, 6 in arms, 2 in the head, 20 in grippers)
Actuators	MX-106 x 18, MX-64 x 6, MX-28 x 2
Gripper	RH2D
Fream	Aluminium alloy 5052
Surface treatment	Anodising & Sanding
External spur Gears	1:2.3(Roll joints in legs), 1:1.67(Pitch joints in legs, 1:1.5(Hip Yaw joints)
Bearing	ball(flanged type), Idler

five joints, excluding the knee joint on each leg. This parallel structure not only contributes to its reduced weight but also renders it more cost-effective. The knee-less joint design is implemented to control the knee joint through precise calculations of the ankle and hip joint movements. With 38 degrees of freedom, Robinion Sr. has ten degrees in each leg, six in each arm, two in the head, and twenty in the grippers, all of which use 30 actuators. To increase torque and ensure optimal performance, the roll and pitch joints of the legs are equipped with dual motors. Despite such enhancements, the combined torque of the two actuators remains insufficient, necessitating the incorporation of external spur gears to provide the necessary torque, as depicted in Figure 2.4. This mechanical design not only enhances the robot's overall performance but also has cost-efficient construction.

Robinion Sr. deviates from the traditional design employed by similar-sized robots that typically utilize four actuators to control single-pitch joints in the legs, such as ankle and hip joints. In a strategic move to enhance efficiency while minimizing weight and cost, Robinion Sr. employs an approach of dual actuators paired with external spur gears, ensuring robustness and cost-effectiveness across all leg joints. These standard spur gears are gear ratios of 24:55 for the ankle and hip roll joints, 24:36 for the knee pitch joints, and 24:40 for the hip yaw joints. The selection of bearings plays a crucial role in achieving precise control and stabilized walking. Specifically, the backlash of the hip yaw joint significantly impacts the walking gait. To mitigate this challenge, the humanoid robot uses idler bearings to reduce backlash for the hip yaw joint, guaranteeing smoother motions assisted by the incorporation of flange-type ball bearings. The

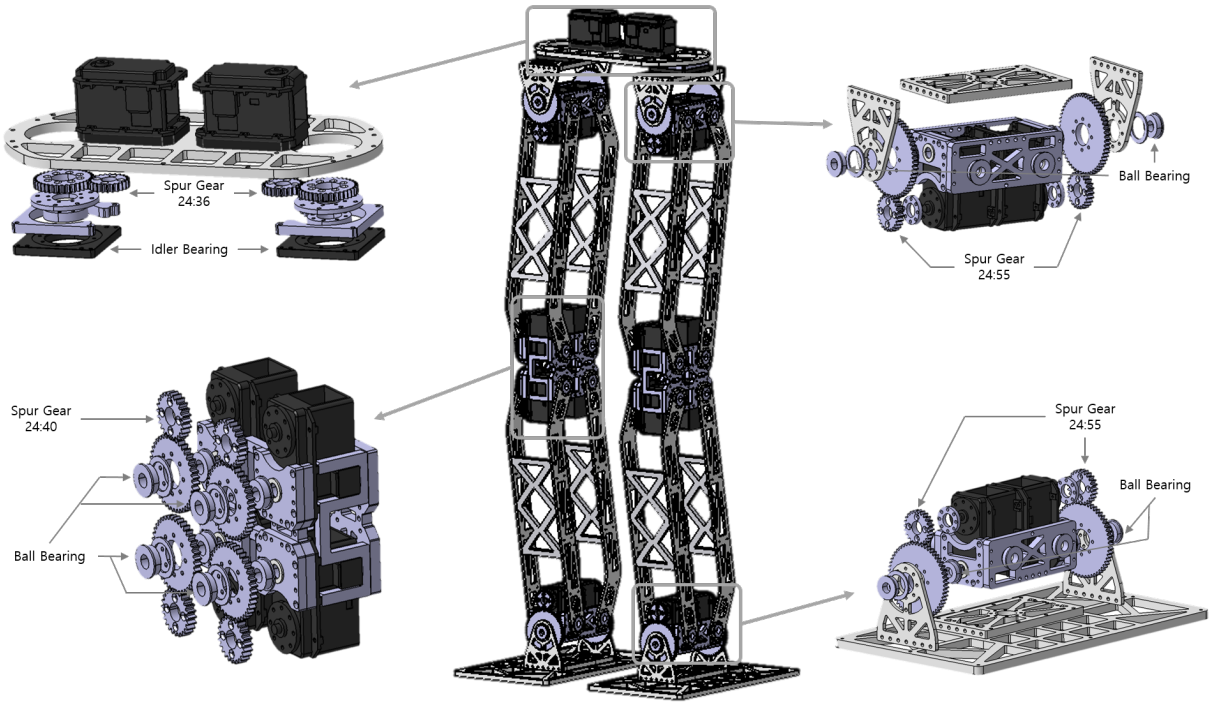


Figure 2.4: Mechanism design of legs with external gears

frame of Robinion Sr. is manufactured using sanded and anodized aluminum, ensuring both strength and lightweight properties. The integration of intelligent actuators from the Dynamixel MX series by ROBOTIS is equipped onto the humanoid robot. The humanoid robot has eighteen MX-106Rs on the legs, six MX-64Rs on the arms, and two MX-28Rs on the head. These features encompass precise position control through PID control, 360-degree positioning, speed control, and torque control via the current sensor, enhancing the robot's overall maneuverability and control capabilities. Moreover, Robinion Sr. is equipped with a gripper known as the RH2D, featuring force control.

2.2.2 Electronics

The electrical element of the humanoid robot encompasses three distinct elements: the main controller, responsible for environmental perception and behavioral management; the motion controller, tasked with inverse kinematics calculation, gait trajectory generation, and intelligent actuator control; and the sensor controller, dedicated to the interpretation of IMU raw data and subsequent Euler angle computation. Table 2.5 provides an overview of the humanoid robot's electronic specifications. This analysis contains details such as the power source, controllers, and sensor system.

Table 2.5: Robinion Sr. electrical specification

Items	Description
Power Source	LiPo battery (14.8V, 6400mAh) Up to 60 min
Main Controller	Tablet PC (Intel Core-m5, 4GB RAM, 128GB SSD, WIFI, Bluetooth) Coral USB Accelerator
Motion Controller	OpenCM9.04 & EXP board (ARM Cortex-M3, 128KB Flash, 20KB SRAM)
Sensor Controller	OpenCM9.04 (ARM Cortex-M3, 128KB Flash, 20KB SRAM)
Sensors	IMU(MPU6050), Camera(Logitec C930e with 150° Wide angle lens)

The electronic system of Robinion Sr. consumes power from a four-cell Lithium Polymer (LiPO) battery with a capacity of 6,400mAh, effectively energizing the motion controller, sensor controller, and intelligent actuators. The electronic system yields up to one hour of battery life during motion control operations. The main controller is a tablet PC, including an Intel Core m5 processor, 4GB of memory, and a 128GB SSD for storage. This tablet PC serves a dual role as the main controller and is improved with a built-in battery to augment the overall power supply longevity. To strengthen its computational power, the robot utilizes a Coral USB accelerator, a cost-effective alternative to high-price GPUs, for high-performance machine-learning inferencing capabilities. This accelerator is instrumental in deploying deep learning techniques to detect objects, including a ball, goalposts, robots, and junctions for autonomous soccer. For efficient control and robust processing, Robinion Sr. uses the OpenCM 9.04[7], an open-source robot controller embedded with an ARM Cortex-M3 microcontroller for the motion and sensor controllers. Its compatibility with the ROS-embedded system provides UART, SPI, and other functions. This system allows the efficient utilization of libraries and programming via the Arduino IDE. The motion controller, extended with the OpenCM 485 EXP[8] extension board, proficiently controls Dynamixel actuators via RS485 and TTL connectors. To optimize computational distribution, the motion controller calculates control commands for inverse kinematics computation and trajectory generation for the walking gait, reducing the computational load on the main controller. The mechanical complexity of the leg structure necessitated the development of a customized analytical inverse kinematics solver utilizing trigonometric formulas. This solver satisfies the 5-DoF parallel system of legs, as opposed to the standard 6-DoF structure,

[7]https://emanual.robotis.com/docs/en/parts/controller/opencm904/
[8]https://emanual.robotis.com/docs/en/parts/controller/opencm485exp/

ensuring precise trajectory calculations. The sensor controller integrates the OpenCM with an MPU6050, combining three-axis gyroscopes and accelerometers. This controller enables stability by capturing six axes of IMU data, which are used to compute Euler angles and assess the robot's balance. With IMU feedback, the closed-loop stable walking gait empowers Robinion Sr. to traverse stably at speeds of up to 25 cm/s. Facilitating inter-component communication, the main controller employs USB connections to relay inverse kinematics parameters to the motion controller and to receive IMU data from the sensor controller. This communication ecosystem culminates in a harmonious orchestration of the robot's functions. Figure 2.3 provides a location of the spatial distribution of these components, elucidating their placements within the robot frame, including controllers within the body, a camera attached to the head, and the LiPo battery installed within the body.

2.3 Robinion2P

This section presents the hardware design of a humanoid robot platform, namely Robinion version 2 with Parallel kinematic leg(Robinion2P), which was developed in 2020 (see Figure 2.5). This section discusses improved mechanical and electrical structures based on development and experience from Robinion version 1[31]. Robinion2P has been improved from the previous version, namely Robinion, regarding the mechanical and electrical parts. In the mechanical design, we enhanced exterior features such as the intelligent actuators, the wiring path, the frame design, and the bearings to reduce backlash. The robot is equipped with a high-performance main controller to improve processing capacity and the Edge TPU to replace heavy GPU for deep learning-based object detection. Furthermore, an RGB-D camera was fitted for various research techniques on visual simultaneous localization and mapping (V-SLAM) and navigation. The platform is also worthy of participating in intelligent humanoid competitions: FIRA Hurocup, Robocup Humanoid League, and IROS Humanoid Robot Application Challenge –Robot Magic and Music (IROS-HAC).

There are many reasonable humanoid robot platforms up to 60cm in small humanoid robots. Darwin-OP3, a humanoid robot developed by ROBOTIS, is the most representative model available for purchase. Developers of humanoid robots are leveraging this platform for extensive research projects. However, few larger platforms exist, such as igus® humanoid, Wolfgang-OP,

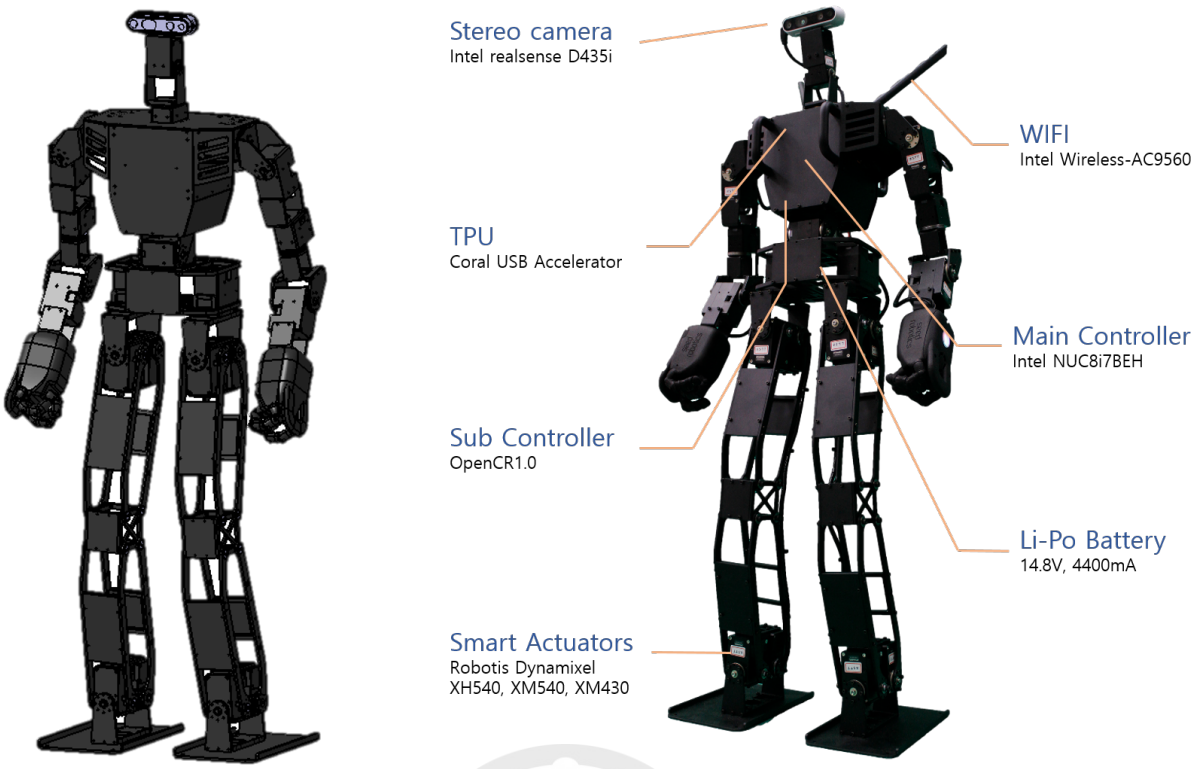


Figure 2.5: 3D design(left) and Robinion2P with position of components(right)

and Nimbro-OP[32, 33, 34]. Some research groups have used the igus® humanoid since its release in 2015. The igus® humanoid robot is a similar-sized robot to Robinion2P. Thus, we also describe a comparison of Robinion2P and the igus® Humanoid Open Platform.

The primary objective of designing the Robinion2P is to develop a lightweight and robust humanoid robot that numerous researchers can use. Most humanoid robot platforms focus on walking gait, recognizing objects, and sensing with various sensors. Robinion2P also focuses on the manipulation task using the 5 degrees of freedom arm with a gripper as well as considering the functions mentioned above. Section 2.3.1 describes detailed information about the mechanical structure. We show enhanced electrical components compared with other humanoid platforms in Section 2.3.2

2.3.1 Mechanics

Robinion2P is an improved design based on Robinion version 1, a humanoid robot platform with a similar mechanical structure. It is an autonomous humanoid robot platform that focuses on various research, such as closed-loop omnidirectional gait, object detection using deep learning algorithms, manipulation tasks, and SLAM for navigation. We consider the mechanical design

to reduce the development cost and time, optimize the wiring path, and widen each joint angle range of our new humanoid robot. We describe the comparison of the mechanical specification of the humanoid robot platforms in Table 2.6. We analyze the mechanical details of Robinion2P in comparison with the similar-sized igus® humanoid robot and the commercial humanoid robot Darwin-OP3.

Table 2.6: Comparison of mechanical specification of the humanoid robot platforms

Type	Specification	Robinion2P	igus®	Darwin-OP3
Dimension	Height(mm)	920	920	510
	Weight(kg)	7.0	6.6	3.5
Kinematic structure	Leg	Parallel	Standard	Standard
Degrees of freedom	Total	41	20	20
	Head	2	2	2
	Arm	10	6	6
	Hand	18	-	-
	Torso	1	-	-
	Leg	10	12	12
Actuators	Model name	XH540, XM540 XM430	MX106, MX64	XM430
Frame	Material	Aluminium alloy 5052	Polyamide 12 (PA12)	Aluminium alloy
	Surface treatment	Anodising Sanding	-	N/A
	Extra Bearing	Needle thrust	-	-
Cost	Total(USD)	9,500	N/A	11,000

Robinion2P, 920mm in height and 7.8kg in weight, is a suitable size to apply various research, join the robot competitions, and use the robot in a human environment(see Figure 2.6). The Robinion series is a humanoid robot platform with high degrees of freedom compared to similar-sized humanoid robots. The standard mechanism on humanoid robots is to have six joints on each leg. With our knee-less design, however, we reduce the total number of actuators on the leg of the humanoid robot. The legs use a parallel mechanism that has the advantage of reducing the weight and cost of the robot. Specifically, the ankle joint and the hip joint control the knee joint. There are 41 degrees of freedom(DoF) in total on the humanoid robot: five in each leg, one in the torso, four in each arm, ten in each gripper, and two in the head for various manipulation tasks as well as walking. The joints throughout the whole body are roll and pitch

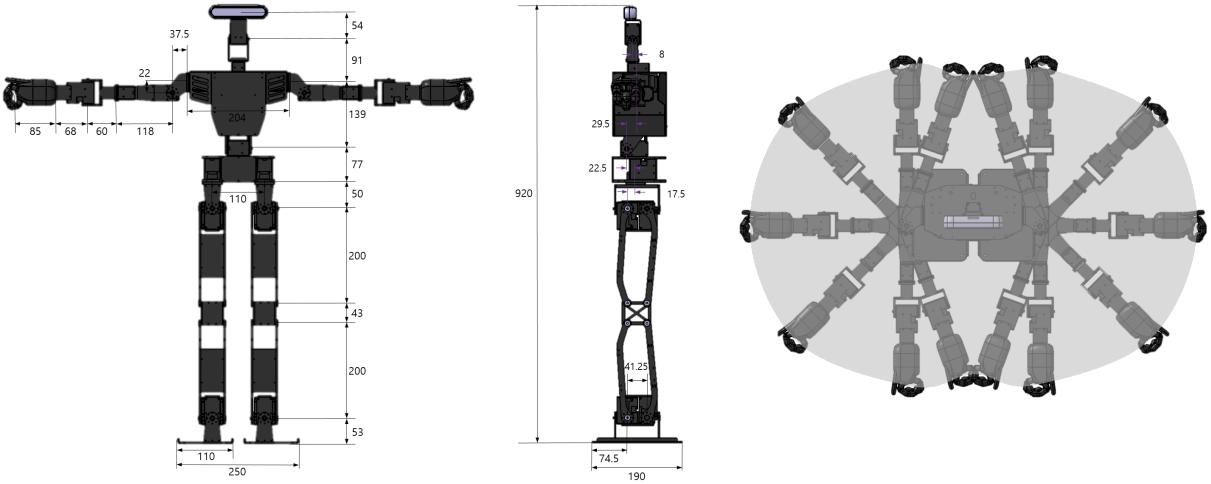


Figure 2.6: Dimension and manipulating workspace

joints in each ankle, roll, pitch, and yaw joints in each hip, roll, and pitch joints in each shoulder, pitch and yaw joints in each elbow, yaw joint in each wrist, and roll and pitch joints in the head. Robinion2P robot is relatively light considering its high DoF, and with five DoF on each arm, it can move around a large workspace for manipulation tasks(see Figure 2.6). Robinion2P has a horizontal reach from the center of the base to the gripper of up to 470mm and a total span of up to 940mm, allowing manipulation over a wider range than its height. A new series of smart actuators manufactured by Robotis, Dynamixel X, is used for all joints. The MX series actuator used by the igus® humanoid robot is the previous version of Dynamixel, and the X series has improved size, performance, and convenience compared to the MX series. The Dynamixel X series improves torque by almost 30 percent and reduces volume by about 32 percent compared to the MX series. The actuator adopts the aluminum case with a tapped hole on all sides for easy assembly and the new technology, such as the hollow cable fastening structure for an excellent wiring path. There are six different control modes: torque control, velocity control, position control, extended position control, current-based position control, and PWM control. An aluminum frame with sanding and anodizing improves strength while making it lighter. The needle thrust bearings are designed for precision manipulation tasks of the pitch joint in each shoulder and the yaw joint in each elbow. Although Robinion2P has a significantly enhanced mechanical and electrical structure compared to Darwin-OP3, the manufacturing price is less than the price of OP3. It is advantageous to manufacture a high-performance humanoid robot such as a robot similar to Robinion2P[35].

Robinion2P is equipped with an advanced gripper named the RH2D from Seed Robotics for drawing pictures or performing magic tricks. The gripper has smart actuators and control systems, including magnetic finger protection, Dyneema (Kevlar fiber) reinforced tendons, and 3-segment fingers. A compact and smaller form factor makes this gripper ideal for power grasping. This product provides a robust weight-to-payload ratio, with only 175g but up to 400g payload capacity. Its distance sensor can detect objects on a palm to measure their proximity. In addition, a high-resolution current sensor in the gripper can estimate force.

2.3.2 Electronics

Figure 2.7 shows an electrical diagram of Robinion2P. In the electrical system, two controllers work together: the main controller, which recognizes the environment and manages the behavior of Robinion2P, and the sub-controller, which controls actuators and reads data from the IMU. The main controller reads frames and measures distances of objects for various research, such as autonomous soccer play, magic tricks, and V-SLAM from the stereo camera via USB. It connects a neural network accelerator (Edge TPU) to detect objects using deep learning algorithms instead of the GPU. The Edge TPU is not only a low-cost machine (60 USD) but also delivers good performance with deep learning algorithms. The sub-controller controls actua-

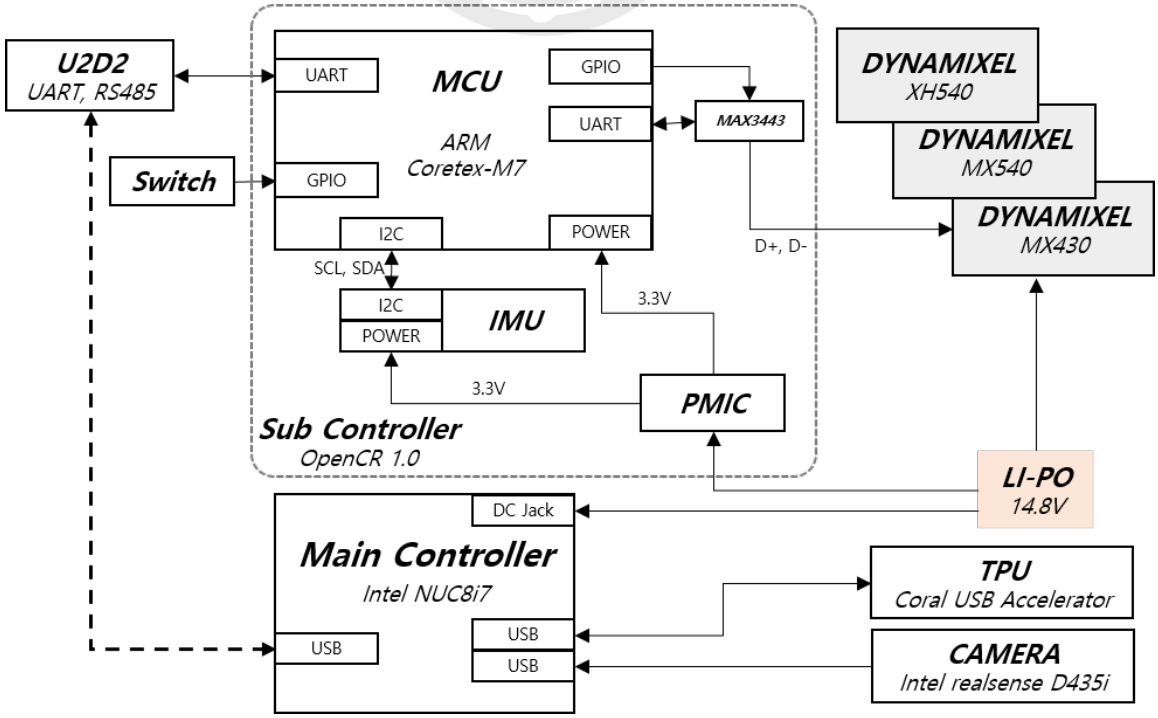


Figure 2.7: Electrical diagram of Robinion2P

tors, reads the IMU sensor information, and calculates the sensor data. U2D2 is a small-sized USB communication converter that supports TTL, RS-485, and UART. The main controller communicates with the sub-controller via U2D2.

Table 2.7: Comparison of electrical specification of the humanoid robot platforms

Type	Specification	Robinion2P	igus®	Darwin-OP3
Power Source	Battery	4-cell LiPo (14.8V, 4.4Ah)	4-cell LiPo (14.8V, 3.8Ah)	3-cell LiPo (11.1V, 1.8Ah)
	Battery Life	Up to 50 min	Up to 30 min	Up to 30 min
Main Controller	Model	Intel NUC NUC8i7BEH	Gigabyte Brix GB-BXi7-5500	Intel NUC N/A
	CPU	i7-8559U	i7-5500U	i3
	Memory	4GB	4GB	8GB
	Network	Ethernet Wi-Fi Bluetooth	Ethernet Wi-Fi Bluetooth	Ethernet Wi-Fi Bluetooth
Sub Controller	Model	OpenCR	CM730	OpenCR
	MCU	Coretex-M7	Coretex-M3	Coretex-M7
	Memory	1MB Flash 320KB SRAM	512KB Flash 64KB SRAM	1 MB Flash 320KB SRAM
Sensors	Position sensor	AS5045(12bit)	AS5045(12bit)	AS5045(12bit)
	IMU	6-Axis (ICM-20648)	6-Axis (L3G4200D, LIS331DLH)	9-Axis (MPU9250)
	Camera	Intel Realsense D435i	Logitech C905	Logitech C920
	Extra Lens	-	150°FOV	-
Neural Network Accelerator		Coral USB Accelerator (Edge TPU)	-	-

Table 2.7 shows the comparison of electrical specifications of igus® humanoid robot platform and Darwin-OP3. As shown in the table 2.6, Robinion2P has the highest degree of freedom and has the most up-to-date electrical components to be described in this section, but can be utilized in various types of research at an affordable development price. Robinson2P features an Intel NUC8i7BEH for its main controller, supporting the Ubuntu 20.04 operating system. The main controller features an Intel Core i7-8559U quad-core processor operating at a maximum frequency of 2.7GHz. The system comes with 8GB of RAM and a 128GB M.2 solid-state drive. It includes four USB 3.1 ports and a header socket for two internal USB 2.0. For access to other

robots and joysticks, the main controller has WIFI (802.11 ac) and Bluetooth 5.0. OpenCR 1.0, an open-source robot control system that takes advantage of the ARM Cortex-M7, is used in our humanoid robot for the sub-controller. The controller developed for ROS-specialized embedded systems supports RS-485 and TTL to control Dynamixel and provides various communication environments such as UART, CAN, and SPI. In addition, this controller has an integrated IMU, which has three axes gyroscopes, three axes accelerometers, and three axes magnetometers for measuring attitude. In order to detect objects and measure distance in real-time, an Intel Realsense D435i stereo camera is installed on the head. It contains an integrated IMU. It is possible to achieve the SLAM tasks using some open-sourced ROS packages. Advanced perception systems commonly use Deep Learning algorithms. Since most of the algorithms are complex and have many parameters, the training and inferencing require a lot of computation. The most common hardware used is the GPU. Using Google's Edge TPU low-cost device, we are able to circumvent the need for a dedicated GPU while still achieving real-time inference speeds. The compact size and low price of Edge TPU make it an ideal choice for lightweight systems such as Robinion2P, which are difficult to mount a GPU. A key feature of Robinion2 that sets it apart from other humanoid robot platforms is its ability to use deep learning-based algorithms in real-time.

2.4 Robinion2S

A renewed Robinion2S based on Robinion2P is designed to suit the balance board with pitch and roll axes. There is the parallel kinematic design on the legs of the Robinion2P, which keeps the pitch axis of the feet parallel to the ground at all times. However, the parallel structure is inevitably limited to movement on a balance board with roll and pitch axes. We modified the leg design of Robinion2P from the parallel kinematic leg to the standard kinematic leg to solve the issue. The standard kinematic mechanism includes six degrees of freedom for each leg, typically utilized on humanoid robots and allowing a more generous range than a parallel kinematic structure with five degrees of freedom. In Robinion2S, all mechanical structures and electrical components except the legs design are provided the same as in Robinion2P. Without going into detail about mechanics and electronics, we briefly illustrate the leg structure of Robinion2S in this section.

As shown in figure 2.8, Robinion2P and Robinion2S are designed in the 3D cad tool. It is worth noting that both the standard kinematics and the parallel kinematics have their own advantages and limitations, and the choice of leg structure depends on the specific requirements of the humanoid robot's intended applications, such as locomotion efficiency, stability, payload capacity, specific tasks, and control complexity. The parallel kinematic leg involves a leg structure where multiple kinematic chains are connected in parallel [36, 37]. In this configuration, the base of the leg connects to multiple branches, each consisting of its own set of joints and links. These branches are then connected to the next joint to rotate parallel direction. The parallel kinematics for a humanoid robot leg offers certain advantages. It often provides enhanced stability, as the load can be distributed among multiple branches, reducing the stress on individual joints. The parallel structure is also able to exhibit higher payload capacities and improved overall stiffness. However, controlling parallel kinematics can be more challenging, as the movements of multiple branches need to be coordinated simultaneously. The standard kinematic leg, also known as the serial leg mechanism, refers to a leg structure where the joints are arranged in a serial configuration, forming a chain-like structure [38, 39, 40, 41]. Each joint connects to the next joint linearly, leading from the base to the end effector, which is a robot foot

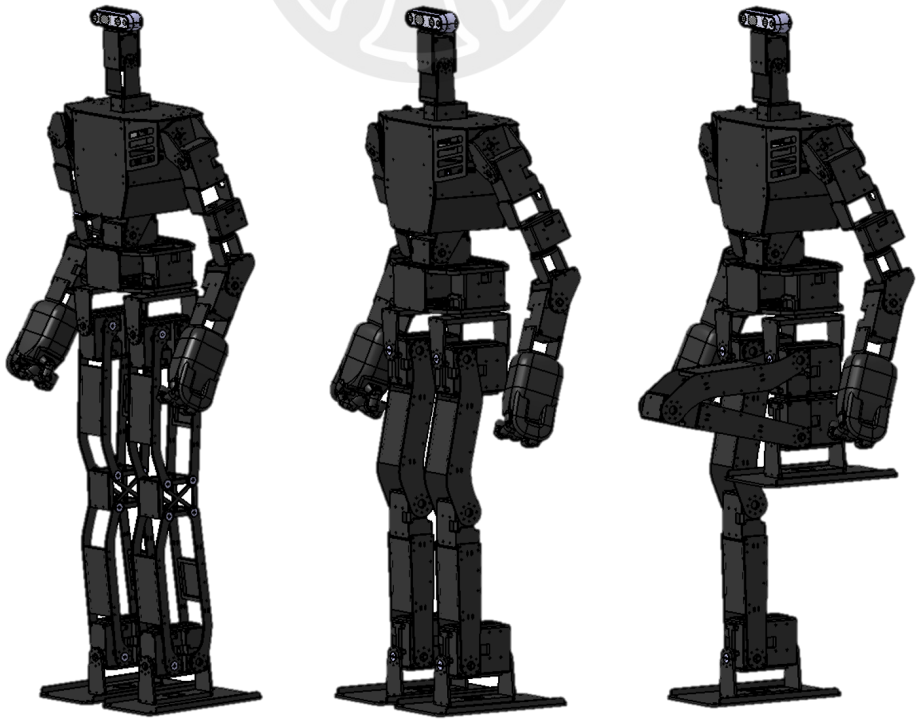


Figure 2.8: 3D Mechanical design of Robinion2P & Robinion2S (Left: Robinion2P, Ceter: Robinion2S, Right: Robinion2S Pitch range of leg)

of the leg. The standard kinematics is the leg structure of a human being, where the hip joint connects to the thigh, which connects to the knee joint, followed by the shin, and finally, the ankle joint connects to the foot. Each joint is actuated individually in humanoid robots with standard kinematics, allowing for precise control of joint angles and movement. This configuration enables more natural walking patterns and dynamic movements, similar to human locomotion. The standard kinematic leg with a human-like leg structure is more efficient than the parallel kinematic leg in overcoming highly dynamic environments such as a balance board.

2.5 Kinematics analysis

The stability of walking is a fundamental component of humanoid robots. Achieving stable walking requires intricate mathematical methodologies such as kinematics, dynamics, the Zero Moment Point (ZMP), the Central Pattern Generator (CPG), and diverse feedback control theories. Mathematical models, which are mentioned, work in unison to regulate and control each phase of the walking gait of humanoid robots, ensuring balance and consistency in movement. Kinematics refers to the mathematical description of motion without considering the forces causing the motion. It provides the foundation for understanding the interplay of joints and segments in a robot, as well as their relationship to the robot's overall posture and movement. On the other hand, dynamics dives deeper into the motion equation by considering the forces and torques causing the motion. The Zero Moment Point (ZMP) is a significant concept in the stability control of bipedal robots. It is the point on the ground where the resultant moment of the inertial forces and the gravitational forces is zero [42]. The location of ZMP is crucial in maintaining balance; if it stays within the support area of the robot's foot or feet, the robot remains stable. The Central Pattern Generator (CPG) is vital in generating rhythmic pattern signals, typically used for controlling locomotion in many living organisms and, by extension, in robotics [43]. In the case of humanoid robots, these CPGs can be used to create rhythmic oscillations necessary for repeatable tasks such as walking. Lastly, feedback control theories are employed to respond to the motion deviations of the robot from its desired trajectory. Sensors gather data about the robot's position and velocity, and any discrepancies are corrected through a feedback loop, thus maintaining the stability and accuracy of the motions.

The usual design for leg structures in most humanoid robots is based on standard kine-

matics, which comprises six degrees of freedom: roll and pitch joints at the ankle, a pitch joint at the knee, and roll, pitch, and yaw joints at the hip [44]. The six degrees of freedom allow for extensive mobility and adaptability in humanoid robots (see Figure 2.9's left). The roll and pitch joints at the ankle allow the robot to maintain balance on uneven terrain by tilting the foot side-to-side (roll) and forward-and-backward (pitch). The ankle joints of humanoid robots emulate the flexible movements of a human ankle, promoting stability and facilitating walking, running, or climbing. The pitch joint at the knee permits bending and extending movements, akin to human knee motion. The knee joint plays a critical role in producing a natural-looking gait and supporting activities such as stepping over obstacles or climbing stairs. It enables humanoid robots to bend its knee for the swing phase of walking and straighten it again for the stance phase. The roll, pitch, and yaw joints at the hip give humanoid robots a broad range of motion. The roll joint enables robot legs to swing outward or inward, supporting balance, while the pitch joint allows the leg to swing forward and backward, powering locomotion. The yaw joint facilitates the rotation of the leg, allowing the robot to turn and pivot. However, the joints in most Robinion series of robots operate on parallel kinematics, allowing five degrees of freedom for each leg: roll and pitch joints at the ankle, and roll, pitch, and yaw joints at the hip (see Figure 2.9's right). In contrast to the standard kinematic design, the parallel kinematic design in the Robinion series removes one degree of freedom at the knee, resulting in unique movements. The roll and pitch joints at the ankle still provide the ability to maintain balance on uneven terrain, and the roll, pitch, and yaw joints at the hip continue to give the robot an extensive range of motion. Despite having one less degree of freedom, parallel kinematics offer advantages in other areas. For one, parallel kinematics can potentially increase the structural rigidity of the robot, as the forces and stresses are distributed over multiple joints. The parallel structure also improves the precision of the robot's movements due to lower mechanical backlash, essentially the slight movement that can occur in mechanical systems when the direction of motion is reversed. Furthermore, the five degrees of freedom design also allows for the more straightforward calculation of inverse kinematics, which can contribute to increased efficiency in processing and less strain on computational resources. It can, therefore, result in smoother and more synchronized movements in real-time, particularly beneficial for the Robinion series' walking and maneuvering capabilities.

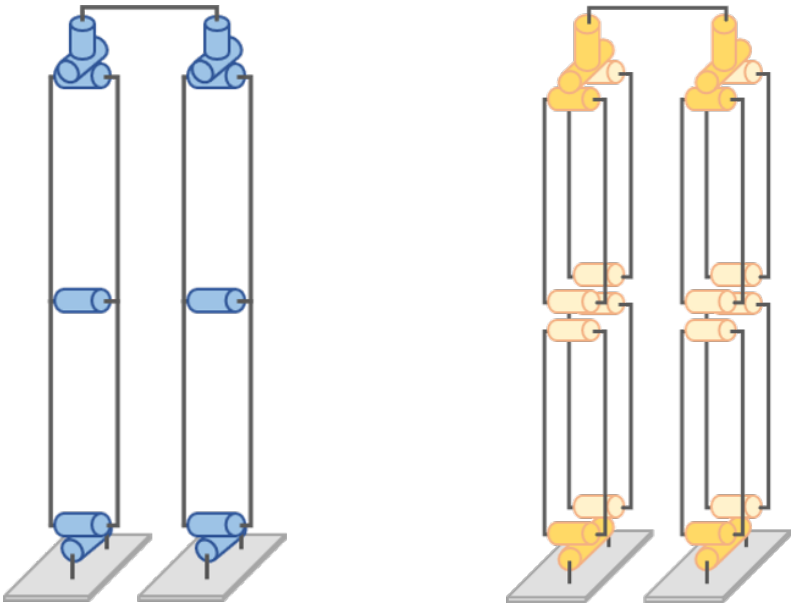


Figure 2.9: Humanoid robot kinematics structure for leg design (Left: standard kinematics, Right: Parallel kinematics). Blue and Yellow components represent actuators

Developers working with humanoid robots typically compute the forward and inverse kinematics of the standard kinematic leg structure utilizing various available libraries. Nevertheless, for the parallel kinematic leg structure, utilizing a standard library, like the Kinematics and Dynamics Library (KDL), is not straightforward [45]. The complexity arises from the inherently different mathematical frameworks underlying standard and parallel kinematics. Standard kinematic models are often easier to solve due to their consisting of a sequence of joints, with each one only affecting the position and orientation of the parts. This leads to a relatively straightforward formulation of both the forward and inverse kinematics problems, which can readily be solved with standard libraries such as KDL. In contrast, in parallel kinematics, each joint is potentially available to affect each other for the position and orientation of the robot. The interdependence between each joint creates a more complex system of equations for the kinematics problems, which is challenging to solve using standard libraries. Consequently, developers need to develop custom algorithms or employ specialized libraries tailored for parallel kinematics. The complexity of solving kinematics does not mean which use of parallel kinematics in humanoid robots is impossible or impractical. On the contrary, parallel kinematics offer advantages such as improved stiffness and precision, as mentioned earlier. But it does mean that developers need to invest more time and effort into handling the kinematic calculations for humanoid robots.

The purpose of this section is to describe the inverse kinematics solution for leg motions of the standard and parallel kinematic structures. The inverse kinematics solution is critical for determining the joint parameters that yield a desired position or movement for the robot. This mathematical model allows the control system to adjust each joint precisely, enabling the robot to navigate a specific path or perform a particular action. Whether standard or parallel kinematics are employed, the ultimate goal of the inverse kinematics solution is to produce smooth, natural, and controlled leg movements for the humanoid robot. By determining the precise joint angles needed for each desired foot position, humanoid robots navigate various terrains, avoid obstacles, and accomplish tasks with more agility and efficiency. For the implementation, it has created a lightweight analytic inverse kinematics solver, designed explicitly for leg movement, that operates on a Micro Control Unit (MCU). This specially designed solver is key to improving the robot's efficiency and speed. Since the MCU is a compact computing unit that uses little power, it is ideal for controlling precise movements in real-time. The solver functions by determining the specific joint angles required to achieve desired leg motions, thus enabling the robot to walk, climb, or perform any other leg-based activity. This lightweight analytic inverse kinematics solver is crafted to deliver high computational efficiency. The lightweight analytic method refers to its streamlined code and algorithms that minimize the computational resources needed, making it suitable for the limited processing power of an MCU. Moreover, it uses mathematical techniques to find exact solutions for the kinematic equations, as opposed to numerical methods that would require iterative computations and more processing time. Having the solver run directly on the MCU also significantly reduces latency. Since the MCU is embedded in the robot, it processes the outputs and adjusts the movements almost instantly. This real-time performance is vital for the robot to maintain its balance, react quickly to changes in its environment, and smoothly perform tasks requiring precise leg movements.

2.5.1 Inverse kinematics

The walking gait of humanoid robots depends upon the position and orientation of the feet. The inverse kinematics principle establishes the angle of the leg joints relative to the foot position. Analytic and numerical methods provide two different approaches to solving the inverse kinematics problem for humanoid robot legs. Numerical inverse kinematics involves iterative

methods to approximate the solution of the joint parameters [46]. The numerical approach does not provide explicit formulas for the solution. Still, it uses optimization techniques to progressively refine an initial guess until it reaches a suitable solution or meets a specified level of accuracy. One popular numerical method for inverse kinematics is the Jacobian-based method. The Jacobian matrix represents the first derivatives of the position and orientation of the end-effector with respect to the joints. The method converges on a solution by iteratively adjusting the joint parameters in the direction that reduces the error between the end-effector's current and desired positions. Numerical methods handle more complex situations and provide solutions where analytic methods might fail. However, they can be slower than analytic methods and sometimes may not converge to a solution, especially without a good initial setting.

Analytic inverse kinematics in the context of a humanoid robot leg involves determining the joint parameters, such as angles that would place the robot's foot at a specific location in three-dimensional space [16]. This is crucial for enabling actions such as walking, running, and climbing. In humanoid robots, a leg typically involves several joints, often including (but not limited to) roll and pitch joints at the ankle, a pitch joint at the knee, and roll, pitch, and yaw joints at the hip. To calculate the required joint parameters, the desired foot position is given in terms of its coordinates and orientation. The analytic inverse kinematics indicates that exact mathematical solutions are used to solve the inverse kinematics problem. This typically involves creating a mathematical model of the robot's leg structure based on its geometry and the types of joints used. Trigonometric equations are commonly used in these models, as they are available to represent the rotational movements of the joints accurately. Once the model has been established, it solves the inverse kinematics problem by inserting the desired foot position and orientation into the model and solving for the joint parameters. The analytic solver is quite complex, especially when there are many joints involved and the possibility of multiple solutions exists. However, it has the advantage of providing precise solutions in real-time, which is crucial for controlling humanoid robot motions. The mathematical model and the solution to the inverse kinematics problem are usually implemented in a computer program, which can quickly calculate the required joint parameters given a desired foot position. The calculated joint parameters control the leg movements of the humanoid robot, enabling it to perform actions such as walking or running. In practice, a combination of both methods is used. An initial solution

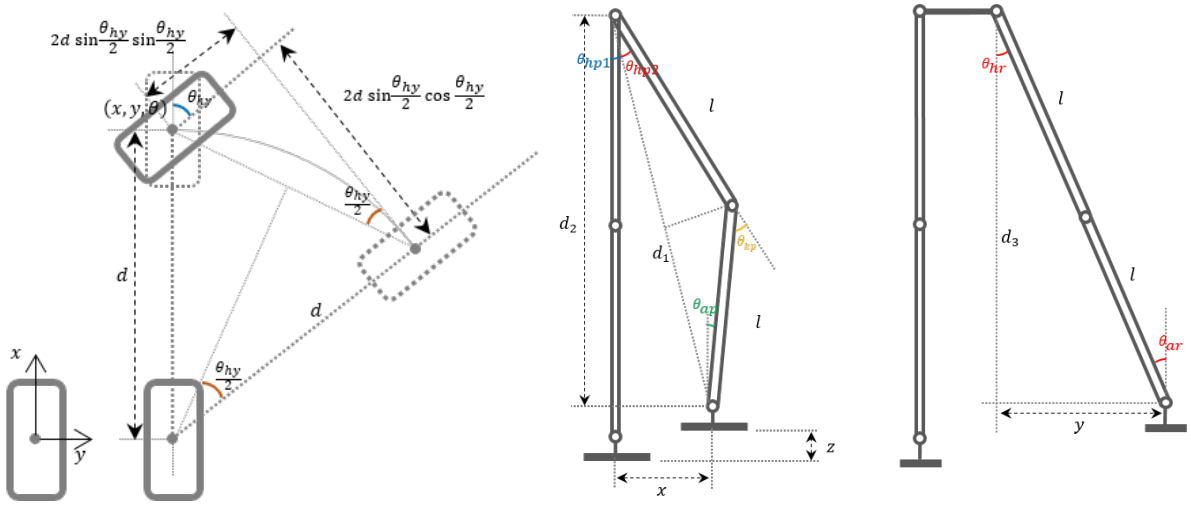
might be quickly found using an analytic approach, then refined with a numerical method. This hybrid approach provides both speed and accuracy when controlling the movements of a humanoid robot leg. However, this section analyzes the analytic inverse kinematics solutions of standard and parallel leg structures in detail.

2.5.2 Inverse kinematics for standard leg mechanism

The inverse kinematics problem involves determining the angles of the joints given a certain position and orientation of the end-effector. The goal is to find the joint angles that allow the end-effector to reach the desired position and orientation. This section solves the analytic inverse kinematics with the standard leg structure of a humanoid robot, which has six degrees of freedom: roll and pitch joints at the ankle, pitch joint at the knee, and roll, pitch, and yaw joints at the hip. The equations for the analytic solution of the inverse kinematics problem for a standard leg structure are typically based on trigonometry and geometry. Computing the inverse kinematics for a standard structure, like a humanoid robot leg, analytically is more straightforward compared to a parallel structure due to using inverse kinematics libraries effortlessly.

Figure 2.11 provides a visual representation along with numerical values and details regarding each joint crucial in calculating the inverse kinematics solution. Each joint is identified by θ and the associated subscript that defines its placement within the robot's leg structure. θ_{hy} represents the yaw joint at the hip, allowing the leg to rotate around a vertical axis. θ_{hr} corresponds to the roll joint at the hip, which facilitates the leg's lateral movement. θ_{hp} is the pitch joint at the hip, enabling the forward and backward swinging of the leg. θ_{kp} stands for the knee pitch joint, which lets the lower part of the leg fold in or stretch out in relation to the thigh. θ_{ap} denotes the pitch joint at the ankle. Lastly, θ_{ar} represents the roll joint at the ankle, controlling the foot's side-to-side motion. The coordination of each joint is essential for producing a fluid and stable walking gait for the humanoid robot.

As illustrated in Figure 2.10a, the proposed inverse kinematics solution employs a unique approach by redefining the conventional x and y values as $\hat{x}$ and $\hat{y}$, respectively, taking into account the motion in the hip yaw rotation angle. This process redefines the z value by calculating the y-direction of the foot, which affects the height of the foot from the ground. As



(a) $\hat{x}$, $\hat{y}$ related with hip yaw angle (b) Pitch joints (c) Roll joints

Figure 2.10: Joints of standard leg mechanism to solve inverse kinematics

highlighted in Figure 2.10c, this change in z prompts us to redefine it as $\hat{z}$ using the relationship $d_3 = 2l\cos\theta_{hr}$. Thus, in this novel inverse kinematics solution, the foot's x, y, and z coordinates are reassessed as per Equation 2.1. This redefinition is based on the yaw angle at the hip joint along with the roll angles at the hip and ankle joints. With the redefined coordinates, the inverse kinematics process then goes on to compute the angle for each joint of the leg. The calculations leverage the x, y, and z values, enabling the leg joints to reach the desired pose.

$$
\begin{aligned}
d &= \sqrt{x^2 + y^2} \\
\hat{x} &= x - 2d\sin\left(\frac{\theta_{hy}}{2}\right)\sin\left(\frac{\theta_{hy}}{2}\right) \\
\hat{y} &= y - 2d\sin\left(\frac{\theta_{hy}}{2}\right)\cos\left(\frac{\theta_{hy}}{2}\right) \\
\hat{z} &= z - (2l - 2l\cos\theta_{hr})
\end{aligned}
\tag{2.1}
$$

The analytic inverse kinematics solver proposed considers both the positions of the legs and the yaw orientation of the hip joint to derive a walking gait. The holistic approach ensures that the entire leg configuration is considered, leading to more accurate movement. As depicted in Figure 2.10, the solver operates by inserting the given values of x, y, z, and θ into Equation 2.2. x, y, and z represent the target position of the foot in the coordinate system of the leg, while θ refers to the target orientation. Upon applying these values to the equation, the solver is able to calculate the corresponding joint angles for the hip yaw (θ_{hy}), hip roll (θ_{hr}), hip pitch (θ_{hp}), knee

pitch (θ_{kp}), ankle pitch (θ_{ap}), and ankle roll (θ_{ar}). The computed angles guide the orientation of the actuators, facilitating the humanoid robot to perform a walking gait.

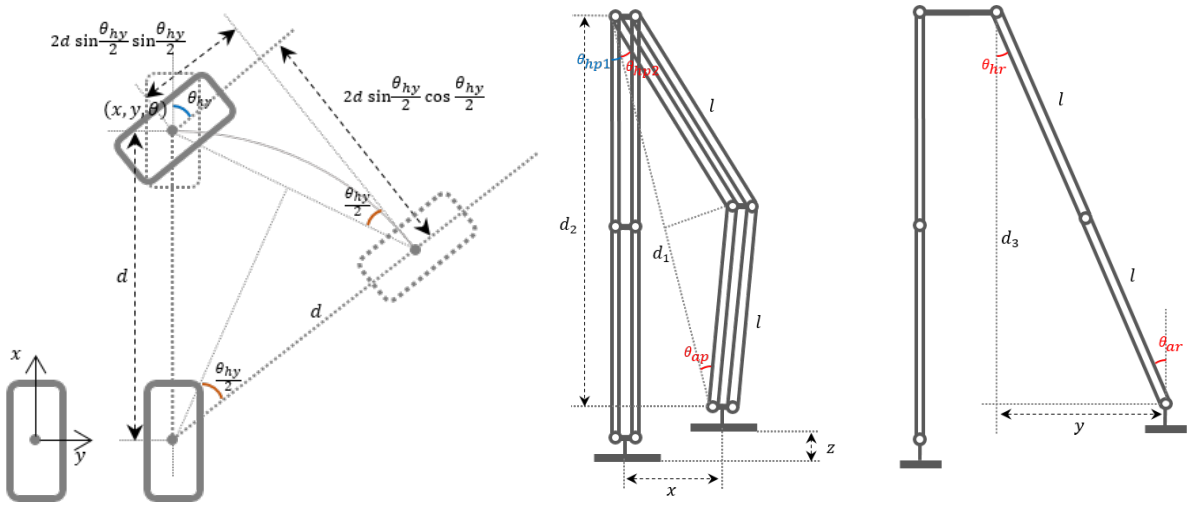
$$
\begin{aligned}
\theta_{hp1} &= \arctan\left(\frac{\hat{x}}{2l-\hat{z}}\right) \\
\theta_{hp2} &= \arccos\left(\frac{\sqrt{(2l-\hat{z})^2+\hat{x}^2}}{2l}\right) \\
\theta_{hp} &= \theta_{hp1}+\theta_{hp2} = \arctan\left(\frac{\hat{x}}{2l-\hat{z}}\right)+\arccos\left(\frac{\sqrt{(2l-\hat{z})^2+\hat{x}^2}}{2l}\right) \\
\theta_{kp} &= 2*\theta_{hp2} = 2\arccos\left(\frac{\sqrt{(2l-\hat{z})^2+\hat{x}^2}}{2l}\right) \\
\theta_{ap} &= \theta_{hp2}-\theta_{hp1} = \arccos\left(\frac{\sqrt{(2l-\hat{z})^2+\hat{x}^2}}{2l}\right)-\arctan\left(\frac{\hat{x}}{2l-\hat{z}}\right) \\
\theta_{hr} &= \arctan\left(\frac{\hat{y}}{2l-\hat{z}}\right) \\
\theta_{ar} &= \theta_{hr}
\end{aligned}
\tag{2.2}
$$

The Robinion2S, one of the latest models in the Robinion series, applies the outputs of Equation 2.2 to determine the specific angle for each joint. The application allows the robot to navigate toward the desired pose, which includes the precise positioning and rotation of each foot. It has the ability to merge forward, backward, lateral, and even rotational walking for the omnidirectional walking gait through a combination of x, y, z, and θ_{hy} values.

2.5.3 Inverse kinematics for parallel leg mechanism

Computing the analytic inverse kinematics for a parallel leg structure of a humanoid robot can be complicated, depending on the leg design structure. The parallel mechanisms tend to have a more complex relationship between joint and end-effector spaces than the standard kinematic structure. The parallel kinematic configuration of humanoid robot legs, designed for the Robinion series, has the roll and pitch joints at the hip and the ankle and disregards a joint at the knee. Assuming the desired position and orientation for each foot are given the same as the standard mechanism solver in the Cartesian coordinate system as x, y, z, and θ, where x, y, and z are the position of the foot, and θ is the yaw angle. For the parallel kinematics leg structure, we implement the analytic inverse kinematics solver for a similar kinematics leg structure of

the Robinion series. Figure 2.11 presents numerical values and joint information to solve the inverse kinematics (θ_{hy}: hip yaw joint, θ_{hr}: hip roll joint, θ_{hp}: hip pitch joint, θ_{ap}: Ankle pitch joint, θ_{ar}: ankle roll joint). The principle is the same as the standard kinematics leg structure.



(a) $\hat{x}$, $\hat{y}$ related with hip yaw angle (b) Pitch joints (c) Roll joints

Figure 2.11: Joints of parallel leg mechanism to solve inverse kinematics

Figure 2.11a illustrates how the hip yaw rotation angle impacts the spatial orientation of the leg, actually modifying the coordinates x and y into renewed coordinates denoted as $\hat{x}$ and $\hat{y}$. As the foot progresses along the y-axis, the changes in height for the z value are illustrated in Figure 2.11c, prompting the need to redefine z as $\hat{z}$. The inverse kinematics solution proposed takes into account these redefined coordinates for each foot, as demonstrated in Equation 2.1. This redefinition is influenced by the hip joint yaw angle and the roll angles of both the hip and ankle joints. After the coordinates are transformed, it becomes feasible to calculate the angle of each leg joint utilizing inverse kinematics based on the values of $\hat{x}$, $\hat{y}$, and $\hat{z}$. This redefinition of coordinates is a vital step in this inverse kinematics solution, enabling a more accurate representation of each foot position relative to the varying angles of the hip and ankle joints. The method is crucial in achieving a stable and efficient walking gait for humanoid robots. By transforming the standard Cartesian coordinates into a coordinate system that considers the robot's joint angles, this approach allows for more precise control of the leg motion, improving the overall locomotion capabilities of the humanoid robot.

The proposed inverse kinematics solution not only takes into account the position of the foot but also incorporates the yaw orientation in its computations, thereby providing a more

comprehensive understanding of the walking gait. As demonstrated in Figure 2.11, the input values of x, y, z, and θ (representing the desired foot position and yaw orientation) are fed into Equation 2.3. The result of this calculation provides the respective joint angles: θ_{hy} for the hip yaw, θ_{hr} for the hip roll, θ_{hp} for the hip pitch, θ_{ap} for the ankle pitch, and θ_{ar} for the ankle roll. By solving these joint angles, the feet of the humanoid robot are accurately directed to desired positions. The solution ensures the precision of each footstep and allows the trajectory of the foot movement to align closely with the planned path, resulting in a more fluid and natural walking motion. This approach is particularly beneficial in enhancing the ability to adapt to complex environments or undertake tasks that require precise foot placement.

$$
\begin{aligned}
\theta_{hp1} &= \arctan\left(\frac{\hat{x}}{2l-\hat{z}}\right) \\
\theta_{hp2} &= \arccos\left(\frac{\sqrt{(2l-\hat{z})^2+\hat{x}^2}}{2l}\right) \\
\theta_{hp} &= \theta_{hp1}+\theta_{hp2} = \arctan\left(\frac{\hat{x}}{2l-\hat{z}}\right)+\arccos\left(\frac{\sqrt{(2l-\hat{z})^2+\hat{x}^2}}{2l}\right) \\
\theta_{ap} &= \arccos\left(\frac{\mid \hat{x} \mid}{\sqrt{(2l-\hat{z})^2+\hat{x}^2}}\right)+\theta_{hp2}-\frac{\pi}{2} \\
\theta_{hr} &= \arctan\left(\frac{\hat{y}}{2l-\hat{z}}\right) \\
\theta_{ar} &= \theta_{hr}
\end{aligned}
\tag{2.3}
$$

The Robinion series, equipped with a parallel leg mechanism, utilizes the results of Equation 2.3 to determine the angle for each joint required to achieve the target pose. The result of the solver includes the position and rotation of each foot, thus ensuring accurate foot placement and alignment. With the analytic inverse kinematics solver, omnidirectional walking is achievable by combining specific values of x, y, z, and θ_{hy}. This combination of values allows for various walking modes: forward, backward, lateral, and rotational. The step size, or how much ground each step covers, is dictated by the value of x. By substituting a positive value into x, forward walking is facilitated, while inserting a negative value makes backward walking feasible. The rotational direction hinges on the θ_{hy} value, corresponding to the yaw angle at the hip joint. Therefore, by adjusting this value, the humanoid robot rotates as needed. Con-

sequently, by deftly manipulating these values, which are x, y, z, and θ_{hy}, the robot is capable of omnidirectional movement. This includes a combination of forward, backward, lateral, and rotational walking. This flexibility in walking patterns enables the robot to adapt to a wide range of terrains and carry out complex navigation tasks with ease and precision.

2.6 Gait Generation

The walking trajectory for humanoid robots is typically generated using various methods that consider stability, smoothness of the gait, energy efficiency, and the capability to navigate complex environments. Designing a walking trajectory for humanoid robots involves complex computations and the application of various methodologies. A few standard techniques are utilized to generate walking trajectories for the humanoid robot. One of the most widely adopted techniques in humanoid robotics, Zero Moment Point (ZMP), is based on maintaining the robot's stability during locomotion. The ZMP is the point on the ground where the total moment of the inertial and gravitational force acting on the robot is zero. The robot balance can be maintained by planning a trajectory that keeps the ZMP within the support area, which is the area under the feet. And Central Pattern Generator (CPG) is used to generate the basic rhythmic patterns for walking. The generated patterns are adjusted in real-time to respond to changes in the environment or the robot's state. Preview Control of ZMP is an advanced method that generates a walking trajectory by predicting the following ZMP positions. The goal is to minimize the deviation of the ZMP from its reference trajectory while also considering future steps. This method allows the robot to maintain its balance during more dynamic movements. Foot placement control positions the robot foot in response to shifts in its center of mass (CoM). The bezier curve, a type of parametric curve often used in computer graphics, is also applied to generate smooth and continuous walking trajectories [47, 48]. The bezier curve allows for precise control over the shape and speed of the robot's movement. It is highly flexible, making them ideal for creating more complex or human-like walking patterns. The linear inverted pendulum model simulates human walking dynamics to generate walking trajectories. It assumes that the walking motion is similar to an inverted pendulum, with the leg swinging forward during a step, similar to a pendulum swing. Recent advances in machine learning have allowed reinforcement learning (RL) to generate walking trajectories [49, 50, 51]. In RL, an agent learns from interactions with

its environment to achieve a goal. The goal is successful and efficient locomotion, and the agent learns the optimal walking trajectory through a trial-and-error process.

Humanoid robots generally generate a trajectory that is created by various computational methods such as ZMP, CPG, COM, the bezier curve technique, dynamic modeling, and reinforcement learning algorithm. The Robinion series of robots are designed with a versatile mobility system in mind, allowing them to traverse diverse surfaces for the generation of their walking trajectory. It includes even, flat terrain and softer and more irregular surfaces like turf. The modeling and control systems embedded within the Robinion series are accordingly ensuring stable and efficient locomotion across some environments. In order to generate the walking trajectory for the Robinion series, twofold approaches have been developed. Firstly, we have designed a simplistic trajectory tailored for the lightweight mechatronic system, offering efficient yet uncomplicated capabilities. The lightweight trajectory method is designed to offer a balance between computational efficiency and effective movement. Secondly, we utilize the more complex Zero Moment Point (ZMP) trajectory generation method. This approach is more computationally intensive, but it offers greater precision and stability in the robot's movements, particularly in more challenging conditions or during more complex tasks.

During the process of walking, the humanoid robot experiences two crucial phases of support: the Double Support Phase (DSP), where both feet are in contact with the ground, and the Single Support Phase (SSP), where only one foot maintains ground contact. The trajectory generator for the robot leverages these phases. It combines them with the data from the x, y, and z positional coordinates and the yaw orientation of the feet. The combination enables the generator to formulate a distinct swing trajectory for each foot. For simplicity, the trajectory is designed to take the form of a quadratic function, as presented in Equation 2.4.

$$Foot_z = height - \frac{(\hat{x} - step\ size)^2}{step\ size^2/height} \tag{2.4}$$

In order to produce the humanoid robot in a cost-effective approach, we devise strategies to reduce the computational load and enhance the operational efficiency of the main controller. One such strategy involves real-time computation of the inverse kinematics solver and the walk-

ing trajectory directly on the Micro Control Unit (MCU). This approach eliminates the need for transmitting data between separate devices for computation, thus minimizing latency and saving critical processing time. By optimizing these key elements of the robot's operation, we aim to achieve significant savings in development costs while ensuring the robot's effective functioning. This approach also contributes to a more lightweight and compact design, enhancing the robot's versatility and deployment potential.

The other technique we employ for generating trajectory involves the utilization of the Zero Moment Point (ZMP). In the case of Robinion2, the robot operates using a responsive, omnidirectional walking gait which is dictated by three key inputs: v_x, v_y, and v_θ. These inputs represent the translation and rotation of the feet from the current pose. These parameters form the foundation for generating footstep patterns, instructing the robot where and how to step next. Furthermore, the same inputs play a pivotal role in generating the ZMP trajectory. The ZMP-based method provides higher stability and natural movements during walking, giving an advantage for navigating complex or uneven terrains. The footstep pattern is depicted through a two-dimensional pose derived from the computation as per Equation 2.5&2.6. In other words, the blueprint of a footstep is symbolized in a planar framework, which consists of positional coordinates and orientation. The planar representation is arrived at by utilizing Equation 2.5&2.6. This approach enables clear visualization of the foot's positional and orientational changes during the robot's walking gait.

$$\begin{bmatrix} p_x^n \\ p_y^n \\ p_\theta^n \end{bmatrix} = \begin{bmatrix} p_x^{n-1} \\ p_y^{n-1} \\ p_\theta^{n-1} \end{bmatrix} + \begin{bmatrix} R(p_x^{n-1} + v_\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ sf^n(2ho + 2ys + v_y) \\ p_\theta^{n-1} + v_\theta \end{bmatrix} \tag{2.5}$$

$$sf^n = \begin{cases} -1, & If\ left\ \ support\ foot \\ 1, & If\ right\ support\ foot \end{cases} \tag{2.6}$$

The variables p_x^n and p_y^n represent the coordinates of the next foothold position in the world coordinate. The variables pinpoint the exact location where the foot will land in the next step with respect to a universally acknowledged coordinate system. On the other hand, p_θ^n specifies

the orientation of the next foothold. The orientation refers to the angle or direction in which the foot will be positioned during the next step. The term ho is a constant parameter that designates the distance from the crotch to the hip in the robot structure. In essence, ho signifies the vertical length between these two points in the anatomy structure, and it remains unvarying irrespective of the movements or postures of the humanoid robot. On the other hand, ys serves as a variable parameter that embodies the offset of the sole in the default pose. That is, ys captures the relative shift or displacement of the sole from a pre-determined reference point when the robot is in its standard position. This parameter can vary, reflecting the dynamic nature of the foot position during various activities or postures.

Equation 2.7 represents the mathematical foundation for generating the Zero Moment Point (ZMP) trajectory. The calculation relies on crucial parameters such as the initial position of the Center of Mass (CoM) denoted as $p_{COM}^{(1)}$, and the final CoM position represented as $p_{COM}^{(n)}$. Furthermore, it also considers the position of the support foot p_{sf}, which remains a critical factor in maintaining the robot's balance during locomotion. As for the timing constants, t_1 is defined as half the Single Support Phase (SSP) duration, i.e., t_{ssp}/2, indicating the time period where the robot balances on one leg. On the other hand, t_2 equals the difference between the total step time (t_{step}) and half of the SSP, effectively capturing the timing of the shift from one support phase to another. These timing constants play a crucial role in coordinating the movement of the robot's limbs and maintaining stability throughout its gait cycle.

$$P_{ZMP}= \begin{cases} p_{CoM}^{(n)} + (\frac{p_{sf} - p_{CoM}^{(1)}}{t_1})t, & If\ t >= 0\ \ and\ t < t_1 \\ p_{sf}, & If\ t >= t_1\ \ and\ t < t_2 \\ p_{sf} + (\frac{p_{CoM}^{(m)} - p_{sf}}{t_1})(t - t_2), & If\ t >= t_2 \end{cases} \tag{2.7}$$

The diagram depicted in Figure 2.12 demonstrates the outcome of a footstep pattern and a ZMP trajectory, derived from the application of the values v_x=0.07m, v_y=0.00m, and v_θ=15deg. The inputs guide the robot to turn leftward, forming a curved path. It showcases the effectiveness of these variables in commanding the orientation and navigation of the robot.

From the generated ZMP trajectory, we acquire the Centre of Mass (CoM) trajectory, lever-

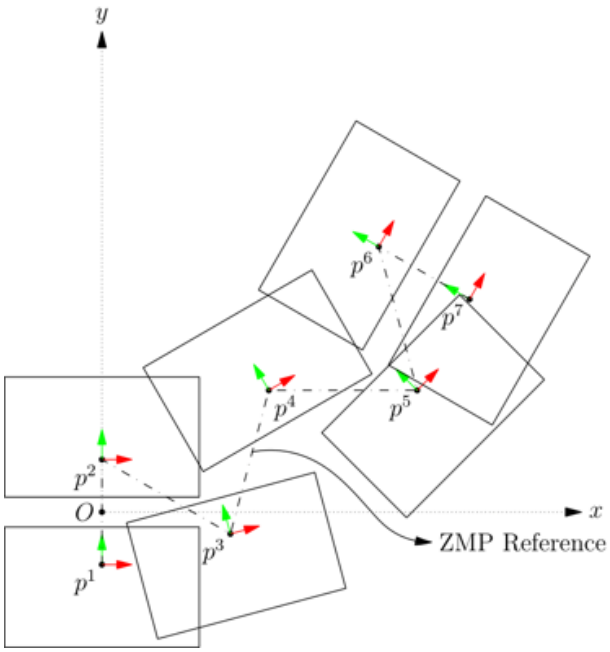


Figure 2.12: Illustration of footstep pattern

aging the analytical approach laid out by Harada et al.[52]. This technique provides an efficient and effective method for extracting the CoM trajectory, an essential component for the implementation of stable robot locomotion. The extracted CoM trajectory acts as a vital input for the process of the gait generation algorithm, steering the robot's motion and informing the foot positions during the walking phase. Nonetheless, to actualize a fully functional walking gait, the CoM trajectory must be integrated with the swing foot trajectory. This interplay between CoM and the swing foot trajectory is crucial, as it brings the theoretical walking model, allowing for the physical execution of the walking motion. It is not enough for the robot to balance its weight distribution effectively, which is the CoM trajectory. The coordinated movement of the swing foot is also essential in delivering a smooth, steady, and natural walking gait. For the Robinion walking gait, the trajectory of the swing foot is defined using the Bezier curve function. The method allows us to express the trajectory as a continuous and smooth curve, determined by several control points, denoted as a_i. The points essentially guide the form of the Bezier curve and, consequently, the movement path of the swing foot, as outlined in Equation 2.8. Utilizing this formulation, we are able to design precise and adjustable foot trajectories, contributing to the fluidity and adaptability of the robot's walking gait. The control points that shape the Bezier curve are defined as vectors, specifically a_0, a_1, a_2, and a_3. These vectors are parameterized as follows: a_0 is defined as $[p_x^{n-1}, p_y^{n-1}, 0]^T$, a_1 is defined as $[p_x^{n-1}, p_y^{n-1}, zf]^T$, a_2 is expressed

as $[p_x^{n+1}, p_y^{n+1}, zf]^T$, and a_3 is defined as $[p_x^{n+1}, p_y^{n+1}, 0]^T$. Each of these vectors contributes to outline the shape of the Bezier curve, ultimately guiding the movement path of the swing foot. The term referred to here symbolizes a scalar coefficient that plays a significant role in determining the peak height that the swinging foot can reach during the robot's gait cycle. This parameter plays a crucial part in defining the dynamics and the overall trajectory of the robot's movement, affecting factors such as step length and speed. It effectively allows for a fine-tuning of the gait characteristics to best suit the task requirements and environmental constraints.

$$b(p)=\sum_{i=0}^{n} \binom{n}{i} (1-p)^{n-1} p^i a_i, \qquad 0 < p < 1 \tag{2.8}$$

The CoM and sole trajectories, derived in the global coordinate, provide essential information about the robot's movement. This data is then converted into joint-space trajectories using a closed-form inverse kinematics solver. In other words, these position and orientation data points, represented in a world coordinate system, are translated to a corresponding set of joint angles that each joint in the robot's leg should follow. This process is crucial for controlling the robot's motion, as the joint angles directly determine the pose and movement of the robot. The final joint angles calculated from the inverse kinematics solver provide the specific positions each joint must move. The calculated angles are relayed to a sub-controller, which controls the operation of the actuators. The process ensures that the specific joint commands are carried out accurately, causing the robot to perform the desired movement or action.

2.7 Software design

Two software platforms have been developed to manage and control the Robinion series: one is based on the Robot Operating System (ROS), and the other utilizes a User Interface (UI)-based system. The ROS-based software is designed for flexibility and adaptability, using the robust architecture of ROS. ROS enables easy integration with a wide variety of hardware and software components, making it ideal for complex robotic applications. It allows for real-time data sharing and modular programming, making the management of the Robinion series more efficient. On the other hand, the UI-based software provides an intuitive and easy-to-use control interface for the Robinion series. The user-friendly platform allows for direct manipulation and

monitoring of the robot's functions and movements. It is designed with simplicity and convenience, catering to users who may not have advanced programming knowledge, thus making the operation of the Robinion series accessible to a broader audience. The comprehensive ROS and a user-friendly control interface ensure that the Robinion series can be effectively managed and controlled by diverse users, from researchers and engineers to everyday users.

2.7.1 UI-based software design

The UI-based software was first developed and implemented with Robinion version 1 (Robinion v1). Notably, Robinion v1 marked the first implementation of a parallel mechanism in the humanoid robot platform. This innovative approach allows for the efficient control and monitoring of the various operations. The UI-based software, designed with accessibility and user-friendliness, serves as an intuitive control system. It enables the user to input commands and view real-time feedback from the robot visually and interactively. It allowed users to easily manipulate the robot's movements and monitor its performance, thereby contributing to the optimization of the robot's operations. This sophisticated system has been designed with a broad range of applications. Its capabilities make it a versatile tool for a host of research domains. One of the essential research areas the system is leveraged for is omnidirectional walking. It involves researching and improving the ability of humanoid robots to move freely in all directions. Additionally, the system is instrumental in conducting research in object detection. The perception system involves the use of image processing techniques and advanced deep learning algorithms to enable the robot to recognize and interpret its surroundings accurately. Such capabilities are vital for tasks such as object tracking, obstacle avoidance, and path planning. While the system was originally designed for autonomous humanoid robot competitions, its versatile features and functionalities have enabled it to become an invaluable tool for a diverse range of research domains. An overview of the firmware and software architecture of the Robinion series for the UI-based application system is shown in Figure 2.13. It offers a comprehensive system structure, providing valuable insights into the operational mechanics of the humanoid robot. It aids in understanding how various software and firmware components collaborate to enable the functionality of the Robinion series.

To optimize computational load and keep the electrical system cost-effective, the design

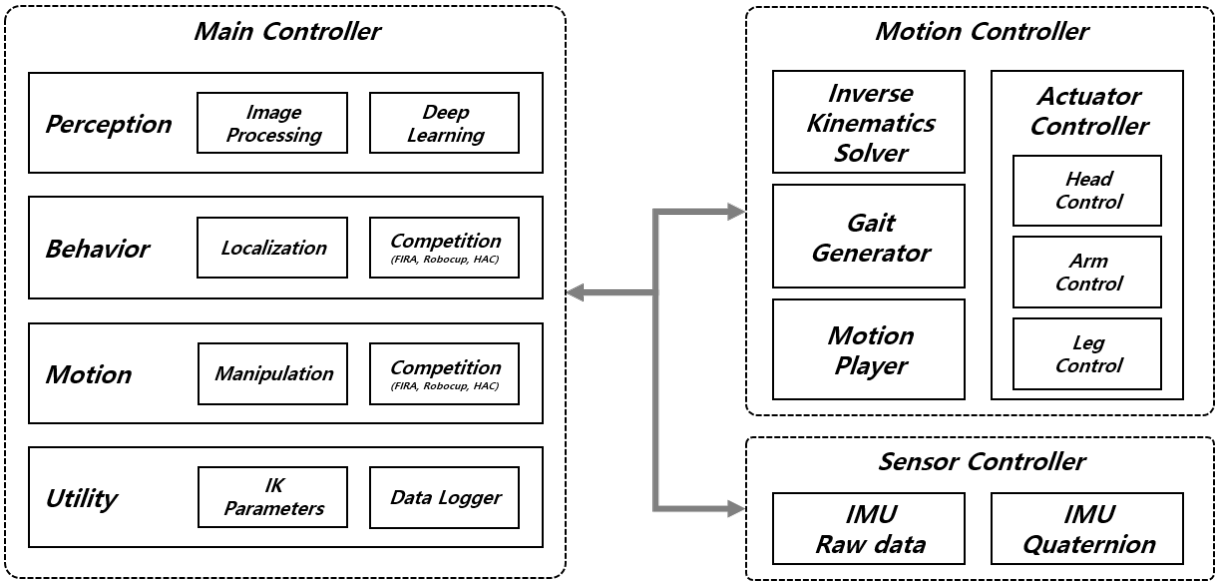


Figure 2.13: System architecture for Firmware and Software of Robinion series

of the humanoid robot contains three separate controllers: the main controller, the motion controller, and the sensor controller. Each controller is assigned specific roles to facilitate a divided approach to managing the functions. Having each controller focus on a distinct subset of tasks effectively divides the overall computational burden, ensuring that no single controller is overloaded with computational demands. This decentralized approach not only aids in managing computational complexity but also keeps the cost of the electrical system within reasonable limits. As each controller can be designed and optimized for its specific tasks, avoiding the need for a single, high-power unit to handle all tasks. The main controller is responsible for environmental recognition and object detection, leveraging image processing techniques alongside deep learning algorithms. Once the surroundings and any objects within them are identified, the robot's actions are dictated by a state machine to execute the assigned task effectively. For instance, in the FIRA Hurocup, Robocup, HAC competitions, and other research activities, the main controller communicates specific parameters pertaining to pre-programmed motions to the motion controller. The data transfer enables the execution of required actions in alignment with the given tasks. In addition, the main controller has a broader set of functions, which includes monitoring data loggers, transmitting inverse kinematics information to the motion controller, and supporting joystick-based control of the humanoid robot. The capabilities of the main controller not only allow for the precise execution of tasks but also provide a robust feedback mechanism. The feedback system ensures real-time updates on the robot's performance, which can

be pivotal for troubleshooting and improving future performance. The motion controller forms the core of mobility and locomotion, handling the crucial real-time computation for the inverse kinematics solution and gait trajectory generation. The motion controller includes a component known as the motion player. The subsystem executes pre-programmed movements, which the robot employs during competitions or when it performs manipulation tasks. These movements have been strategically coded and integrated into the firmware and software, enabling the humanoid to conduct actions for tasks with precision and consistency. The motion controller governs the position and speed of every actuator within the robot's structure, including those in the head, arms, and legs. This allows for coordinated and harmonized movements across the robot's body, leading to smooth, precise actions. By continuously monitoring and controlling the motions, the motion controller ensures that the humanoid is available to adapt to different tasks and environments, leading to more versatile performance. The sensor controller is an element of the robot system, incorporating an Inertial Measurement Unit (IMU) for feedback loop controls and direction recognition. In the dynamic environment where humanoid robots operate, recognizing instantaneous angular velocities and accelerations from IMU is crucial for maintaining stability and balance. A noteworthy feature of the sensor controller is using a Digital Motion Processor (DMP), which exploits quaternions to ascertain the Euler angles - roll, pitch, and yaw. Euler angles are integral to various complex tasks that the robots perform, such as Simultaneous Localization and Mapping (SLAM), localization, and navigation. By accurately determining these angles, the humanoid robot orients itself appropriately within its environment, making it more effective in navigating and interacting with its surroundings.

In Figure 2.14, a Python and Qt-based graphical user interface (GUI) is displayed, serving as an effective tool for controlling robot functionalities. The GUI software includes several categories related to different competitions, such as FIRA Hurocup, Humanoid Robot Application Challenge (HAC), and Robocup Humanoid League, as well as distinct research areas. The software categorizes further broken down into three dedicated pages: Game, Image, and Motion. The game page allows users to monitor the behavior of the humanoid robot, display the results of object detections for competitions and research, and examine the real-time robot orientation value calculated from the IMU sensor. It features functional buttons for experimenting with robot movements such as omnidirectional walking, ball kicking, and head moving, along with

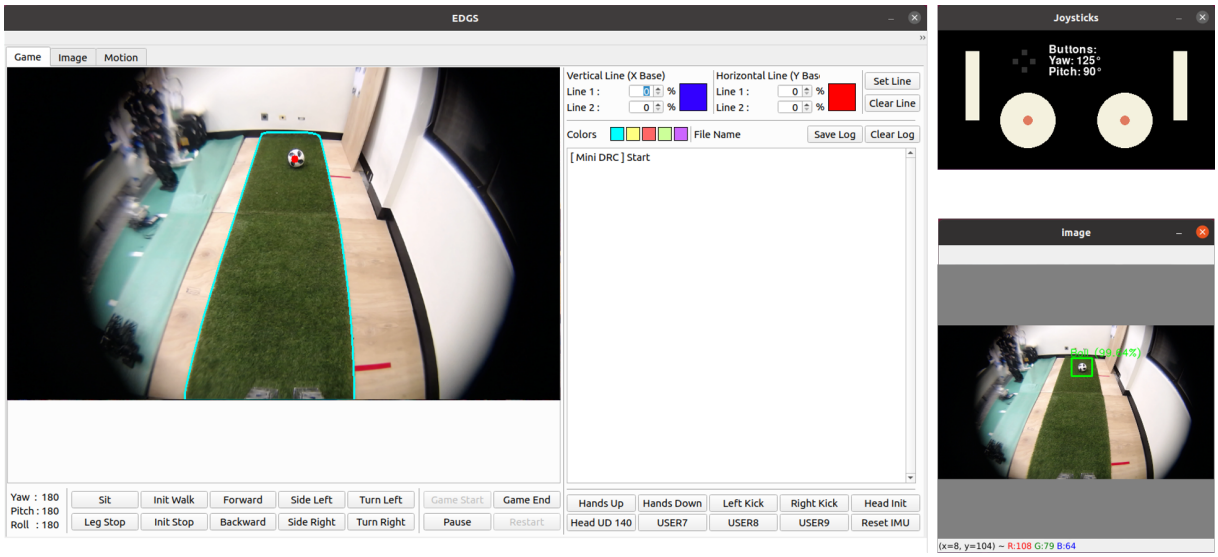


Figure 2.14: GUI-based software to control Robinion Series

options to set initial orientation values. The image page facilitates object recognition through color detection. Users are able to use mouse dragging to detect colored objects, leveraging the Lab color space's robustness against varying light conditions for effective object detection. Slide bars present on this page enable users to adjust the L, a, and b ranges for more precise detection [53]. On the motion page, users transmit real-time inverse kinematics parameters for walking to the robot, allowing them to fine-tune the parameters to find optimal values, as well as the capability to save and load these values for future use. Additional software features include object detection using advanced image processing algorithms and deep learning models, joystick-based robot control, interaction capabilities with other robots for multi-robot collaboration, voice recognition, and text-to-speech (TTS) functionality for uttering specific phrases. The software also establishes serial communication between each controller and reads video frames from the camera, providing comprehensive control and communication for the users.

2.7.2 ROS-based software design

The Robot Operating System (ROS) has been chosen as the middleware operating system and main software framework for the Robinion series. The particular version utilized is ROS Noetic, which holds the distinction of being the latest stable release of ROS with comprehensive support for Python 3. The motivation behind choosing ROS stems from its modular design, a feature that allows individual ROS packages to be efficiently reused, repurposed, and extended. The modularity streamlines the development process and benefits in maintaining a scalable and

organized codebase. Another benefit of ROS lies in its integrated suite of tools for logging and visualization. It significantly facilitates and accelerates the debugging process during development, ensuring rapid identification and rectification of errors or inefficiencies in the code. Python 3 was chosen as the default programming language for the software framework, a decision that reflects recent trends in the robotics and artificial intelligence (AI) fields. Although Python, an interpreted language, may not offer the raw execution speed of other languages, it comprehensively provides ease of use and extensive support from numerous scientific libraries. These attributes make it a well-suited language for accelerating research and development in complex robotics applications. Moreover, other programming languages with ROS client libraries, such as C++, can be used if developers prefer or need them for specific tasks. This flexibility makes ROS an even more attractive choice, allowing multiple languages within a single project to suit the needs of different system parts.

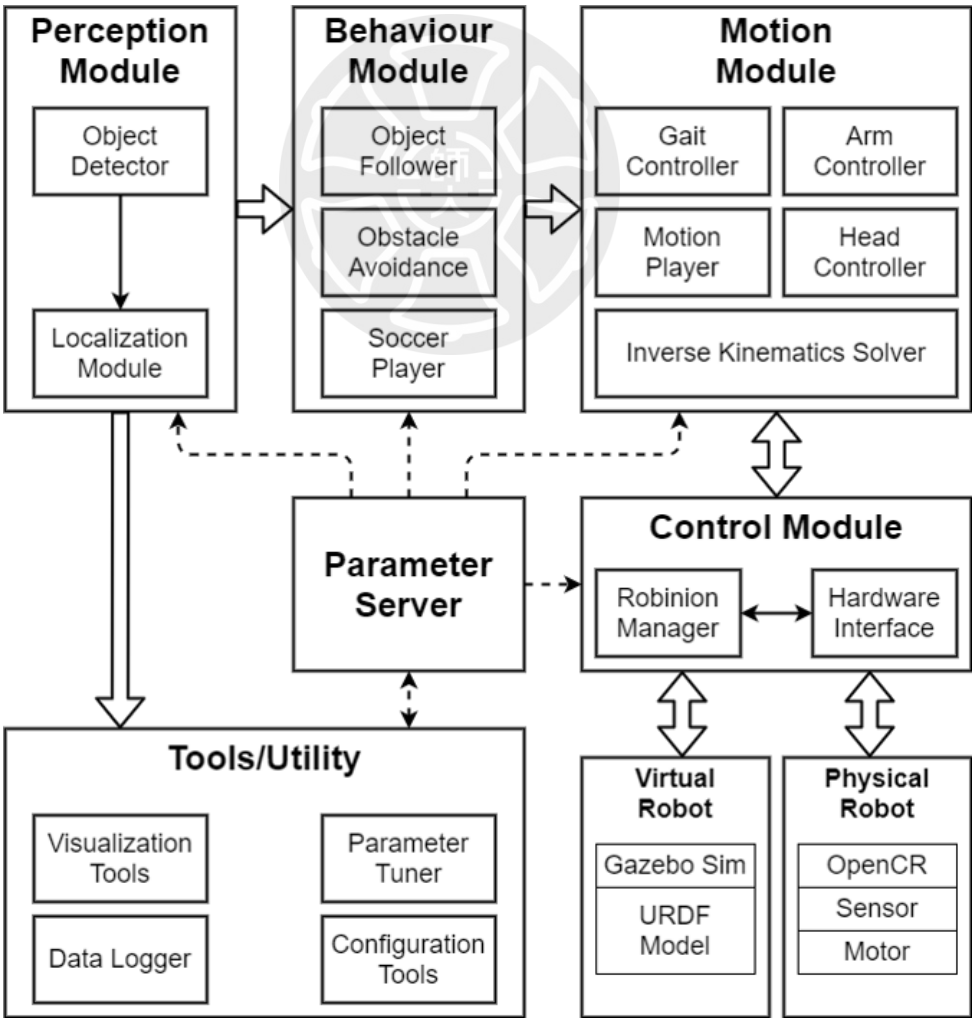


Figure 2.15: Robinion2 software architecture

The software architecture of Robinion2 is depicted graphically in Figure 2.15. The architecture is comprised of four modules: Perception, Behavior, Motion, and Control. Each of the modules, in turn, is made up of multiple ROS nodes. These nodes serve as the fundamental building blocks of the ROS computational graph, carrying out a variety of tasks within their respective modules. A significant component of the designed architecture is the parameter server, a central hub for storing all the robot parameters. The server is universally accessible to all the modules within the framework, thus ensuring uniformity and consistency in the operational parameters of the Robinion series across all tasks. In addition to facilitating parameter modifications, the server also facilitates centralized parameter management. The use of a centralized server in this manner ensures that any changes in the parameters are instantaneously reflected across all functions of the humanoid robots. For data visualization, the architecture employs the ROS RViz tool. The Rviz provides real-time visualizations of the data gathered by the Perception module, aiding in understanding and debugging the robot's perception of its environment. Furthermore, the architecture includes the rosbag package for data logging, which is a vital tool within the ROS, allowing for the capture and storage of important data generated during the robot operation. After the operation, the recorded data can be played back and analyzed, which facilitates detailed troubleshooting and post-operation analysis. The Robinion software framework is designed for use with both simulated and real-world robots. In terms of simulation, the Gazebo simulator is employed to create a virtual representation of the robot and its surrounding environment. The structure of the virtual robot is determined by the Unified Robot Description Format (URDF), a widely utilized scripting language for robot modeling. While Gazebo is the default simulation environment for the Robinion series, the use of URDF means the virtual robot is not restricted to this particular simulator. It is possible to be integrated with any other simulator that supports URDF, granting developers flexibility in choosing their simulation environment. When it comes to operating on the actual robot, the software framework interfaces with the hardware via OpenCR, an embedded system designed specifically for ROS. The embedded system serves as a bridge between the software and the physical components of the robot, facilitating their communication and operation. The versatility of the Robinion framework, in terms of its ability to function on both virtual and real robots, provides a significant advantage for developers. The software system is available to create, test, and refine their algorithms in the

simulated environment and then directly apply these same algorithms to the actual robot without needing to make any modifications. This convenience streamlines the development process and enhances the efficacy of algorithm testing and deployment. The perception module of the software framework comprises two integral ROS nodes: one for object detection and another for determining the robot's position, known as self-localization. The object detection node utilizes a finely tuned deep learning model that can operate on a compact deep learning accelerator, specifically, the Edge TPU from Google. The Edge TPU offers a cost-effective solution by significantly enhancing the speed of the inference process and overall system response time, all without the necessity of costly hardware. Detailed information regarding this object detection algorithm is further elaborated in Section 2.8. In addition to object detection, the perception module also houses a node for self-localization. The localization node is responsible for determining the robot's current position within a specific environment, such as a soccer field and race track. This crucial information allows the robot to navigate its surroundings accurately, playing a pivotal role in task execution, path planning, and obstacle avoidance. Self-localization is thus a component in ensuring effective movement within its intended environment. The motion module comprises several nodes that collectively manage the robot's movements. The motion module is split into four distinct nodes for refined control over each aspect of the robot body: the gait controller, arm controller, motion player, and head controller, along with an inverse kinematics solver node. In the domain of humanoid robotics, various body parts are moved to achieve specific actions. For this reason, the motion module designates a controller for each part of the body. For instance, the gait controller governs the humanoid robot's leg movements to enable walking. Certain humanoid robot applications necessitate arm movements for tasks like object grasping and picking. For the Robinion series, the manipulation tasks are accomplished by the arm controller node, which executes the provided trajectory to perform the task. In addition, humanoid robots are the ability to perform static motions by using pre-recorded joint states. This functionality is handled by the motion player node in the motion module. The motion player node records predefined joint states and timings, which are then executed through ROS actions to perform specific actions. This capability is beneficial for generating specialized motions such as ball kicking, handshaking for greetings, and more. The last node in the motion module, the head controller, manages the robot's gaze direction. The head controller receives

target joint angles for pan-tilt joints and movement timings, then calculates the joint value using the minimum jerk trajectory method. Both the walking gait and manipulation require an inverse kinematics solver to convert the pose into joint angles. The inverse kinematics node manages this task using two methods. For leg movements, a closed-form method is employed, and for arm movements, an iterative method is used. The leg joints require a closed-form solution as the gait controller utilizes them and must function in real-time. Conversely, for manipulation tasks, the computations aren't time-critical, making the iterative method a suitable choice.

2.8 Perception

For a robot to successfully function in human-engineered environments, it is critical that it demonstrates a broad range of capabilities and maintains high-performance levels. One fundamental aspect of this is the robot's ability to recognize and interact with various elements in its surroundings. Contemporary perception systems frequently utilize Deep Learning methodologies for tasks such as object detection, image classification, pose estimation, and segmentation across diverse research fields [54]. Among these methods, convolution is a core operation, forming the backbone of numerous prevalent architectures. However, these systems often tend to be relatively intricate, with a significant number of parameters, thus demanding extensive computational resources for training and inference. While GPUs are a common choice for handling such computations, recent trends lean towards innovative technologies like the tensor processing unit (TPU) supported by Google [25, 55, 56, 57]. Designed to streamline vector and matrix operations, TPUs are strategically pre-configured to optimize each ALU location, thereby providing enhanced computing power in specific environments compared to GPUs. Nevertheless, in robotic applications, both TPUs and GPUs have limitations, primarily due to their substantial weight and the requirement of expensive hardware infrastructure. The perception for the Robinion series employs the Edge TPU-based Coral USB accelerator to navigate this challenge in our perception system. This ASIC USB hardware is precisely engineered to execute neural network model inference on low-power devices. The accelerator delivers an impressive performance of four trillion operations per second (TOPS) while consuming a meager two watts. It is achieved through the utilization of 8-bit integer operations, contributing to the device's extraordinary inference speeds and low power usage. Despite its capabilities, models to be deployed

on this device are subject to certain restrictions. They include 8-bit fixed-point tensor parameters, fixed tensor sizes and model parameters at compile-time, and the acceptance of 1-, 2-, or 3-dimensional tensors. Furthermore, the models must solely use the operations supported by the accelerator. In the perception system of the Robinion series, two methodologies are utilized for object detection tasks: YOLO-based techniques and heatmap-based approaches.

For enhancing the ball detection technique, previously based on the Hough circle algorithm, Robinion is equipped with the Coral USB accelerator [58]. The Edge TPU allows for the implementation of a ball detection algorithm with the Tiny YOLO-V3 model ([59]), thus contributing to more effective and accurate recognition performance. The training process for the model adheres to the standard methodology, which means it involves the use of 32-bit floating-point numbers. Once the training phase is fully completed, the model undergoes a conversion process to make it compatible with the Google Tensorflow library. The conversion process includes a significant step known as post-training quantization [60]. During this operation, the trained model is optimized for performance and size, making it more efficient for deployment on hardware devices. The transition process encompasses a significant step referred to as post-training quantization. This technique entails feeding examples from the dataset into the already trained model to facilitate the computation of transformations from 32-bit floating-point numbers to more manageable and compact 8-bit integers. The optimization that quantization offers is critical in reducing the model's resource demands, enhancing its speed, and decreasing its footprint. This makes it much more suited for deployment on hardware devices, such as Google's Coral USB accelerator, without causing substantial compromises on the model's predictive capabilities or overall accuracy. Upon completion of the 8-bit quantization process, the transformed model can be prepared for deployment. It is achieved by compiling the model using a specialized tool that the manufacturer of the device supports. This compiler tool is custom-made by the device manufacturer, which in this case is Google, for devices such as the Coral USB accelerator [26]. It helps translate the model into a format that the targeted device can understand and execute efficiently, ensuring smooth deployment and operation of the model on the device. The compiler tool performs an important verification step by confirming that all the operations utilized by the model are compatible with the Coral USB accelerator. It guarantees that the model's computations can be correctly interpreted and executed by the hardware. Following this validation, the

compiler transforms the model into a file format that is specifically tailored for the Coral USB accelerator. This allows the device to efficiently load and run the model, which optimizes the performance of the machine learning operations on this specific hardware platform. The design of the proposed model follows the architectural blueprint of YOLO-V3, maintaining most of its original components. The only deviation is the replacement of the leaky ReLU activation functions with standard ReLU (Rectified Linear Unit) activation functions. This alteration was made due to the Coral USB accelerator's compatibility constraints - leaky ReLU operations are not supported on this particular hardware. Consequently, standard ReLU activation functions, which are well-supported, were implemented. This change ensures efficient processing on the Coral USB accelerator without significantly impacting the model's performance. Figure 2.16 presents a schematic diagram that apprehends the sequence of actions involved in the conversion process. The depiction provides assistance in understanding the integral transformations necessary for the effective utilization of our deep learning model on the chosen hardware platform.

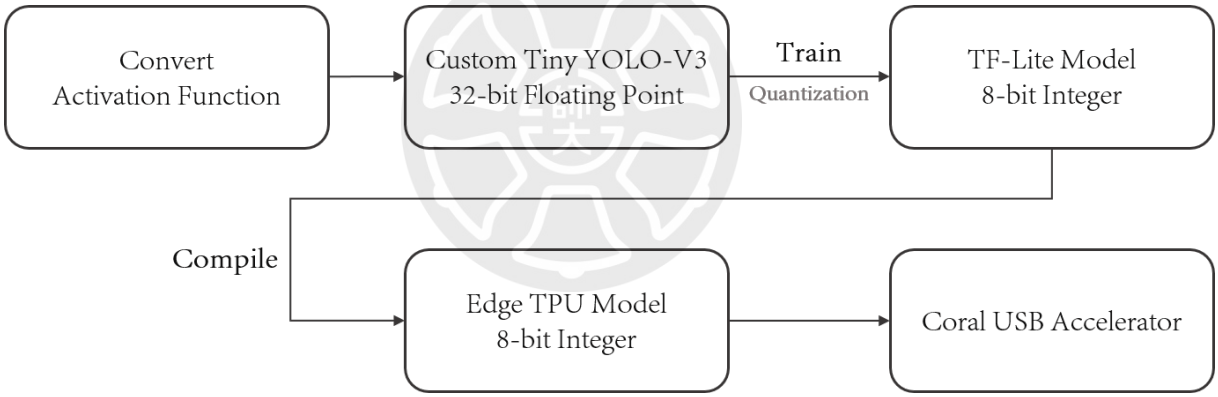


Figure 2.16: Flowchart illustrating the steps involved in converting a trained Tiny YOLO-V3 model into a deployable format compatible with the Edge-TPU

In autonomous humanoid robot competitions such as the RoboCup Humanoid League and the FIRA Hurocup penalty kick event, the ability to recognize a soccer ball becomes of paramount importance to performing autonomous soccer. The task of ball recognition serves as the subsequent actions, which are the localization, movement, and kicking ball, which are critical for the soccer-playing humanoid robot. The better a robot is at spotting the ball, the more effectively it can strategize and execute game plays. Therefore, refining and perfecting the soccer ball detection capability is an essential preference in the development of an autonomous soccer-playing robot for these events. To operate our customized tiny YOLO-V3 model to en-

able autonomous robotic soccer, a uniquely curated dataset of soccer ball images is used. This dataset is a critical role in training and refining our model to detect and interact with soccer balls in competitions. The dataset, employed for substantiating the efficacy of the approach, is composed of various pictures featuring a soccer field and a ball. The images provide a realistic representation of the environment in which the humanoid robot will operate, therefore, fostering a comprehensive validation of our methodology under actual conditions. The soccer field presented in the dataset is the similar turf used in the RoboCup Humanoid League, explicitly maintaining a similar grass height, albeit scaled down to a smaller size. The turf is a more realistic training and testing environment, mirroring the conditions the robots would experience during competitions. Our dataset comprises 800 comprehensive images. Out of the dataset, a subset of 120 images representing 15% of the total is randomly chosen as a holdout validation set. This set is crucial in evaluating the model performance. The remaining dataset, amounting to 680 images or 85%, is dedicated to training purposes, providing the model with ample instances. To thoroughly evaluate the performance of our model, a balanced validation set is devised. The dataset consists of an equal number of images containing soccer balls and images without soccer balls. This way, we can comprehensively measure the model's effectiveness through several metrics. The dataset validates true positives (TP), where the model correctly identifies an image containing a soccer ball, and true negatives (TN), where it correctly recognizes the absence of a ball. False positives (FP) are instances where the model mistakenly detects a ball, and false negatives (FN) occur when the model fails to identify a soccer ball in the image. The measures allow us to calculate precision (how many of the predicted positives are truly positive), recall (the proportion of actual positives that were correctly classified), and overall accuracy of the model's predictions. In section 2.9, we contrast the performance of two distinct methods: our optimized Tiny YOLO-V3 model, which runs on the Coral USB accelerator, and a traditional image processing technique reliant on the Hough Circle Transform. The comparison evaluates the efficacy of both approaches in terms of precision, recall, accuracy, and speed under various conditions. It provides valuable insight into the advantages of using a deep learning model over a conventional image processing algorithm regarding its adaptability and efficiency, particularly in real-time robotic applications.

The other technique we use for object detection is a deep learning approach based on

heatmap [61, 62]. The heatmap-based approach is a prominent method in deep learning, often used for detecting and localizing objects in images. A heatmap model was developed and implemented for the Robinion series, focusing on fast inference speeds. In object detection, a heatmap essentially refers to a grid overlay on the image where each cell is assigned a probability that it contains the object of interest. The proposed method is particularly effective for detecting multiple objects and their respective locations within the same image. For autonomous soccer, humanoid robots have to be equipped with the ability to identify essential elements within the soccer field accurately. The elements include the soccer ball, goal posts, field boundaries, penalty marks, other participating robots, as well as the intersections of the field lines (see Figure 2.17).

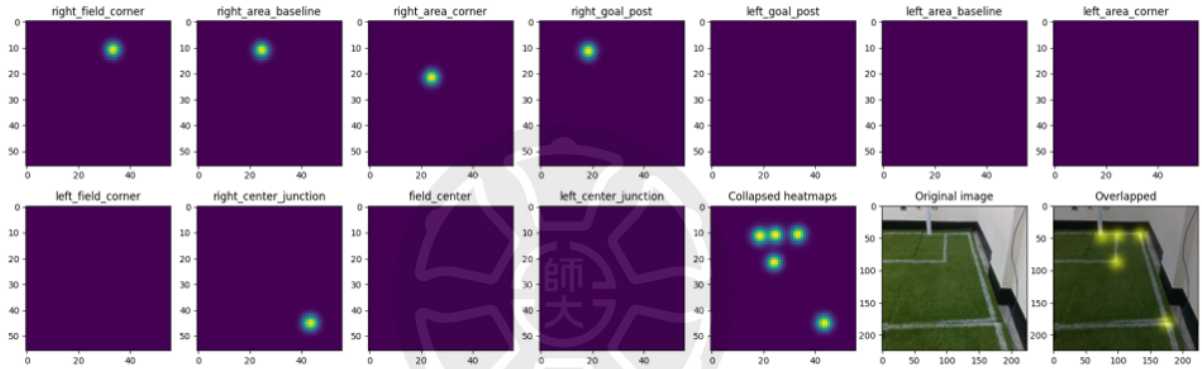


Figure 2.17: Heatmap-based object detection for autonomous soccer play

Each object plays a significant role in determining the robot's strategy during the game. The ability to accurately recognize the elements impacts not only the performance of the humanoid robot but also its ability to react appropriately to changing dynamics of the game, enhancing its overall gameplay effectiveness. Conventional strategies for object recognition might involve gathering a dataset and training either a bounding box detector model or a segmentation model [63, 64]. However, it is a trend among Robocup teams leveraging models we refer to as heatmap models. As its name suggests, the heatmap models generate a two-dimensional tensor (image), where the objects of interest are represented as distinct blobs. The approach marks a deviation from the traditional methods and is a more intuitive way of visualizing and understanding the presence and position of objects. Each class can be encoded uniquely in a separate channel or using distinct color coding in a three-channel image. An advantage of this method is that there is no need for any post-processing of the network outputs, a common requirement in bounding box

detector models. Furthermore, the heatmap models can be trained supervised manner, treating it as a regression problem. The approach is simplified further by using a straightforward loss function, such as the mean squared error, thereby minimizing the computational complexity while maintaining efficient learning. Under these constraints, we developed an architecture that is based on a Mobilenet-V2 backbone pre-trained on the Imagenet dataset with the classification layers removed [65]. We chose this backbone because of its support by the Edge TPU device. On top of the backbone, we built our regression head composed of a sequence convolution, upsampling (bilinear interpolation), and batch normalization blocks of layers [66]. The proposed architecture is shown in Figure 2.18 and has a total of 2.3 million trainable parameters.

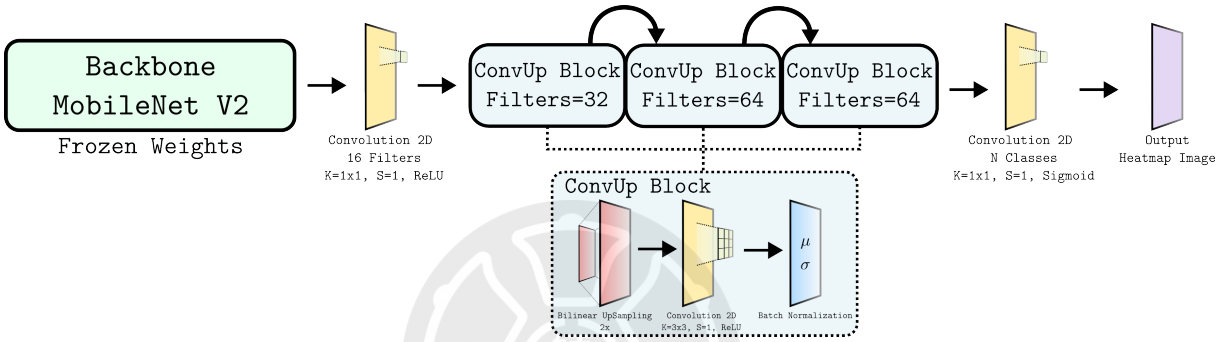


Figure 2.18: Architecture of the perception system with MobileNet-based deep learning

Each class of interest is output in a separate channel in a similar fashion. In contrast, we adopted a more compact input and output (heatmap) resolution, roughly six times smaller than the other approaches. Our input resolution is a 224x224 image, and the output heatmap is just 56x56 pixels. These compact representations enable the model to fit the Edge TPU device and achieve fast inference speeds. Our proposed model was implemented using Tensorflow 2.0, so we could easily convert the trained checkpoints to a Tensorflow Lite model. After the training and the quantization process are finished, the resulting model is compiled into a special format that can be loaded by the Edge TPU device.

2.9 Evaluation

The comprehensive details of the Robinion series, designed for efficient real-world operation, have been described. The details encompass a cost-effective yet robust mechatronic system, well-architected software platforms, simple and ZMP-based walking gait patterns, and a lightweight and sophisticated perception system that utilizes deep learning techniques. The sub-

sequent section explains into the findings drawn from a systematic evaluation of experiments conducted on the mechatronic system of Robinion Sr. and Robinion2. The experiments are meticulously designed to evaluate the performance of crucial aspects of the proposed robotics solution, such as the effectiveness of the walking gait algorithm and the accuracy of the perception system designated by deep learning methodologies. The outcomes of the experiments offer insights into functionality and efficiency in a real-world setting. Firstly, section 2.9.1 presents a result of the walking gait algorithm, focusing specifically on its implementation of an analytical approach to inverse kinematics solver along with its walking trajectory generator. The rigorous analysis allows us to examine how the inverse kinematics solver and gait generator contribute to the robot's mobility and stability. The following section examines the perception system designed especially for autonomous soccer and how the method leverages our customed deep learning techniques to detect and interact with the soccer ball efficiently, effectively turning the raw sensory data into actionable information for the robot.

2.9.1 Walking Gait

The implementation of the walking involves the calculation of the proposed gait trajectory generator with an analytical inverse kinematics solver. It represents a significant computational aspect of the walking mechanism, wherein the gait trajectory generator models the desired mo-

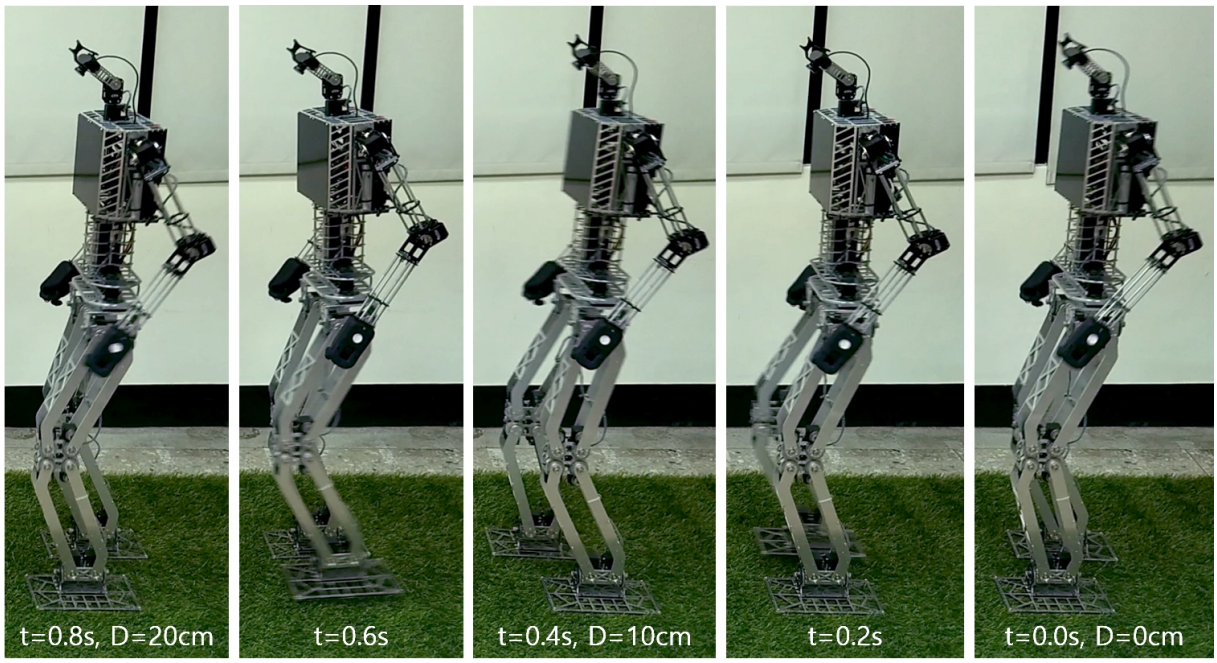


Figure 2.19: Stable walking gait of *Robinion Sr.* (Speed: 25cm/s)

tion path while the inverse kinematics solver determines the necessary joint angles to achieve the path. An evaluation of the solution is based on the stability with which it walks. In other words, the evaluation criterion is how consistently and efficiently the robot can maintain balance and avoid faltering or toppling during walking. Factors such as robot speed, which is related to step size, are also taken into account in the evaluation of walking stability. The trajectory generator is leveraged to actualize the omnidirectional walking gait, enabling the robot to move fluidly in any direction. It not only covers forward and backward walking but also includes lateral and rotational walking. Given the emphasis of the perception system on autonomous soccer play for humanoid robots, it is essential to conduct walking tests in a setting that closely mimics an actual soccer field. Therefore, a series of experimental trials are carried out on artificial turf, a commonly used surface in RoboCup humanoid league. Artificial turf poses considerable challenges for the walking gait of humanoid robots. Its uneven surface introduces complexities beyond those found on flat floors, making ambulation notably more difficult. The sporadic highs and lows, or turf irregularities, increase the probability of the feet catching or tripping, potentially destabilizing the robot. Such terrain intricacies necessitate more advanced gait strategies and balance control mechanisms to ensure smooth and uninterrupted motion. To validate the stability and velocity of the walking gait, Robinion Sr. and Robinion2P executed an experiment where the robots were tasked with traversing a length of $3m$ on the artificial turf surface. The purpose of this trial was twofold: firstly, to assess the ability of the robot to maintain a balanced posture and steady walking on turf, and secondly, to evaluate the speed at which the robot could cover the given distance. The experiment was set up to carry out a walking test with a fixed step cycle of $400ms/step$ for Robinion Sr. and $350ms/step$ for Robinion2P while adjusting the stride length within a range from as short as $2cm$ to up to $15cm$. The Robinion Sr. and Robinion2P models demonstrated stable walking over a distance of $3m$, maintaining balance up to maximum step sizes of $10cm$ and $8cm$, respectively. However, their stability decreased when the stride length exceeded these thresholds, and they performed an unstable gait. The result implies that although they can achieve considerable step lengths, an upper limit exists with open-loop walking gait for step size, beyond which closed-loop walking gait is required. The backward walking gait of both humanoid robots demonstrated more excellent stability at a speed of $30cm/s$ compared to its forward counterpart. Furthermore, the omnidirectional walking gait integrates forward, back-

ward, lateral, and rotational walking and serves stability contingent upon the specific values of x, y, z, and θ_{hy}. It indicates that the stability of omnidirectional walking is related to fine-tuning the parameters. Figure 2.19 illustrates the stable walking gait of Robinion Sr., which maintains its stability at a maximum speed of $25cm/s$. It demonstrates that the humanoid robot conducts the walking gait with a period of $0.4sec/step$ and a step size of $10cm$.

Figure 2.20 demonstrates a depiction of the stable walking gait of Robinion2P, featuring a maximum speed of 23 cm/s, with a cycle of $0.35sec/step$, and a stride size of $8cm$.

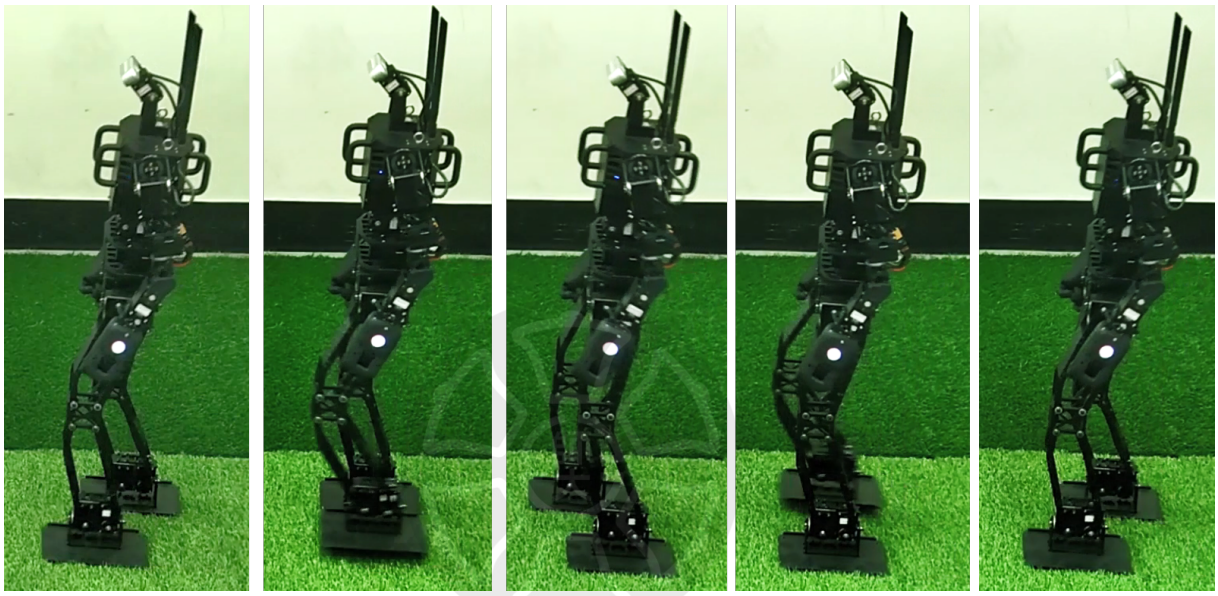


Figure 2.20: Stable walking gait of *Robinion2P* (Speed: 23cm/s)

Robinion Sr. and Robinion2P show impressive and stable walking gaits, well-suited for tasks in various real-world environments. The performance results demonstrate the effectiveness of the robots' design and the implemented trajectory generator, representing their potential for autonomous soccer and other humanoid robot applications.

2.9.2 Perception

This section evaluates the proposed deep learning-based perception algorithms for autonomous soccer: tiny YOLO-v3-based and MobileNet-based architectures. The perception system is designed to operate on the Edge TPU, a high-speed machine-learning accelerator. The evaluation aims to discern how the systems perform in real-time object detection and recognition tasks, effectively boosting the autonomous functionalities of robotic systems. The perception system is interested in its performance in applications related to autonomous soccer playing for

humanoid robots. Firstly, the results are compared with the proposed perception approach based on Edge TPU and the previous ball detection method, the Hough Circle transform. The comparison provides us with a measure of performance, understanding how much more effective the new method is in terms of ball detection in a robot soccer game. For the purpose of the evaluations, the images that comprise the dataset are collected while the robot is in a static or non-moving state. This setup allows the model, trained on the dataset, to be used in a humanoid robot soccer game. It accomplishes that the Robinion series play soccer autonomously, recognizing the environment while standing, walking, or kicking the ball. A performance comparison of the two methodologies under evaluation, conducted on the same validation dataset, is presented in Table 2.8. The comparative analysis shows a clearer understanding of the effectiveness and accuracy.

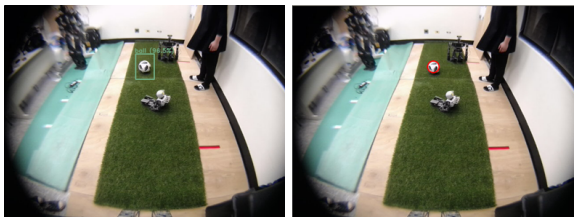
Table 2.8: Comparison of the detection results between the proposed method and the Hough Circle method for detecting the soccer ball

	TP	TN	FP	FN	Precision	Recall	Accuracy	Inference
Tiny YOLO-V3	100%	92.04%	7.96%	0%	92.63%	100%	96.02%	38ms
Hough Circle	52.6%	71.64%	28.36%	47.4%	64.97%	52.6%	62.12%	69ms

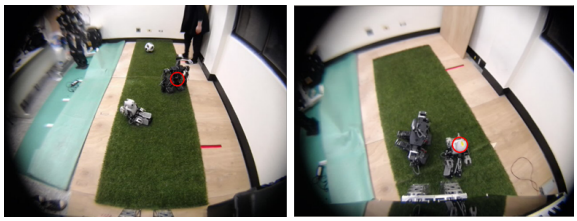
In order to assess the effectiveness of the tiny YOLO-v3-based object detection model, several commonly used metric evaluations were considered. The indicators include precision, which measures the ratio of correctly predicted positive observations to the total predicted positives; recall, which is the ratio of correctly predicted positive observations to all observations in actual class; accuracy, which shows the ratio of correctly predicted observations to the total observations; and inference speed, indicating the time taken by the model to predict the output after receiving input. Analyzing and comparing these key performance indicators makes it possible to understand and further enhance the model's performance and efficiency in object detection tasks. To analyze critical performance indicators such as precision, recall, and accuracy, it examines the fundamental parameters of a classification task: true positive, true negative, false positive, and false negative. The terms are used per their standard definitions in Equation 2.9 to compute precision, recall, and accuracy. The values are fundamental to understanding how well the model performs on different classification tasks.

$$
\begin{aligned}
Precision &= \frac{TP}{TP + FP} \\
Recall &= \frac{TP}{TP + FN} \\
Accuracy &= \frac{TP + TN}{TP + FN + FP + TN}
\end{aligned}
\tag{2.9}
$$

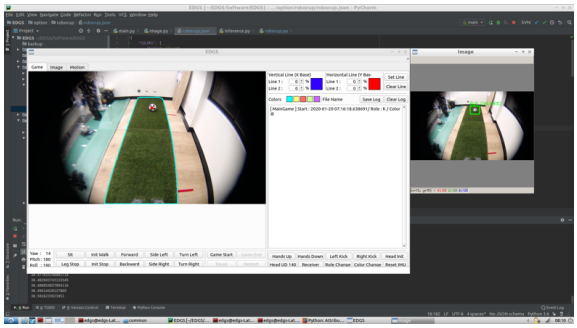
As anticipated, the deep learning-based system outperforms the Hough Circles approach regarding evaluation: precision, recall, and accuracy. The result is not unexpected, given the advanced capabilities of deep learning methodologies. The evaluation metrics derived from the validation set illustrate that the custom YOLO model successfully identified all instances of the soccer ball, labeling them as true positives. This demonstrates a 100% recall rate, which strongly indicates the model's effectiveness at detecting the presence of the ball in all circumstances within the test data. On the other hand, the Hough Circles method identified only about half of the soccer balls in the images. This points to a significant limitation of the Hough Circles technique, as it misses several instances where the ball is present, reducing its effectiveness as a ball detection system. Moreover, while it is true that the deep learning model isn't entirely immune to generating false positives, the rate at which it does so is significantly lower, approximately 8%. This is in stark contrast to the image processing method, which produces false positives at a much higher rate of approximately 28%. This means that, in the deep learning model, only about 8% of the time does it mistakenly identify something as a soccer ball when it is not, a far cry from the nearly one-third rate of the image processing method. The lightweight deep learning method we have proposed delivers significantly improved precision and recall metrics results. The proposed deep learning approach correctly identifies a higher proportion of true instances (high recall) and makes fewer mistakes in falsely identifying other instances as the target class (high precision). This indicates a significant improvement over traditional methods, demonstrating the value of our lightweight deep learning approach in enhancing object detection performance. Figure 2.21 presents a comparative analysis between the deep learning-based soccer ball detection system and the Hough Circle based on traditional image processing techniques. By providing a side-by-side comparison, this figure clearly represents the effectiveness and superiority of the deep learning-based approach over the traditional image processing method in soccer ball detection.



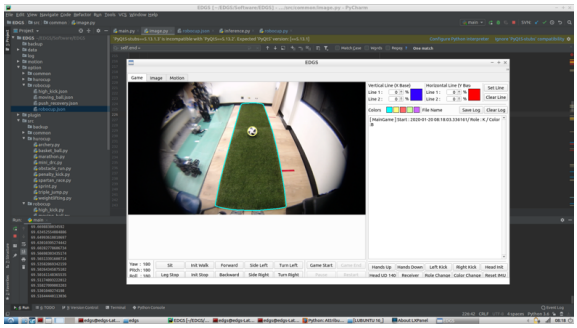
(a) True Positive



(b) False Positive



(c) Tiny YOLO-V3 Inference



(d) Hough Circle Inference

Figure 2.21: Comparison of ball detection results between Tiny YOLO-V3 and the Hough Circle approach. The top row displays examples of true positives from both models, as well as false positives from the Hough Circle method

One of the primary benefits of the proposed YOLO model is its speed—it operates roughly twice as fast as the Hough Circle method, possessing a steady inference time of 38 milliseconds. As a result, It processes approximately 26 frames per second, which is ample for real-time applications such as autonomous soccer and manipulation tasks. The use of the Coral USB accelerator device for inference execution ensures minimal strain on the CPU (or GPU). It frees up the CPU to manage other tasks, optimizing the system's performance. Taken as a whole, the ability of the deep learning-based perception system to detect the soccer ball for autonomous gameplay far exceeds that of its predecessor. It proves higher accuracy and a significantly reduced rate of false positives. The proposed perception method contributes to improved performance and reliability and a more efficient and autonomous soccer gameplay experience.

To validate the reliability and adaptability of the perception system, deep-learning-based ball detection experiments were conducted under various conditions, including areas outside the soccer field and when the robot is walking. To achieve a high-accuracy model, the training dataset is diversified to include the soccer field and areas outside, including images with the ball present. It equips the model with the capability to detect the ball with different backgrounds and settings. One such experiment illustrated the recognition of the ball in cluttered environments

with multiple objects, as depicted in Figure 2.22. The results demonstrate the system's stability in recognizing the ball even in challenging circumstances - for instance, near a chair whose color closely matches the ball, on a shelf amidst an array of different objects, or even within blurred images. Such outcomes highlight the robustness of the proposed perception method, underscoring its ability to reliably identify the soccer ball irrespective of environmental variations and complexities.

Figure 2.22: The perception system results demonstrate robustness in challenging scenarios, including scenes with similar colored objects, a cluttered shelf, and blurry images

For autonomous soccer play, it is necessary for the robot not only to detect objects while standing accurately but also during walking and across various regions, as shown in Figure 2.23. A soccer ball-kicking algorithm was implemented to accomplish the experiment. It includes an omnidirectional walking gait that allows the robot to approach the ball from a distance of 2 meters. The walking speed is set to its maximum step size of $7cm$, with a step cycle time of $400ms$. In order to determine the optimal position for the kick, the robot adjusts its step size and the θ value of the hip yaw joint while approaching the ball. Omnidirectional walking contributes

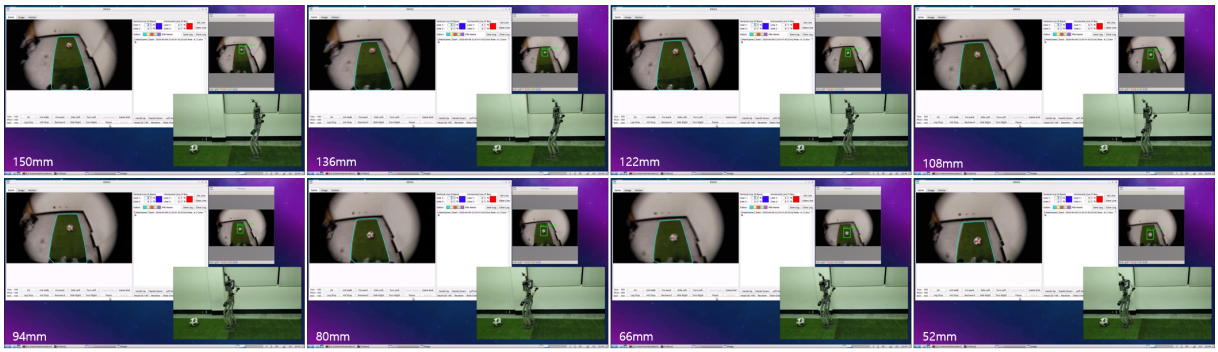


Figure 2.23: Demonstration of the perception system robustness during autonomous soccer play

to precise positioning and the successful kicking of the ball. To quantify the robustness of this process, the recorded video of the experiment was split into frames at a rate of 30 frames per second, facilitating the measurement of the recognition rate. The results demonstrated a perfect recognition rate of 100%, indicating that the robot could consistently identify the ball throughout the experiment. It shows the efficacy of the proposed perception system, particularly its stability and adaptability under the dynamic conditions of autonomous soccer play.

In addition to the YOLO-based object detection method, we also employ a sophisticated deep-learning approach that makes use of heatmaps. This heatmap-based method is a significant advancement in the field of object detection, as it provides a visual representation of data where different color shades represent varying values. It essentially produces a two-dimensional graphical illustration of the probability distribution for the presence of an object in an image. This heatmap-enhanced technique goes beyond the basic capabilities of YOLO-based object detection. It doesn't just identify the existence of objects, but it also delivers a comprehensive understanding of their spatial location within an image, thus enriching the overall object detection process with improved precision and robustness.

Table 2.9: Comparison of different heatmap models on the soccer field dataset. Precision and recall are averages computed over the validation set

Model	Precision	Recall	Inference	Parameters
Sweaty-V1 - GPU	99.04 %	99.44 %	5.32 ms	679 k
Sweaty-V2 - GPU	99.31 %	98.37 %	4.24 ms	483 k
Sweaty-V3 - GPU	99.49 %	98.93 %	4.91 ms	331 k
Nimbro - GPU	89.19%	81.96%	11.88ms	12.4 M
Ours - GPU	99.49 %	98.57 %	31.59 ms	2.3 M
Ours - Edge TPU(Rasp. Pi 4)	98.90 %	95.59 %	8.79 ms	2.3 M

In our results, in Table 2.9, we compare the performance and inference times of the original, not quantized model running in GPU, and the 8-bit quantized model running in the TPU. Furthermore, we also implemented the methods from past approaches, named Nimbro and Sweaty, and presented the results for the same dataset. The Nimbro and Sweaty approaches were implemented using PyTorch and ran on the same GPU. All of the implementations, including the Edge TPU compiling code, are available as open-source software hosted on GitHub. To validate

our method, a new dataset of pictures of a soccer field is built. There is a field based on the official Robocup field at a reduced size for the perception system. The pictures are annotated for six classes of interest: ball, goal post, field center, penalty mark, corner lines, and T-junctions. The annotations represent the XY coordinate of the center pixel of the object of interest. The annotations are used to paint blobs around the center of the objects of interest. The blobs are generated using a 2D multivariate Gaussian distribution with a covariance of 4.0 for both dimensions. It means that pixel values in the annotations (and the regressed heatmap) represent probability densities. Example pictures from our dataset and its labels are shown in Figure 2.24. The dataset contains a total of 1992 pictures. 15% (298) of the pictures were randomly selected for a holdout validation set, while the remaining 1694 were kept as the training set. Furthermore, for each picture on the training set, data augmentation was used to generate two more pictures, increasing the size of the training set threefold to a total of 5082 pictures. Data augmentation and the training-validation split were done on disk so we could use the exact same data splits for all the models.

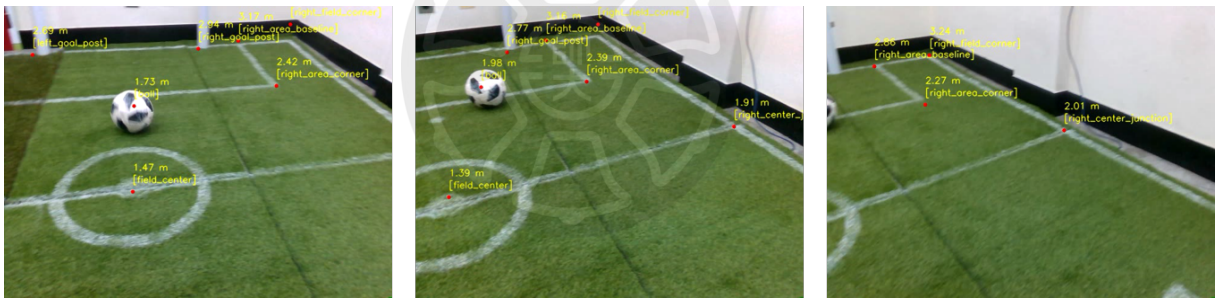


Figure 2.24: Results for six classes of interest: ball, goal post, field center, penalty mark, corner lines, and T-junctions

The results of our proposed method on the Edge TPU and GPU are shown in Table 2.9, along with the results from our implementation of the comparison methods. Precision and recall are defined as is standard: $P = TP/(TP + FP)$, and $R = TP/(TP + FN)$, and are averaged over all classes to compute average precision and average recall. A true positive (TP) is defined as a predicted blob with at least 50% overlap with the ground truth annotated Gaussian blob. We can see that the overall precision and recall are pretty similar for all models except for the Nimbro model, which performed quite a bit worse. Nimbro is the only model to use a 3-channel heatmap output to color code the class blobs. This suggests that the approach of Sweaty and ours of predicting each class as a separate channel is the superior one. On the inference time

spectrum, our approach is the slowest one when running on the GPU, while the smallest model Sweaty-V3 was the fastest. Moving along, we can see that our model still clocks in a respectable inference time at just 7.44 ms when running on a Raspberry Pi with the Edge TPU. This means that it can run inference at more than 130 frames per second at almost the same performance accuracy on a GPU. The small performance hit observed is certainly an effect of the quantization operation, which reduces the numerical precision of the operations.

2.10 Discussion

In conclusion, Chapter 2 provided an overview of the humanoid robot platforms and their essential components, focusing on the mechatronic system, walking gait algorithm, and perception system. The mechatronic system of the Robinion series was described as a cost-effective yet robust platform. It incorporates advanced technologies such as ZMP-based walking, compliant joints, and lightweight structure, enabling stable and efficient walking gait for the humanoid robot platforms. The walking gait algorithm was presented, emphasizing the importance of inverse kinematics solvers and gait trajectory generators in achieving stable and balanced walking. The experiments conducted on artificial turf demonstrated the robots' ability to maintain stability with varying step sizes. The results validated the effectiveness of the walking gait algorithm in challenging environments. The perception system, designed for autonomous soccer play, was highlighted as a critical component for recognizing essential elements on the soccer field. The deep learning-based approaches, including the Tiny YOLO-v3 and MobileNet models, showed superior performance compared to traditional image processing techniques like the Hough Circle transform. The evaluation results demonstrated higher precision, recall, and accuracy of the deep learning models, along with faster inference speeds. Furthermore, the application of heatmap-based object detection was introduced, providing a visual representation of the probability distribution for objects. The evaluation of different heatmap models presented the effectiveness of our proposed method, which achieved high precision and recall while maintaining efficient inference times. The findings presented in Chapter 2 validate the reliability, adaptability, and efficiency of the Robinion series platforms and their associated algorithms. The optimized mechatronic system, walking gait algorithm, and deep learning-based perception system contribute to the performance and functionality of the humanoid robot platforms, par-

ticularly in autonomous soccer. These advancements show the potential of humanoid robots in real-world applications and the significance of integrating cutting-edge technologies to enhance their capabilities.

Chapter 3. Literature Review for Balance control

This chapter serves as the literature review on the problem of balance control in a highly dynamic environment, focusing on the specific case of Robinion2S balancing on a balance board. Initially, we tried to approach the solution through simple conventional control strategies, implementing a Proportional-Integral-Derivative (PID) controller on the real robot. PID controllers have long been the backbone of control engineering, offering an intuitive, three-term control strategy that adjusts system outputs based on the proportional, integral, and derivative terms of the error signal [67, 68, 69]. However, despite its strengths, the PID controller alone was not sufficient to handle the complexity and dynamism of the balance problem effectively. Given the result, our focus shifted towards more sophisticated algorithms, capable of rising to the challenges posed by the balance board. We pivoted to reinforcement learning (RL), which emphasizes learning by interaction and has shown great promise in the realm of control optimization [70]. Utilizing the powerful, GPU-accelerated simulation environment provided by Isaac Gym, we trained Robinion2S on the simulator with PID & Inverse Kinematics (IK), IK positions, and joint angles using Proximal Policy Optimization (PPO) [71]. After the training phase, we needed to transfer the skills learned by Robinion2S in the simulated environment to the real world. To do this, we employed a technique known as Sim2Real, which leverages the knowledge gained in a simulated environment to improve performance in a real-world setting [72]. This chapter explains into each of these components in detail - PID, RL, and Sim2Real - explaining their theory, application, and contribution to solving the balance problem.

3.1 PID

Feedback control theories are required to enable legged robots, such as humanoid robots, quadruped robots, and legged-wheel robots, to balance and maneuver through various environments (see Figure 3.1). The most standard one of the control method used in the field of robotics is the PID controller, which stands for Proportional-Integral-Derivative controller. The favor of PID controllers can be attributed to the simplicity and efficiency of the controller. The PID controller combines the P (Proportional), I (Integral), and D (Derivative) controls, each contributing a different aspect to the overall control strategy. To generate an optimal response, the gains for

P, I, and D are adjusted. The PID gains are tuned to achieve fast response, reduce overshoot, and attain steady-state stability, which are all critical for balance control.

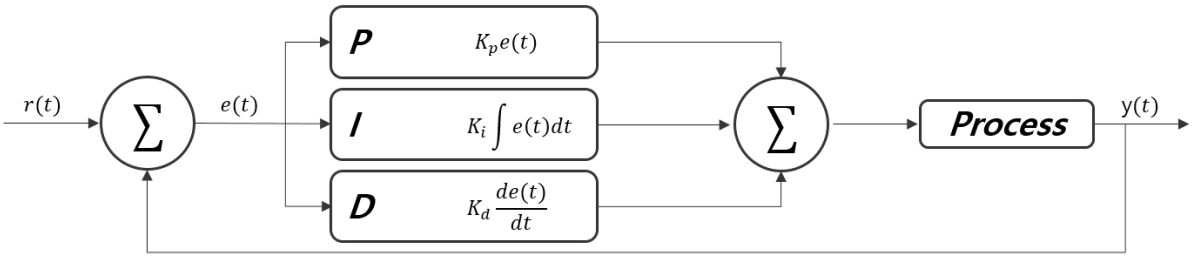


Figure 3.1: Block diagram of a PID controller in feedback loop

The Balance control of legged robots is a complex task that requires an intricate understanding of the current state, which are the position and orientation, which is typically achieved through the use of various sensors [73]. The sensors provide the feedback needed for the PID controller to adjust the balance of the robots. There are considerable sensors for the balance control for the legged robots, such as the Inertial Measurement Unit (IMU), Force-torque Sensor, Encoder, and Vision System [74, 75]. The use of Force-Torque (FT) sensors with a PID control system offers a more dynamic approach to balance control for legged robots (see Figure 3.2).

Figure 3.2: Six-axis Force-Torque(FT) Sensors

FT sensors measure the force and torque applied on the platform, providing the contact dynamics sensitivity between the ground and the feet of the robot [76, 77]. When applied to a legged robot, the FT sensor typically resides at the feet of robots or is integrated into the leg joints. The FT sensors provide information about the ground reaction forces and moments experienced at each foot, which is essential for maintaining balance, particularly during dynamic tasks like walking or running. The PID controller, coupled with Force-Torque sensors, effectively controls the balance of a legged robot by taking appropriate actions based on the force and torque deviations from the equilibrium state. The approach, FT sensors and PID, to balancing

control is particularly useful in situations where the robot has to navigate over uneven terrain or perform dynamic maneuvers.

The use of a vision system in balance control for legged robots offers methods for maintaining stability [78]. The vision system, typically composed of mono and depth cameras, provides the robot with an understanding of its surrounding environment and its own state within it. It is possible to consider a legged robot using a vision system for balance control with the PID controller. The vision system can capture images or videos of the environment, which can then be analyzed to infer the tilt. If the robot is leaning to one side, it can be deduced from the visual data. The tilting information serves as the input error for the PID controller. However, Image capture, processing, and analysis can be quite time-consuming. Thus, the control system has to be designed to ensure real-time processing to balance the robot using PID effectively. It is also important to remember that a vision system can be affected by lighting conditions. Therefore, relying solely on the vision for balance control may not be sufficient in all scenarios, and it is often beneficial to use it in conjunction with other sensors, such as IMU and Force-Torque sensors, for a more robust solution.

To solve the balance problem, legged robots necessitate an IMU to obtain essential information such as angular velocity, Euler angle, and quaternion [79]. The IMU is a device that contains a three-axis accelerometer and a three-axis gyroscope, and some IMUs include a three-axis magnetometer to overcome yaw drift (see Figure 3.3). The accelerometer measures linear acceleration, including the force of gravity, which can be used to determine the orientation relative to the Earth. The gyroscope measures the rate of rotation around each of the three axes, which can be integrated over time to compute the current rotation or orientation.

The PID controller is utilized in the balance control of a legged robot when the IMU data is compared with the desired setpoint. In legged robots, the IMU is used to monitor the orientation of the robot with the Euler angle. The PID controller could take the desired tilt, which is usually upright, and the measured tilt as input, compute the error and then adjust the angles of the robot joints to reduce the error for a stable balance. If the robot starts leaning in any direction, the IMU will detect the tilt. The proportional component of the PID controller would react proportionally to this error, meaning the more the robot leans, the stronger the response. The

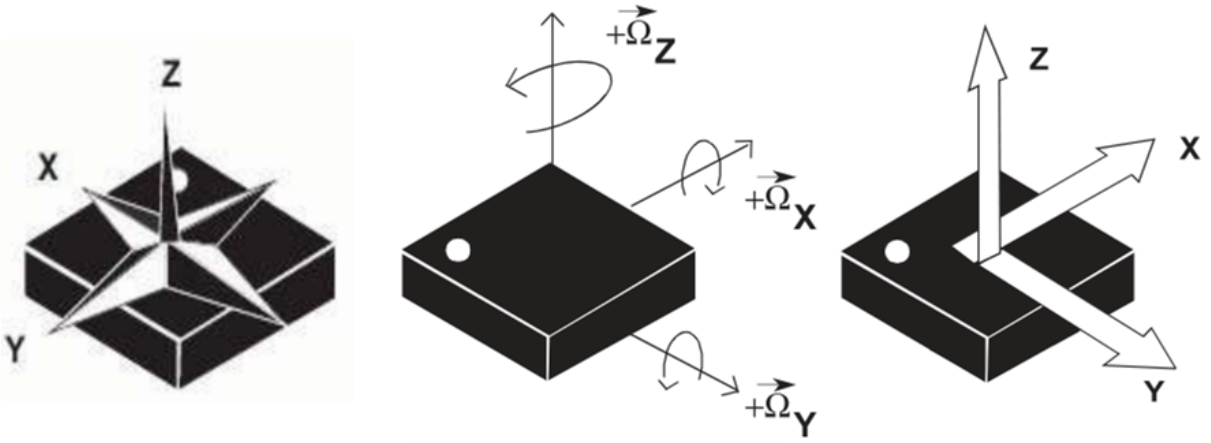


Figure 3.3: Three key properties of IMU (left: magnetometer, middle: gyroscope, right: accelerometer)

response will give an action to move in the opposite direction to prevent a fall. However, the P control might result in overshooting and causing the robot to fall down. The integral component considers not just the current error but the accumulation of past errors. It helps to eliminate the residual steady-state error that occurs when the proportional control cannot bring the robot exactly to the upright position. If the legged robot is consistently leaning slightly, the integral term will gradually increase to push the robot back to the upright position. The derivative component predicts future errors based on the current rate of change. It can dampen the system, preventing overshooting caused by the proportional and integral components. If the robot is rapidly leaning, the derivative term will work to slow down this change. The PID controller continuously adjusts the movements based on the sensed orientation from the IMU, ensuring the robot balances continuously. Fine-tuning the PID controller's gains is critical to achieving stable and responsive balance control. Different-legged robots may need different PID gains due to variations in size, weight distribution, and mechanical design. For balance control of legged robots with IMU data as input for PID, it is possible to be designed different control modes such as the Proportional (P), Proportional-Integral (PI), Proportional-Derivative (PD), and Proportional-Integral-Derivative (PID) (see Equation 3.1).

$$
\begin{aligned}
P &: u(t) = K_p e(t) \\
PI &: u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau \\
PD &: u(t) = K_p e(t) + K_d \frac{de(t)}{dt} \\
PID &: u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}
\end{aligned}
\tag{3.1}
$$

In Equation 3.1, $u(t)$ represents the control signal at a given time t, which is the output of the controller that will actuate the system. The K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively, which determine the influence of each term on the control signal. The gains are generally determined through system identification or trial and error. $e(t)$ denotes the error signal at a time t, calculated as the difference between the desired and the actual system output. The error signal is what drives the control action, aiming to reduce the discrepancy between the system's actual and desired output. P control is the most straightforward type of control. The control action, such as the adjustment to the robot movement, is proportional to the error. The error, in this case, could be the difference between the desired robot posture (e.g., upright) and its current posture (e.g., leaning to one side). If the error is large, the control action will also be large. This makes the robot react more enormously when the error is high. However, P control alone is usually not sufficient for the balance control of legged robots. While it can help correct substantial errors quickly, it may not eliminate the error completely, resulting in a steady-state error. For instance, legged robots might continuously oscillate around the desired upright position but never settle in the correct position. To eliminate the steady-state error, P control can be PI control by adding an integral term. The PI control takes into account not just the current error but the accumulation of past errors. It helps to eliminate the constant offset error that might persist with a P-controller. If legged robots are slightly off balance over a long period, the accumulated error becomes significant, and the integral term generates a large control signal to correct it. The risk is that a large error over a prolonged period can make a high oscillation response when the error is finally corrected. Another extension of proportional control is PD control. The derivative control acts on the rate of change of the error. If the balance of legged robots is rapidly deteriorating, the derivative term generates a strong control signal to counteract it. The derivative term inherently has a damping effect, which can prevent the overshoot and

oscillations that might occur with P or PI controllers. PD control might be a good choice for balance control in legged robots because it can react quickly to changes (P-term), dampen out oscillations, and provide stability (D-term). However, PD control will not eliminate steady-state error if one exists. PID control combines all three elements and is available to provide rapid response, stability, and steady-state error correction. For legged robot balance control, PID control adjusts the angles of the robot joints based on the error in the robot posture measured by IMU, the accumulated error over time, and the rate of change of the error. However, the implementation and tuning of PID control for legged robots is a complex task. Each of the PID terms must be carefully tuned according to the dynamics of robots. Considerably emphasis on the proportional term leads to instability, largely on the integral term leads to a sluggish response, and broadly on the derivative term leads to excessive sensitivity to sensor noise. For balance control of Robinion2S in a highly dynamic environment, the Proportional-Derivative (PD) controller is implemented. Since the PD controller aims to foster a swift system response and ensure the stability of the robot [80]. The controller is critical for the unpredictable environment, where rapid changes can occur. Nevertheless, it is also worth noting that while PID controllers are widely used, they may not be sufficient for more complex tasks and environments. Advanced control techniques such as linear quadratic regulator(LQR), model predictive control (MPC), reinforcement learning, or a combination of these might be needed for challenging locomotion tasks and rough terrains [81, 82].

3.2 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning approach in which an agent learns how to behave in an environment by performing certain actions and observing the results [83, 84]. The goal of the agent is to learn a policy, which is a mapping from states to actions, that maximizes the cumulative reward over time. Components in the RL setup are the agent, the environment, states, actions, rewards, and the policy (See Figure 3.4). The components interact in several steps for updates. First, the agent observes the current state of the environment. And then, based on its observation, the agent selects an action to perform. The environment transitions to a new state based on the performed action. And the agent receives a reward or penalty from the environment depending on how beneficial or detrimental the action was. Finally, the

agent updates its policy based on the reward and the observed transition. The interactive process continues until a termination condition is met. The goal of the agent is to learn a policy that maximizes the expected cumulative reward, also known as the return, from each state.

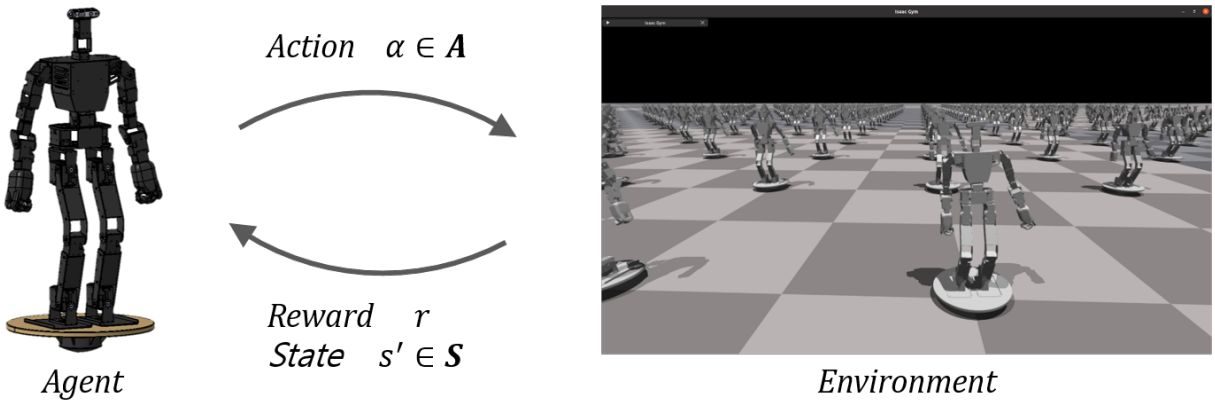


Figure 3.4: A simplified diagram of the Reinforcement Learning loop

In the robotics field, RL is used in a variety of approaches to solving various complicated problems. There are several reasons why roboticists are increasingly leveraging RL for controlling robots [85]. The first reason is Ability to handle complexity. The legged robots are highly complex systems with many degrees of freedom and non-linear dynamics. Analytical methods for control, such as traditional PID, although effective for simpler systems, can become highly complex or even infeasible when applied to the full dynamics of a legged robot. RL, on the other hand, can handle such complexity by learning from the interaction with the environment, making it a powerful tool for these tasks [1]. The second reason is Adaptability [86]. RL methods can learn and adapt to changes in the environment or the robot itself. If the robot's dynamics change, for example, due to wear and tear or carrying different loads, an RL controller can adapt its behavior through further learning. This adaptability can be critical for long-term autonomy in changing environments. And the third reason is Generalization [87]. Once trained, RL methods have the potential to generalize across different but related tasks. For example, a legged robot trained in a variety of environments could potentially adapt to new environments more readily, given that they share similar characteristics with the training environments [88]. The fourth reason is Optimality. RL techniques have the ability to find optimal or near-optimal solutions given enough training time [89]. This could potentially lead to more efficient or robust control strategies than those designed by traditional control theories. The last reason is Learning from scratch. One of the most interesting aspects of RL is that it enables robots to learn complex

tasks from scratch. The robot starts with random behavior, and through repeated interaction with the environment, it gradually learns to perform the task. This means that RL can potentially discover novel control strategies that humans might not think of. However, it is also worth noting that while RL holds a lot of promise, it also has its challenges. These include the need for large amounts of data, the difficulty of specifying appropriate reward functions, and issues with reproducibility and reliability. Yet, despite these challenges, the potential benefits of RL for legged robot control are driving a lot of exciting research in this area.

There are several RL methods that are employed for balance control in legged robots. The approaches include methods such as Q-Learning, Deep Q-Networks (DQN), and Soft Actor-Critic (SAC), Proximal Policy Optimization (PPO), among others [71, 90, 91, 92]. Q-Learning is a values-based learning algorithm in RL. Instead of directly optimizing the policy, it learns a value function, which is a measure of how good it is to take a particular action given a state. Once the agent has a good estimate of this function, it can derive a policy by choosing the action that gives the maximum value for each state. Q-Learning is quite efficient and can be highly effective in scenarios with a smaller state-action space. However, for complex problems like balance control in legged robots, we often need to combine Q-Learning with other techniques, such as function approximation, to make it feasible, leading to methods like DQN. DQN is an extension of Q-Learning, which uses a deep neural network as a function approximator to represent the Q function. This makes it possible to deal with high-dimensional state spaces, such as images from a camera feed, which would otherwise be infeasible to handle with basic Q-Learning. For example, a DQN agent could use a stream of images as input (state) and output the Q-value for each possible action the robot can take. The agent can then select the action with the highest Q-value to execute. SAC is a recent approach that combines ideas from DQN and Actor-Critic methods. It learns a policy like PPO but also maintains a value function as in DQN. It encourages the agent to explore a wide range of actions by incorporating an entropy term into the objective, which encourages randomness. This can lead to more robust policies that are better able to handle a wide range of scenarios. SAC can show impressive performance on a range of challenging control tasks, including balance control in legged robots. PPO is a type of policy optimization method that we mentioned for training agents. It is widely used due to its effectiveness and efficiency. It is an on-policy method, meaning it learns a policy that

directly decides the action to take based on the current state, and it learns from the experiences generated by the current policy. When applied to balance control in legged robots, PPO utilizes the robot's sensor readings, such as pose and velocities, as input states. The actions would be the desired joint torques or positions. The PPO agent aims to learn a policy that can maintain the robot's balance, meaning the robot stays upright and minimizes tilting. This makes PPO more stable and reliable for complex tasks such as balance control in legged robots. To solve the intricate challenge of maintaining stability for the Robinion2S robot on a balance board, the robot is trained with PPO, a sophisticated reinforcement learning algorithm, within the powerful environment provided by Isaac Gym from Nvidia (see Figure 3.5. The balance board problem with Robinion2S presents a complex and dynamic environment, requiring an algorithm capable of handling continuous states and actions with great finesse. While the PID controller is a powerful tool for tackling various control problems, it might not be sufficient on its own when dealing with a highly dynamic environment such as the balance board problem. Such situations demand more than just the adjustment of system outputs based on the proportional, integral, and derivative terms of the error signal. Therefore, when faced with these dynamic problems, it may be necessary to integrate the PID controller with other control strategies or employ more advanced control algorithms that can better capture and respond to these complexities. Therefore, PPO, with its unique characteristics of enforcing a boundary on the changes to the policy at each update, provides a pragmatic solution, ensuring stable, yet efficient learning. Implemented within the highly optimized and parallel environment of Isaac Gym, the PPO algorithm helps Robinion2S learn to proficiently navigate the delicate balance dynamics, ultimately achieving sustained control and stability on the balance board.

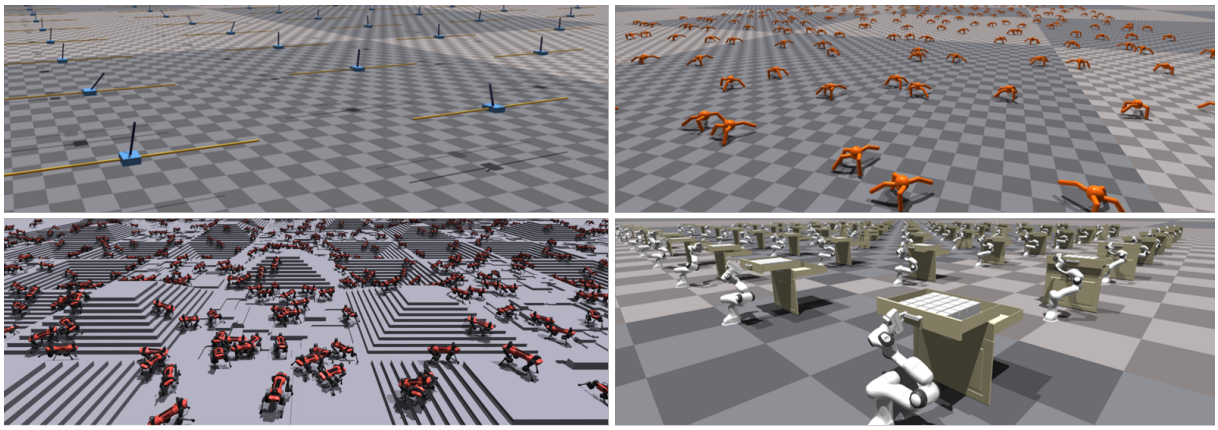


Figure 3.5: Examples to solve simple and complex tasks on Isaac Gym

3.2.1 Proximal Policy Optimization - PPO

Proximal Policy Optimization (PPO) is an advanced reinforcement learning method that utilizes several techniques to stabilize training and improve performance. PPO was introduced by OpenAI, and it represents an attempt to combine the benefits of Advantage Actor-Critic (A2C) and Trust Region Policy Optimization (TRPO) [93, 94]. It tries to take a step toward the direction of the steepest ascent, just like in A2C, without taking too big of a step like in TRPO. This makes the algorithm more stable and prevents it from making drastic updates that could harm learning. Especially, PPO is the algorithm widely used for training reinforcement learning models in an environment with continuous states and actions, such as controlling a legged robot for balance. To carry out the Proximal Policy Optimization (PPO) approach, there exists a sequence of defined steps. First, we initialize the parameters of the policy network (θ) and the value function (ϕ). The policy network generates the policy $\pi_\theta(a_t|s_t)$, the probability of taking action a_t given state s_t. The value function $V_\phi(s_t)$ estimates the expected return (accumulated rewards) at state s_t. Then, the agent interacts with the environment to collect experiences. These experiences form trajectories consisting of states, actions, and rewards. The advantage function quantifies the relative value of an action in a given state, compared to the average value of all possible actions in that state. It is used to determine how much better or worse an action is, compared to the average action. Equation 3.2 is used to measure the difference between the new policy and the old policy.

$$r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{3.2}$$

Computing the surrogate objective is the main objective of PPO that is being optimized (see Equation 3.3). It introduces a clipped version of the objective function to prevent overly large policy updates.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r(\theta)\hat{A}_t, \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{3.3}$$

In Equation 3.4, the parameters of the policy are updated using the gradient ascent method with the Adam optimization algorithm.

$$\theta \leftarrow \theta + \alpha \nabla_\theta L^{CLIP}(\theta) \tag{3.4}$$

Equation 3.5 loss function quantifies the difference between the predicted value of a state and the actual return.

$$L^{VF}(\phi) = \frac{1}{2}\hat{\mathbb{E}}t[(V\phi_{\text{old}}(s_t) - \hat{R}_t)^2] \tag{3.5}$$

The parameters of the value function are updated using the gradient descent method with the Adam optimization algorithm (see Equation 3.6).

$$\phi \leftarrow \phi - \alpha \nabla_\phi L^{VF}(\phi) \tag{3.6}$$

The entire procedure is repeated for N iterations to improve the policy and the value func-

Algorithm 1 PPO Algorithm

Initialize policy parameters θ, and value function parameters ϕ
for iteration=1, 2, 3, ..., N **do**
  Collect a set of trajectories by executing the current policy in the environment
  At each timestep in the trajectories, compute the advantage estimates $\hat{A}_1, \hat{A}_2, ..., \hat{A}_T$
  Optimize the surrogate objective:
  for optimization epoch=1, 2, ..., K **do**
    for minibatch in the collected trajectories **do**
      Compute the ratio $r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$
      Compute the surrogate objective $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r(\theta)\hat{A}_t, \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$
      Update the policy by maximizing the PPO-Clip objective via Adam: $\theta \leftarrow \theta + \alpha \nabla_\theta L^{CLIP}(\theta)$
    end for
  end for
  Fit value function by regression on mean-squared error:
  for fitting epoch=1, 2, ..., M **do**
    for minibatch in the collected trajectories **do**
      Compute the value loss $L^{VF}(\phi) = \frac{1}{2}\hat{\mathbb{E}}_t[(V_{\phi_{\text{old}}}(s_t) - \hat{R}_t)^2]$
      Update the value function by minimizing the value loss via Adam: $\phi \leftarrow \phi - \alpha \nabla_\phi L^{VF}(\phi)$
    end for
  end for
end for

tion continuously. During each iteration, the agent collects new trajectories under the current policy, updates the policy to encourage actions with higher advantage, and refines the value function based on the observed returns. The PPO algorithm for training is presented in Algorithm 1.

3.3 Real2Sim2Real

Handling the balancing problem of legged robots in dynamic environments necessitates a sophisticated approach. The state-of-the-art solution leans towards the utilization of Sim2Real, a technique that takes advantage of reinforcement learning within a simulated environment, then applies the trained model to the real robot [95, 96, 97]. Simulation platforms are specifically specialized towards reinforcement learning, such as Isaac Gym and pyBullet [98]. Isaac Gym from Nvidia performs outstandingly due to GPU-accelerated training. The remarkable capability allows it to solve a myriad of challenges associated with legged robots effectively. GPU technology of Isaac Gym offers a significant advantage in conducting extensive simulations at a faster rate. To ensure the validity and transferability of the training results, it is important to reduce the gap between the simulation and real-world conditions. This is where the concept of Real2Sim becomes pivotal [99]. Real-to-Simulation (Real2Sim) strives to mimic the actual environment within the simulator as accurately as possible, both in terms of the physics laws governing the robot's interactions and the robot's behavioral dynamics. Real2Sim is a meaningful process in the field of robotics and AI to solve complicated tasks efficiently in the real world. This process involves using real-world data and experiences to create more accurate and reliable simulation models. The primary goal of Real2Sim is to make the simulated environment as close to the real-world environment as possible, in terms of both physics and behavior. This process is crucial in robotics, where simulations are often used to train and validate models before they are deployed in real-world scenarios. There are a few key steps and methods commonly used in the Real2Sim process.

- ***Data Collection***
  - surface condition, inertia, friction, damping, and more
- ***Parameter Estimation***

  - Estimate the parameters for dynamics, joint torques, and friction coefficients

- ***Simulator Tuning***
  - Reflect real-world dynamics with the parameters

- ***Verification***
  - Compare the results of the real world and the simulator

In legged robots and Isaac Gym, it might involve mimicking the physical properties, like mass, friction, speed, direction, and inertia, of the robot joints and links in the simulation. It also includes modeling the physics of interactions between the robot and its environment - such as the forces experienced while walking or maintaining balance on uneven terrain. For instance, if legged robots walk on different terrains, the Real2Sim process might involve creating different ground conditions in Isaac Gym, each one reflecting a specific type of terrain, such as stairs and objects, in the real world. It might also include capturing the real-world physical parameters of these terrains, like friction coefficients, and integrating them into the simulation. This way, the robot can learn to adapt to various terrains in the simulation before being deployed in the real world.

On the other hand, Sim2Real is the process of transferring knowledge or behaviors learned in a simulation environment to a real-world system. This is typically done by training an agent in a simulator and then deploying the learned behaviors or control policies onto the real robot. There is a more detailed Sim2Real process with the simulator.

- ***Simulation Setup***
  - Set up the tasks, robot, and environments in the simulator

- ***Policy Training***
  - Train agent using RL, PPO algorithm, in the simulator

- ***Domain Randomization***

  - Ensure the trained policy is robust and generalizable

- ***Policy Transfer***

  - Transfer well-trained policy to the real robot

- ***Real-world Validation***

  - Test the real robot for the performance matches close to the simulator

In legged robots, a policy learned via reinforcement learning in Isaac Gym's environment needs to work effectively when applied to the actual robot. Suppose the robot has been trained in Isaac Gym to maintain balance on a board by shifting its weight. The Sim2Real challenge is to ensure that the same policy allows the real robot to balance on an actual balance board. Addressing the Sim2Real challenge often involves fine-tuning the learned model with real-world data, known as domain adaptation. This can help account for the 'reality gap' caused by simplifications or inaccuracies in the simulated environment. For example, we might fine-tune the balancing robot's model with data collected from real-world trials, improving its performance in the actual environment.

The use of Real2Sim and Sim2Real in Isaac Gym is about creating an accurate simulation of the real-world (Real2Sim) and ensuring that the learnings from this simulation can be effectively applied to the real-world (Sim2Real). This can significantly accelerate the development and deployment of legged robots by allowing them to learn complex tasks in a safe, controlled, and cost-effective manner before they are deployed in the real world.

Chapter 4. Balance Control with PID

This chapter places considerable emphasis on outlining the practical aspects of deploying a PID controller to solve the complicated task of maintaining balance for Robinion2S while it performs on the balance boards. As an integral part of the control mechanism, the PID controller is significant in providing stability and responsiveness to the robot. There is a comprehensive analysis of two challenging distinct types of balance boards for balancing Robinion2S (see Figure 4.1).

Figure 4.1: The balance boards for the proposed balancing control

The first type is a balance board equipped with a separate roller from the board, where the user balances stability by controlling only the roll angle of the board. This board demands proficiency in lateral balance, focusing specifically on the left and right oscillations. On the other hand, the second type of balance board presents a more dynamic challenge, requiring the user to concurrently balance both the roll and pitch angles of the board. This board involves a higher level of complexity as it necessitates control over both lateral and longitudinal balance. It is more challenging as it requires the user to consider balancing the board by controlling the roll and pitch angle together. The first balance board operates on a system where the roller maintains a line contact with the ground. This implies that a line on the roller contacts the ground at any given time. This type of balance board makes the task of balancing a controlling the roll angle. Contrastingly, the second type of balance board is engineered in a manner that the roller touches the ground only at a single point. This significantly raises the complexity of the balance task. With only a point of contact to support the board, the stability is significantly less compared to the line contact and requires control of both the roll and pitch angles. This results in a more dynamic balancing challenge, pushing the boundaries of the robot balancing capabilities and control algorithms. In order to solve the intricate dynamics of balance control, we implement a

PID controller that directly interfaces with the inverse kinematics position. The idea is to leverage the ability to minimize error over time through its proportional and derivative terms as a PD controller and utilize this feature to control Robinion2S. More specifically, the PD controller generates commands that control the joints to target positions computed through the inverse kinematics positions. In order to mitigate excessive oscillations that can potentially destabilize the robot, we implement a low-pass filter into the control system. The low-pass filter is applied to the output of the PD controller prior to its utilization in the computation of the inverse kinematics positions. The low-pass filter serves to smoothen the output, which is often associated with rapid and erratic changes in control commands. The filter system aims to ensure smoother and more stable control signals that effectively control the motion of Robinion2S without provoking oscillations. This strategy enables a more stable balance control, generating a response more adept for the dynamic problem of the balance board. Due to the limitations of tuning the PD gains with the trial and error method directly in Robinion2S, we adopt an alternative strategy leveraging the power of simulation. Specifically, we establish parallelized simulation environments within Isaac Gym by implementing the Real2Sim methodology [100]. Within this environment, we conduct an exhaustive exploration for optimal PD gains using a random search algorithm. The GPU-accelerated computation significantly expedites this process, allowing us to evaluate various gain combinations in a much shorter time compared to real-world tests. The objective is to identify the PD gains that affect the highest balance performance under simulation environments, which can then be transferred to the real robot for enhanced balance control.

4.1 Methodology

In this methodology section, we clarify the intricate process of achieving balance control on two different balance boards with the humanoid robot platform, Robinion2S. The approach is the utilization of IMU roll angle for the first board(see Figure 4.1 left) and roll and pitch angles for the second board(see Figure 4.1 right) as inputs to the PD controller, which subsequently generates the inverse kinematics (IK) positions in the x, y, and z coordinates as outputs. A low-pass filter is then applied to the output from the PD controller to lower oscillation before they are operated to control the IK of Robinion2S.

The block diagram illustrated in Figure 4.2 shows the process for controlling Robinion2S

Figure 4.2: Process for controlling Robinion2S on the first balance board

on the first balance board. This board is designed for uni-directional movement, allowing Robinion2S to shift only along the x-axis. It means that we consider the roll angle of IMU in the robot, which is the only variable needed for this control process. The roll angle detected by the IMU is directly fed into the PD controller. The PD controller, which uses the roll angle as its input, calculates the necessary inverse kinematics (IK) positions to maintain balance. However, the output from the PD controller occurs oscillation due to the high sensitivity of the p and d gains of the controller. The oscillation is mitigated by applying a low-pass filter, thus providing a more stable signal. This filtered output from the PD controller then serves as the input for the IK computations for Robinion2S. The computations use the filtered PD output to determine the joint positions from IK to balance on the board.

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt} \tag{4.1}$$

In Equation 4.1, the PD controller is described. The designated setpoint for the PD controller is 0°, implying that our goal is for Robinion2S to maintain a level orientation from the ground. The error, which is an important part of the PD controller, is thus calculated as the difference between the setpoint and the current roll angle measured by the IMU. The roll angle of the IMU calculated by the Euler angle provides the real-time actual orientation in the environment. By consistently adjusting the controller outputs to minimize this error, the PD controller governs the movements in a balanced state.

$$y[n] = (1 - \alpha) \cdot y[n-1] + \alpha \cdot x[n] \tag{4.2}$$

In the above Equation 4.2, the low pass filter is expressed. The y[n] is the filtered output of the PD controller, x[n] is the output of the PD controller as the input of the filter, and y[n-1] is the filtered output at the previous time step, and α is the smoothing factor, a value between 0

and 1. This simple equation defines a first-order low-pass filter, which can effectively smooth out rapid changes in the input x[n] over time.

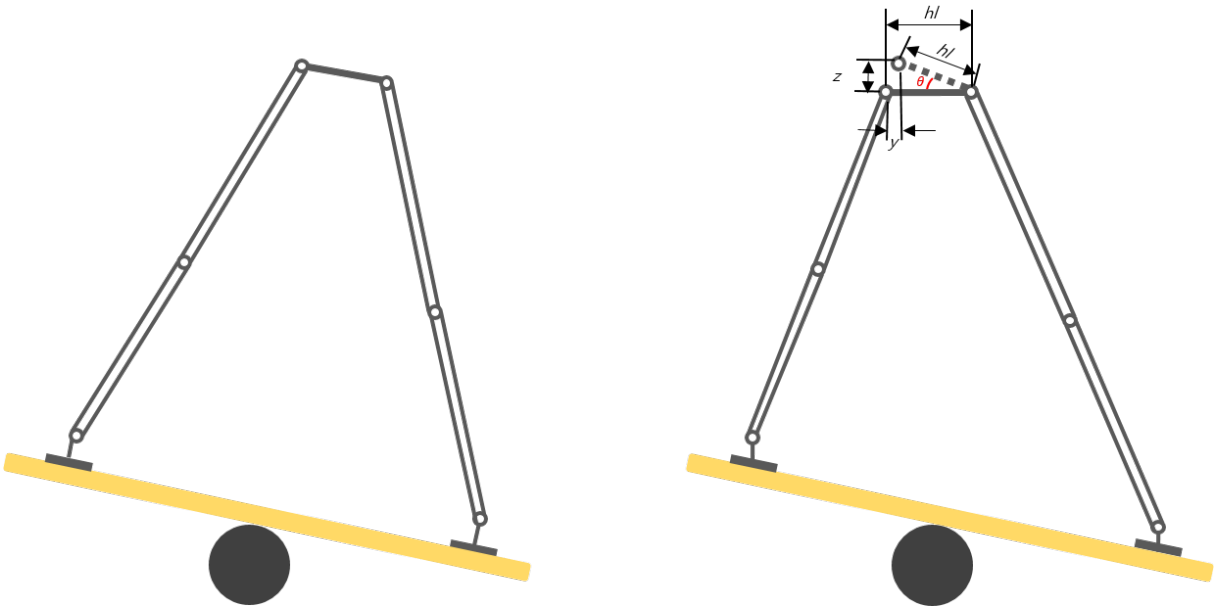


Figure 4.3: The motion of leg orientations for the first balance board

In our approach to maintaining the balance of Robinion2S on the balance board, we implement a strategy where the robot performs movements that are parallel to the ground. It means that Robinion2S calculates the Inverse Kinematics (IK) positions corresponding to the angle of the ground, derived from the output of the PD controller. The positions are then used to control the movements in such a way that they parallel the ground and move in the opposite direction of the tilt. The aim is to keep the robot body as level as possible, mirroring the ground surface for balance. Figure 4.3 depicts how the robot legs move in sync to maintain a parallel orientation with the ground, providing balance control to Robinion2S on the balance board.

$$
\begin{aligned}
y\_position &= y\_position + (hl - (hl * \cos\theta(PID\ output)) \\
z\_position &= z\_position + (hl * sin\theta(PID\ output))
\end{aligned}
\tag{4.3}
$$

To maintain the parallel orientation to the ground, Robinion2S is designed to orient its body towards the setpoint as determined value by the roll angle from the IMU. It means a dynamic adjustment of the robot poses in accordance with the real-time data on board tilt. Such responsive movement is calculated by an underlying mathematical framework that combines two inverse kinematics solvers. Firstly, we explained the analytic inverse kinematics solver for the standard

mechanism given by Equation 2.2. The equation computes the required joint angles for the desired foot position and orientation. However, to enable movements that maintain parallelism with the ground, we integrate one more analytic IK solver, which is Equation 4.3. The equation models the kinematics required for the robot to move its body orientation in response to changes in the roll angle of the ground, as detected by the IMU. By combining these two mathematical approaches, the robot is able to adapt to the varying ground inclination, thus balancing Robinion2S on the dynamic balance board.

Following tests with Robinion2S on the first balance board, we confronted a significant challenge: the robot was unable to maintain its balance on the board continuously. The root of the problem lay in the robot's incapability to estimate the position of the roller beneath the board. To solve the problem, Robinion2S would need a highly sensitive sensor system, such as advanced force-torque sensors or a global vision system. The sensors assist the robot in balancing on the board, but we wanted to build an environment where the Robinion2S could balance in human-like conditions. Although the actuators of Robinion2S could measure torque using the current sensor in the actuator, the high gear ratio resulted in a lack of the necessary sensitivity for this highly delicate task. Given the limitation, we made a strategic decision to shift our focus to the second balance board, where the roller is fixed directly under the board.

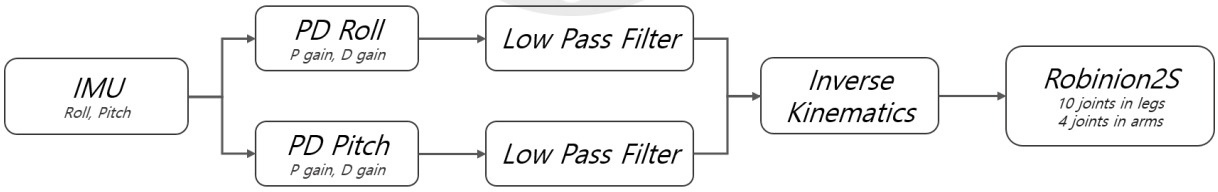


Figure 4.4: Block diagram designed for the balance control with Robinion2S

Figure 4.4 presents the block diagram designed for the balance control with Robinion2S on the second balance board. Unlike the first board, which involves control of only roll movement, the second board necessitates simultaneous control of both roll and pitch motions, adding a higher degree of dynamism and complexity to the task. The configuration of the control system is broadly similar to the first board, except that it contains two separate loops, one for each degree of freedom: roll and pitch. Each loop is equipped with its PD controller and Low Pass filter, designed to handle the dynamics associated with each axis. The control process works in a method like the one for the first board. The input to each PD controller is the IMU-measured

roll or pitch angle, which is then compared to a pre-set desired value, which is 0°. The output from the controllers is then processed through the low-pass filters to smoothen the output and attenuate oscillations. Following the filtering step, the outputs for both the roll and pitch are input into the IK solver. This ensures the robot makes the necessary positional adjustments, in both dimensions, to maintain balance on the second balance board.

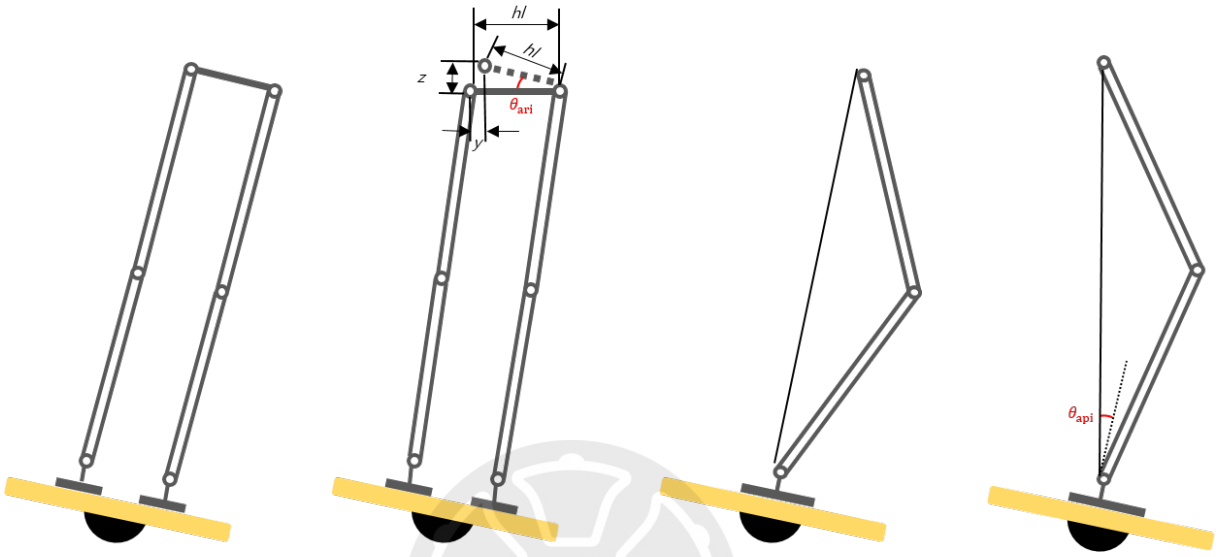


Figure 4.5: The motion of leg orientations for the second balance board

Similar to the control process for the first balance board, the key aim for the second board is to maintain a state of equilibrium that is parallel to the ground. However, an added level of complexity with the second board due to its requirement for managing both roll and pitch movements simultaneously, as depicted in Figure 4.5. In order to achieve this, Equation 4.3 is initially utilized to solve for the IK about the roll angle of IMU, effectively determining the robot's joint angles required to achieve the desired position and orientation.

$$\theta_{ap} = \theta_{ap} + \theta_{api} \tag{4.4}$$

Subsequently, Equation 4.4 is used as an additional compensation step to control the ankle's motion in the pitch direction. The compensation helps to adjust the robot's posture and ensure that it aligns parallel to the ground, thereby achieving the necessary balance on the more dynamic second board. Nevertheless, upon deployment, it became evident that the balance board presented a highly dynamic environment. The dynamism posed a challenge for the PID

tuning approaches. The conventional methods for finding suitable p and d gains, including trial-and-error, rule-based tuning, and the Ziegler-Nichols method, were insufficient in providing an optimal solution. Trial-and-error tuning, while practical in simpler control systems, proved inadequate for our task due to the balance board's dynamic nature and the complexity of the humanoid robot control. Rule-based tuning offered some initial guidance, but it fell short of delivering the necessary precision required for the highly dynamic balancing problem. Similarly, the Ziegler-Nichols method, a well-established technique for tuning PID controllers, was not flexible enough to handle the balance board's varying dynamics. There were limitations of traditional PID tuning methods in the experiments when dealing with highly dynamic and complex environments such as the balance board.

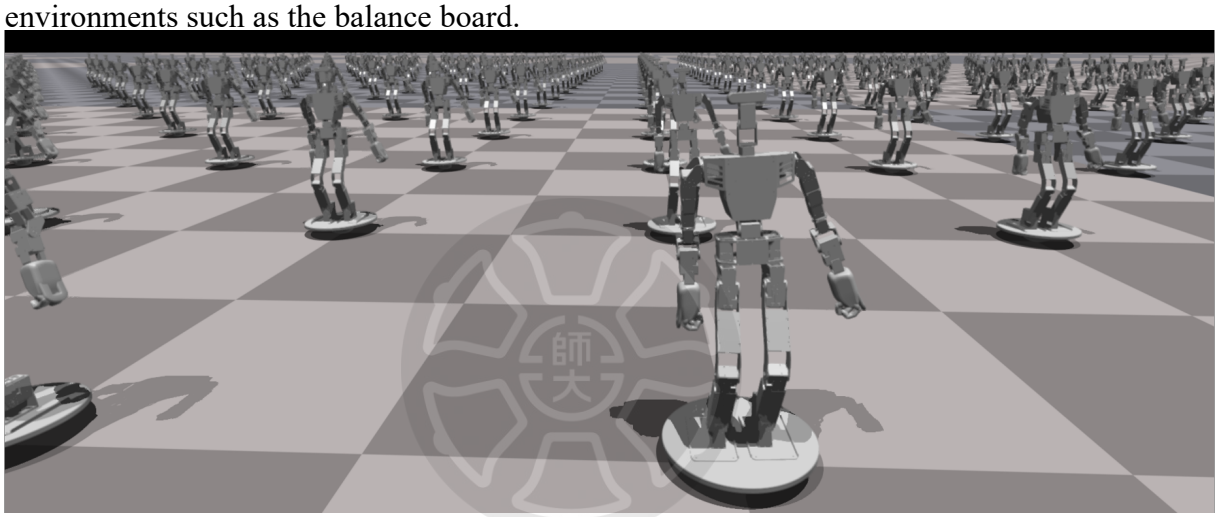


Figure 4.6: Isaac Gym parallel simulation environment for the balancing control on the balance board

To obtain optimized p and d gains, we changed the strategy using a simulation environment. To achieve finding the gains of the PD controller, we utilize Isaac Gym's parallel simulation environment, which offers a computational platform to simulate multiple environments simultaneously (see Figure 4.6). Sim2Real, creating a simulation environment that accurately mirrors the physical attributes of the real robot, is paramount. It entails a detailed process that begins with applying the real robot link mass within the CAD tool, a step crucial in obtaining an accurate inertia matrix that reflects the real robot characteristics. Beyond the inertia matrix, other parameters of Robinion2S are equally significant. It includes the directions of the joints, actuator torques, and the p and d gains of the actuator. This rigorous simulation setup provides an accurate and robust platform for the exploration and determination of optimized PD gains

for the balance control on the balance board. In our goal of achieving optimal balance control, we determine 11 distinct parameters, the precise tuning of which is to the successful balancing of the robot. The parameters include the range of the values for the PD controller for both roll and pitch angles, weights associated with roll and pitch angles, roll and pitch angles for both left and right arms, and the α value for the low pass filter. Optimizing the parameters is not an intuitive process. We apply a random search algorithm to efficiently traverse the high-dimensional parameter space, leading to a set of optimized values for each parameter. An extensive discussion detailing the range and specific values of these parameters, and the rationale behind these values, is presented in section 4.2.

4.2 Evaluation

In this evaluation section, we explain the analysis of three different experiments that we implemented to assess the balance control of Robinion2S on the two balance boards. The first experiment focuses on the real-world execution of the robot on the balance board, which designs to challenge the balance problem on only the roll axis. Secondly, we implement the second balance board, which adds an additional dimension to the balance control to overcome the limitation of the first balance board. This board demands balance control from the robot on both the roll and pitch axes, presenting a more dynamic environment and, thus, a more challenging task. As with the first experiment, the second experiment is also performed with the real robot, providing actual data on the robot's performance in real-world conditions. The last experiment in this section takes place in the highly capable Isaac Gym simulator. We implement the parallel environments in the simulator with the more dynamic second balance board capable of Sim2Real. This experiment is available to explore the optimized p and d gain with a random search algorithm while utilizing advanced features such as parallel simulation environments and realistic physics environments.

4.2.1 Balance Board #1

This section discusses the initial challenge of establishing balance control on the first balance board. This board presents a highly dynamic environment that necessitates an efficient approach to maintaining balance. For this purpose, we develop a control mechanism using the PD controller designed for the Robinion2S. The integral gain, a component of the well-known

PID controller, often generates oscillatory behavior in control systems. This fluctuation leads to instability, particularly in dynamically changing environments. To mitigate this risk, utilizing a PD controller instead of the full PID controller is decided. The PD controller effectively counteracts the potential for overshoot and the resulting oscillations that may be generated with P or PI controllers. This strategy is designed to offer a more stable and reliable response to the challenging balancing problem of the Robinion2S on the first board. However, even with the PD controller, there is a tendency for oscillations to occur when the P and D gains are amplified. The situation leads to unwanted instability and potentially undermines the balancing ability of Robinio2S. To counteract this, we design a low-pass filter into the system architecture. The filter works as a required element in moderating the fluctuations, guaranteeing that the output of the PD controller does not rapidly or excessively change from its current state. By blocking high-frequency oscillations, the low-pass filter provides an additional layer of stability to balance control for the Robinion2S.

Table 4.1: The best performance setting values for PD gains and Low pass filter α in the PID first experiment

Parameter	Roll p gain	Roll d gain	Low Pass Filter α
Value	1.5	0.5	0.004

In the pursuit of optimal PID gains that Robinion2S efficiently balances on the board, three tuning methods are engaged, including the trial-and-error strategy, rule-based tuning, and the Ziegler-Nichols technique. Among the tuning techniques, the trial-and-error method proves to be the most effective. It is due to its flexibility and repeatability, allowing for continuous refinement and adaptation based on the immediate performance feedback of the robot. By making iterative adjustments and checking the performance, the most efficient gains are obtained. After a series of experiments, the most suitable values for PD gains and low pass filter α parameters are reached. The optimal values, which maximized the balance performance of Robinion2S on the board, are presented in Table 4.1.

As detailed in Figure 4.3 and Equation 4.3, the posture adjustments of Robinion2S are calculated by the analytic inverse kinematics solver to guarantee the Robinio2S body maintains

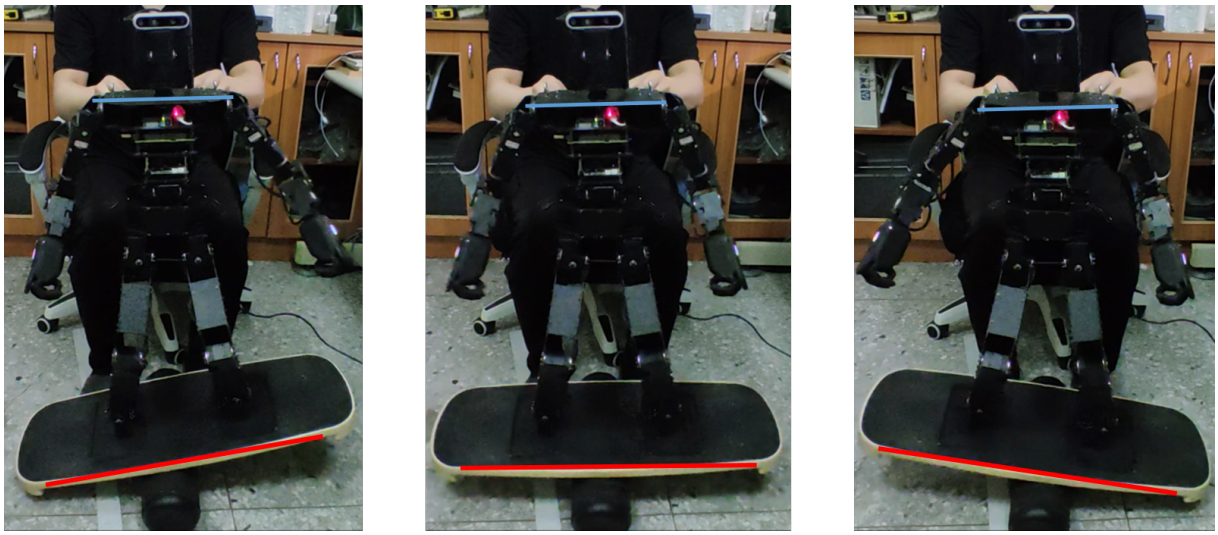


Figure 4.7: The result of the PD controller maintained parallel to the ground for the robot body

a parallel orientation with the ground. This is a crucial aspect of achieving balance control. Figure 4.7 illustrates the movement of Robinion2S in relation to the angle of the balance board. Despite the tilted board, the PD controller is effective in ensuring that the body maintains a parallel orientation to the ground. It is the effectiveness of the PD controller in balancing the dynamic environment. Despite efforts for balance control, we faced two significant challenges that were the unique values, which are PD gains and the α value of the low pass filter, could not solve continuous balance control on the board. Firstly, it became evident that pinpointing optimal PD gains for the varying states of the board was a challenging task. Only one optimal PD gain for the board was not effective for other states of the board. Secondly, Robinion2S had to estimate the roller position for balance control, but estimating the roller position proved impossible without equipping Robinion2S with an accurate force-torque sensor in its ankles. Both solutions were considered unsuitable for Robinion2S. The goal of the project is to achieve balance control with a cost-effective humanoid robot, which the need for an expensive, high-precision force-torque sensor would contradict. Furthermore, tuning PD gains for every possible board state would not be feasible in practice. Given these challenges, we decided to transition to using the second balance board. The second board is equipped with fixed rollers, which adds an extra layer of complexity to the balance control task as it now involves control over both roll and pitch axes. The change mitigates the problem of estimating the roller position. Figures 4.8 and 4.9 present the achieved results when using the first balance board in conjunction with the PD controller and the low pass filter.

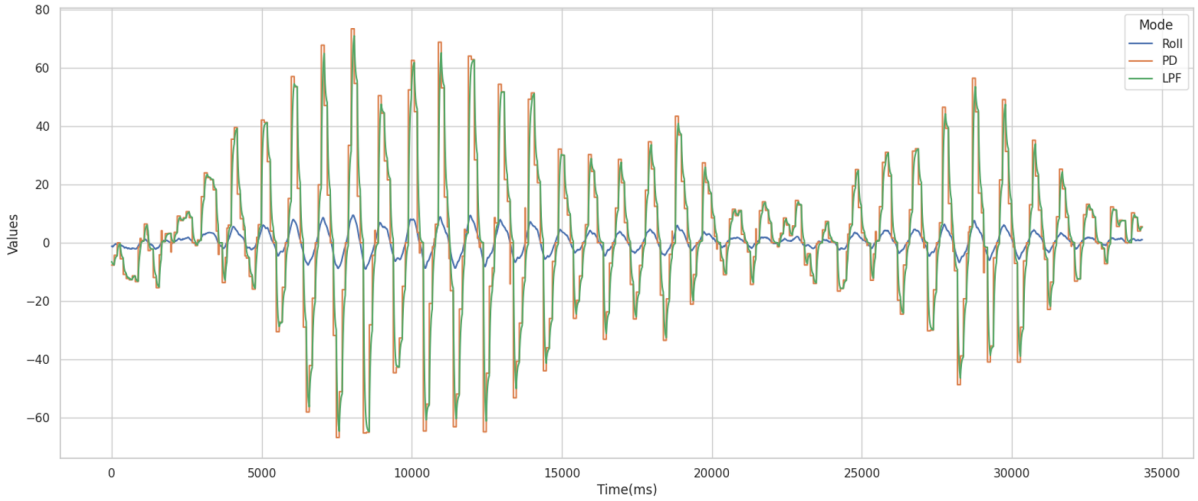


Figure 4.8: Balance control with IMU roll angle and the output of PD controller and Low pass filter with manual control of the user

Figure 4.8 represents the results achieved during the balance control test with the PD controller. The graph elucidates data from the IMU sensor, the output from the PD controller, and the results from the low-pass filter. Notably, human assistance is intervened in certain instances when the robot's movements are excessively fast, or the roller strays considerably from the center. As indicated by the data around the 5000ms mark, the output from the PD controller progressively escalates, prompting manual intervention from the user to help maintain balance. When such interventions are factored in, the graph demonstrates that it is possible for the robot to maintain balance for a period exceeding 30 seconds.

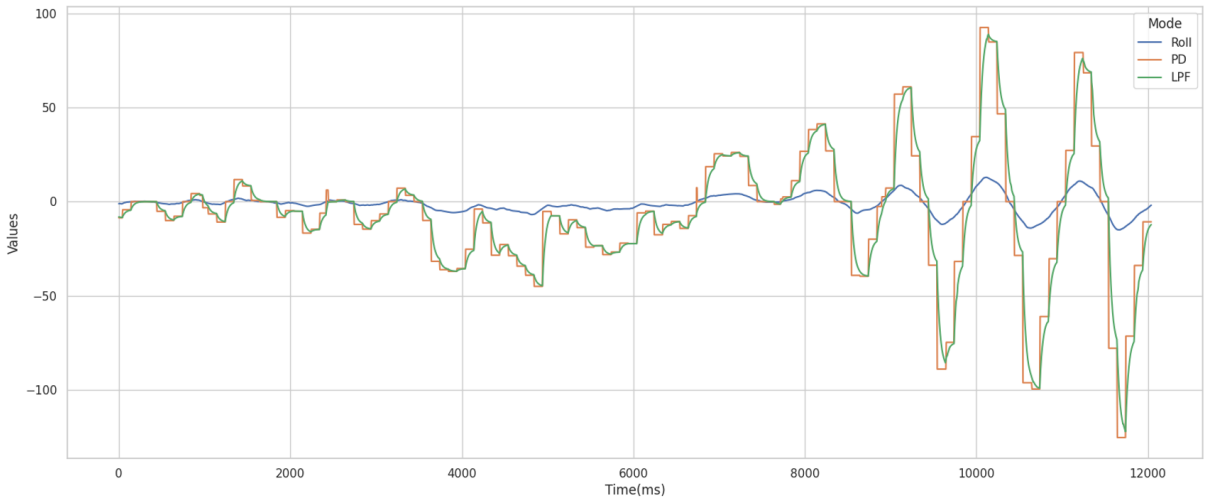


Figure 4.9: Autonomous balance control with IMU roll angle and the output of PD controller and Low pass filter

Figure 4.9 presents data pertaining to the autonomous balance control performed by the

robot without any manual intervention from the user. The graph illustrates the small changes in data for a span of around 7000 ms, right until the moment when the robot sets the initial position and gets positioned on the board. It then balanced on the board for roughly 4 seconds, where the robot attempted to maintain balance autonomously, starting around 8000 ms. The reason why the output of the PD controller increases is that the PD output value diverges as the balance board moves in the opposite direction after hitting the ground.

4.2.2 Balance Board #2

In alignment with Section 4.2.1, we employ a trial-and-error tuning method to determine the optimal setting values for the PD gain for both Roll and Pitch axes and the low-pass filter's α value. The determined values, which give the best performance in the experiments, are presented in Table 4.2.

Table 4.2: The best performance setting values for Roll & Pitch PD gains and Low pass filter α in the second PID experiment

Parameter	Roll p gain	Roll d gain	Pitch p gain	Pitch d gain	LPF α
Value	1.5	0.3	2.5	0.5	0.004

As displayed in Table 4.2, the PD gain value for the Roll axis is set lower than that for the Pitch axis. The result is because the Roll axis is more sensitive to oscillations, meaning that even slight adjustments in the P gain significantly impacted the balance control. Consequently, we fine-tuned the P and D gains for the Roll and Pitch axes to measure optimal balance control.

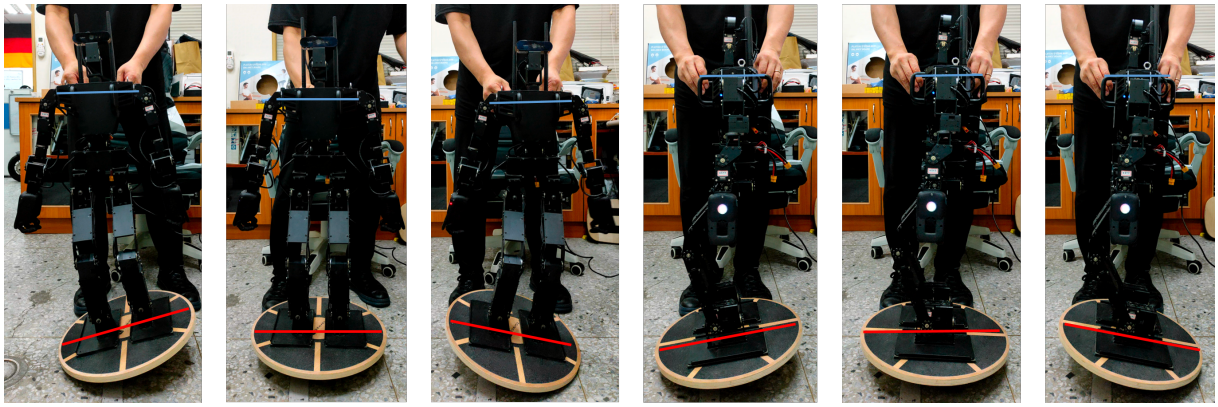


Figure 4.10: The result of the PD controllers of Roll and Pitch maintained parallel to the ground for the robot body(left three pictures: Roll, Right three pictures: Pitch)

As thoroughly elaborated in Figure 4.5, alongside Equation 4.3 and Equation 4.4, the pose performed by Robinion2S are controlled by the analytic inverse kinematics solver. It ensures that the robot's body, in terms of the Roll and Pitch angles, stays parallel to the ground. This element is required for achieving successful balance control. Figure 4.10 depicts how Robinion2S adjusts its movements in response to the changes in the Roll and Pitch angles on the second balance board. It effectively shows the interplay between the robot's motion and the dynamics of the balance board, indicating the significance of the implemented balance control strategy.

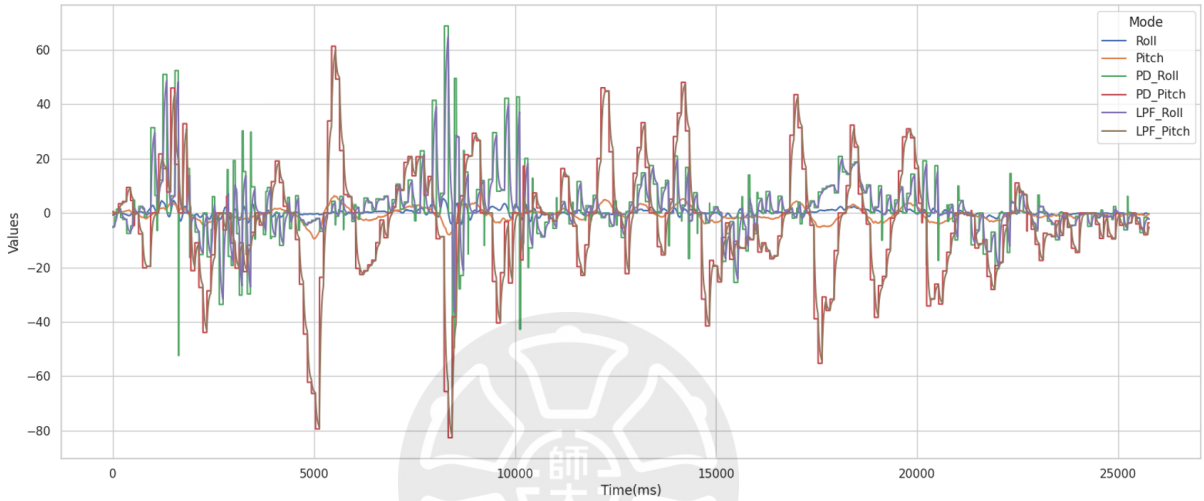


Figure 4.11: Balance control with IMU Roll and Pitch angles and the output of PD controllers and Low pass filters of Roll & Pitch with manual control of the user

Figure 4.11 presents the results derived from the balance control experiment using the PD controllers, taking into account the Roll and Pitch values calculated from the IMU sensor. The graph traces the IMU sensor data for both Roll and Pitch, the corresponding outputs from the PD controller, and the final output after passing through the low pass filter. It becomes apparent from the graph that Robinion2S exhibits more oscillations when it moves predominantly in the Roll direction compared to when it operates along the Pitch direction. This result stems from an experimental setup where human intervention was employed to aid the robot in maintaining its balance and preventing any falls. The resulting data, as depicted on the graph, displays stable performance with no signs of divergent behavior. Human assistance provides a safety net for the robot, ensuring its continuous stability throughout the experiment. Equivalent to the challenges encountered with the first balance board, the second board also presents difficulties in manipulating all board states using a single PD gain for Roll and Pitch. Furthermore, the

trial-and-error tuning method is not an ideal approach for optimization as it cannot account for all possible combinations of integers and floating-point numbers that can be used as PD gains. This limitation of the trial-and-error method poses a challenge in achieving an optimal balance control approach.

4.2.3 Random Search with Balance Board #2

To solve the challenge and discover optimized PD gains for the balance board, we designed an experiment using Isaac Gym from Nvidia, which allows for high throughput in parallel simulation utilizing GPU capabilities. This environment enables comprehensive testing across a wider range of PD gains, potentially leading to finding more optimal values for balance control solutions. This approach aims to overcome the shortcomings of previous methods and provide a more efficient and comprehensive way to determine the best set of PD gains for our balance control problem.

Table 4.3: Parameter ranges for Random Search Approach in Isaac Gym

Parameters	**Value rnage**
Roll p gain	[0.0, 10.0]
Roo d gain	[0.0, 5.0]
Pitch p gain	[0.0, 10.0]
Pitch d gain	[0.0, 5.0]
Weight: Roll leg gain	[0.0, 0.5]
Weight: Pitch leg gain	[0.0, 0.5]
Weight: Roll left Arm gain	[-1.0, 1.0]
Weight: Pitch left arm gain	[-1.0, 1.0]
Weight: Roll right Arm gain	[-1.0, 1.0]
Weight: Pitch right arm gain	[-1.0, 1.0]
Low pass filter α	[0.0, 0.1]
Number of Environments	10,000
Number of Trial	50
Searching Time / Trial	5 min

Table 4.3 illustrates the specific parameters established for balance control within the Isaac Gym simulation environment, along with their respective ranges for implementing the random search algorithm. In total, we establish 11 parameters, such as PD gains of Roll, PD gains of Pitch, Weight values, and the low pass filter α to fine-tune balance control, with each parameter

falling within the value range outlined in Table 4.3. A total of 10,000 parallel environments are set up within Isaac Gym, enabling the efficient exploration of most values within the defined range to expedite the discovery of optimal values. Each random search trial is set to run for five minutes, with an overall design of 50 trials, to determine the values that best enable balance control over the longest duration on the balance board.

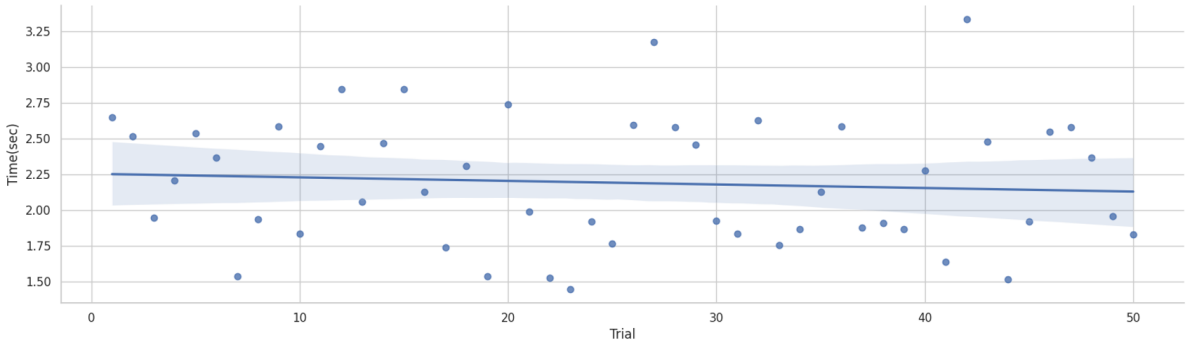


Figure 4.12: Distribution of Random Search algorithm result to find optimized values for 11 parameters

Figure 4.12 presents the results obtained through the experimental approach delineated above. With the implementation of the parameters defined in Table 4.3, the experiments conducted within 10,000 parallel simulation environments yielded a maximum balance control duration of 3.34 seconds when unique values were assigned to each parameter. The median line in Figure 4.12 represents the linear regression outcome of this distribution. Evidently, by utilizing this algorithm, the robot can achieve balance on the balance board for an approximate duration of 2.5 seconds, which, albeit short, is a meaningful step towards improved balancing capabilities. To validate the performance of the control system in a real-world experiment, the specific values of the parameters from Table 4.4 that corresponds to the maximum balance control duration of 3.34 seconds observed in the random search experiments must be applied into the control system of Robinion2S. An essential factor is that the simulating time was configured at 10 milliseconds, that is, 100 Hz. The frequency is purposely determined to align with the PID control loop rate in the Robinion2S system, thereby ensuring a more accurate transference of the simulation results to the physical robot performance.

The parameter values listed in Table 4.4 show marked differences from the PD gain and low pass filter α values detailed in Table 4.2 that were previously determined through the trial-

Table 4.4: The parameter values for the Best result

Parameters	**Value**
Roll p gain	8.16745376586914
Roo d gain	3.2218499183654785
Pitch p gain	4.086885929107666
Pitch d gain	4.987691879272461
Weight: Roll leg gain	0.2953128218650818
Weight: Pitch leg gain	0.47979602217674255
Weight: Roll left Arm gain	0.9960763454437256
Weight: Pitch left arm gain	0.8760212659835815
Weight: Roll right Arm gain	-0.23003923892974854
Weight: Pitch right arm gain	-0.6996899247169495
Low pass filter α	0.09156200289726257
Best Steps (seconds)	*334 steps (3.34 seconds)*

and-error approach. To explore and understand the discrepancies, we set up a single simulation environment in Isaac Gym, and the environment was configured to use the specific parameter values found in Table 4.4.

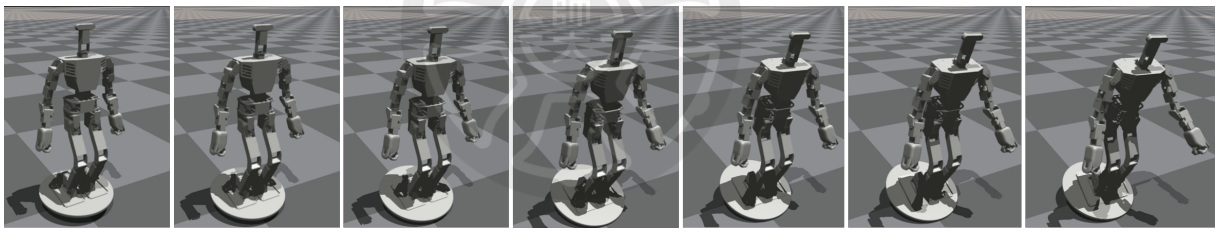


Figure 4.13: Simulation screenshots of balancing with the best steps

Contrary to initial expectations, the result of the random search algorithm for the robot in the simulation does not maintain balance control on the board through dynamic movements. Instead, as shown in Figure 4.13, the longest balance duration was achieved when the robot fell into a static state. The result suggests that it is not feasible for Robinion2S to balance on the highly dynamic environment of the balance board by relying solely on specific Roll and Pitch PD gains and low-pass filter α value. It requires a more sophisticated approach to maintaining balance, especially during dynamic movements.

4.3 Discussion

The chapter explored the application and testing of a PD-based balance control approach with Robinion2S in multiple experiments. The experimental settings involved two different balance boards, each possessing a unique set of dynamics and posing distinct challenges. In both balance board experiments, the trial-and-error tuning method was employed to determine the optimal settings for the PD gain and the low-pass filter's α value. However, the limitations of the trial-and-error method were apparent. It was not ideal for optimization since it could not account for all possible combinations of integers and floating-point numbers that could serve as PD gains. To overcome these limitations, a high-throughput random search algorithm was deployed within Nvidia's Isaac Gym simulation environment. It allowed for a more extensive exploration of parameter combinations, potentially enabling the discovery of more optimal PD gains for balance control. The results from the Isaac Gym experiments, however, presented a different perspective. While the simulation environment facilitated the discovery of a set of parameters that led to the longest balance duration in the simulation, it was observed that the longest balance duration was achieved when the robot fell into a static state. This result was counter to the dynamic balancing activity that was initially expected. Reflecting upon the results of this chapter, it was clear that specific parameters, even when optimized using the wide random search method, inevitably lead to failure in balance control when dealing with highly dynamic environments. Such methods expose the limitations of a simple PD-based control approach, which seems incapable of handling the complex interactions in maintaining balance. The result shows the need for more robust control methods to adapt to changing dynamics and ensure sustained balance.

Chapter 5. Balance Control with Reinforcement Learning

In this chapter, we extend beyond the limitations of traditional balance control methods, the Proportional-Derivative (PD) controller, to implement advanced approaches to balance control in the humanoid robot, Robinion2S. Chapter 4 discussed the challenges posed by PD controllers. Despite its notable performance, we Confronted inherent limitations, such as the incapability to handle dynamic states optimally and the necessity of fine-tuning the PD gains for different states, which can be impractical. To overcome these challenges, we turn to the progressive field of machine learning, specifically reinforcement learning, and launch an experiment on Proximal Policy Optimization (PPO). PPO is an advanced reinforcement learning algorithm appreciated for its balance between policy improvement and policy stability, making it an alternative to the PD controller. The implementation for solving the balance problem is based on Isaac Gym, a cutting-edge, high-fidelity physics simulation environment by NVIDIA designed specifically for robotics research. The advantage of Isaac Gym is its capacity to facilitate parallel simulations on a GPU, enabling efficient testing of various conditions and thereby speeding up the training process. The goal of the proposed experiment is the balance control of Robinion2S on a balance board, a highly dynamic environment that poses significant challenges. To solve the balance problem on the board, three separate control strategies with PPO are considered: one with PID & Inverse Kinematics (IK), one with IK poses, and the other with joint control. Through these experiments, we aim to develop the most effective approach for maintaining balance control on the dynamic balance board with Robinion2S. In this chapter, we research for a comprehensive analysis of these experimental results, intending to draw impactful conclusions and suggestions for future developments for balance tasks in humanoid robotics and control strategies.

5.1 Methodology

This section presents the methodology for training Robinion2S to balance on a board using Proximal Policy Optimization (PPO), dividing it into two subsections. The first subsection, Simulation Setup, details the hyperparameters employed and the different reset conditions configured during the training procedure. This subsection elaborates on the environment setup in the Isaac Gym simulator, where models are trained. The setting for hyper-parameters is crucial

as they dictate the model's training speed, the exploration-exploitation trade-off, and the overall performance. The reset conditions are diverse situations wherein Robinion2S starts in different states, such as varying different initial positions on the balance board. The hyper-parameters are essential as they drive the model's training process, and the reset methods ensure variability and robustness in the training steps. The second subsection, Task Description, elaborates on the specific details of the training task. It involves detailing the observation space and action space, which are the fundamental inputs and outputs of our reinforcement learning model. Furthermore, the reward function design that motivates the agent towards achieving the target of maintaining balance is explained. In the upcoming subsections, we explain deeper into each component of the methodology, detailing how to enable Robinion2S to learn the complicated dynamic task of balancing on a board.

5.1.1 Simulation Setup

This section clarifies the setup of our simulation environment, which includes the hyperparameters and the pre-established conditions leading to a reset. The core of our simulation setup revolves around the intricate task of training the robot, Robinion2S, to maintain its balance on a board. This setup, developed in the parallel structured simulation environment, Isaac Gym, not only accelerates the training process by running a large number of simulations concurrently but also creates a more generalized and robust policy by exploring a comprehensive range of possibilities. This simulation environment is optimized for a robust PPO training process and to ensure high-quality results for our humanoid robot, Robinion2S. The hyperparameters are configured to facilitate the efficient training and convergence of the PPO algorithm. Table 5.1 presents the specific hyperparameters utilized in the simulation setup. Each parameter is instrumental in defining and refining the training process of Robinion2S in the Isaac Gym simulation environment. The list of hyperparameters provides an optimized balance between exploration and exploitation, stability and flexibility, and long-term and short-term rewards, thus creating a conducive training environment. The selection of hyperparameters is effectively calibrated to the PPO algorithm's training process.

By creating 4096 parallel environments, we strengthen the possibility for the policy to explore different strategies and methods for balancing on the board. This considerable number of

Table 5.1: Hyper-parameters for training in the environment: PID&IK, IK, and Joint

Parameters	**Value**
Number of Environments	4096
Kullbak-Leibler divergence threshold	8×10^{-3}
PPO Mini-batch size	32768
Number of PPO Epochs	5
Adam Optimizer Learning Rate	5×10^{-4}
Discount factor γ	0.99
Generalized Advantage Estimation λ	0.99
PPO Clip threshold	0.2
Entropy coefficient	0.0
Hidden Units	[400, 200, 100]

simultaneous environments not only speeds up the training process but also enables a thorough exploration of the state and action spaces, contributing to a more robust and generalized policy. The Kullback-Leibler (KL) divergence threshold is set to maintain the proximity between the old and new policy during the policy update step. The KL divergence ensures that the policy does not change drastically during an update, aiding in maintaining stable learning. The mini-batch size for PPO is set to 32768. Mini-batches are subsets of the total experiences collected during the trajectories. They provide a balance between computational efficiency and gradient estimation accuracy, aiding in faster and more stable learning. The PPO training is conducted over five epochs. Each epoch represents a complete pass through the full batch of experiences, with multiple epochs helping in learning from the same data multiple times, ensuring better convergence of the policy. Adam Optimizer, with a learning rate of 5×10^{-4}, is selected for its efficient gradient descent optimization with dynamic learning rates. It is set to ensure that the policy parameters adjust at a pace that is neither too slow (leading to delayed convergence) nor too rapid (causing overshooting and potential instability). The discount factor, set at 0.99, determines the weight given to future rewards in the total return calculation. A higher value encourages Robinion2S to focus on long-term balancing strategies rather than immediate, shortsighted gains. The Generalized Advantage Estimation value is important in reducing the variance in the Advantage function estimation. By doing so, it helps to make the policy updates more reliable and stable. The PPO algorithm uses a clipping mechanism to restrict the policy update, preventing huge updates that could destabilize the learning process. The PPO clip threshold is set to 0.2. The

entropy coefficient encourages the exploration of the policy. However, it is set to 0 in this experiment as exploration was adequately provided by a large number of parallel environments and the stochastic nature of the policy. The policy network architecture is chosen to be a multi-layer perceptron with hidden layers of sizes 400, 200, and 100 units, respectively. This configuration provides sufficient complexity for the policy network to learn and represent diverse behaviors needed to balance on the board.

Specific reset conditions are designated to develop a meaningful learning environment and advance efficient training. The simulation environment is set to five different reset conditions and resets when the following conditions are met.

- ***Exceeding Maximum IMU Angle***
  - The environment resets if the Roll, Pitch, or Yaw of Robinion2S exceeds a predefined maximum IMU angle. This limit ensures that the robot maintains a relatively stable orientation.
- ***Continuous Ground Contact***
  - The environment resets if the board continuously touches the ground for over 2 seconds. This encourages the robot to learn strategies that prevent the board from touching the ground.
- ***Large Displacement of Torso***
  - The environment resets if the difference between the robot's current torso position and its initial torso position is more than 20cm. This encourages the robot to maintain its position on the board.
- ***Large Displacement of Foot***
  - The environment resets if the difference between the robot's current feet position and its initial feet position exceeds 3cm. This ensures that the robot learns to keep its foot position stable.
- ***Maximum Episode Length***

– Each episode is capped at a maximum length of 1000 steps, equivalent to 10 seconds. This time limit prevents overly long episodes and encourages the robot to learn efficient balancing techniques within this time frame.

The simulation setup ensures that the training environment is suitable for the complex task of balance control and that the robot learns a versatile and robust policy to maintain balance on the board.

5.1.2 Task Description

In the robotics field, RL has been particularly beneficial, leading to breakthroughs in complex tasks such as locomotion, manipulation, and balance. A particularly challenging application is controlling a humanoid robot to maintain balance on a balance board, which presents a highly dynamic, continuous, and unstable control problem. To tackle this problem, we employ the Proximal Policy Optimization (PPO) algorithm, a state-of-the-art RL technique known for its effectiveness and efficiency. This algorithm iteratively refines the policy used by the agent to select actions based on observed states, aiming to maximize the cumulative reward over time. The agent is the Robinion2S humanoid robot with advanced capabilities and 22 controlled joints. The complexity of the problem stems from the high dimensionality of the observation and action spaces, along with the unstable dynamics of the balance board. Below, we explain the three main constituents of the RL problem: the observations, actions, and rewards.

Observations are essential in reinforcement learning, providing the agent with valuable information about the current state of the environment and required data to evaluate and adapt the robotic behavior. Observations enable the agent to make informed decisions and learn from the consequences of the agent's actions. In the Robinion2S humanoid robot balancing on a balance board, the observation space is high-dimensional and consists of various information related to both the states of the robot and the balance board. In robotics simulation, building and training models in an environment that can closely mirror the complexities and uncertainties of the real world is imperative. To verify this idea, we develop the creation of two separate simulation modes: the Ideal mode and the Sim2Real mode. Table 5.2 shows the observation space for three different modes, which are PID&IK, IK, and Joint.

Table 5.2: Observations

Observation space	**PID & IK**		**IK**		**Joint**	
	Ideal	Sim2Real	Ideal	Sim2Real	Ideal	Sim2Real
Robinion2S Position	3	-	3	-	3	-
Robinion2S Orientation	3	3	3	3	3	3
Robinion2S Linear Velocity	3	-	3	-	3	-
Robinion2S Angular Velocity	3	3	3	3	3	3
Robinion2S Upward Projection	3	3	3	3	3	3
Position of Controlled Joints	22	22	22	22	22	22
Velocity of Controlled Joints	22	-	22	-	22	-
Board Position	3	-	3	-	3	-
Board Linear Velocity	3	-	3	-	3	-
Board Orientation(Roll & Pitch)	2	-	2	-	2	-
Board Upward Projection	1	-	1	-	1	-
Previous Actions	12	12	11	11	16	16
Number of Observations	80	43	79	42	84	47

The ideal mode is designed to serve as a theoretical benchmark, encompassing a wide array of observations with every bit of information about the robot and the environment available. It provides complete access to all possible observations. The ideal mode for the balance board includes measurements such as linear velocity, robot world position, joint velocity, board position, board linear velocity, orientation, and board upward projection that are not observable in the real world. The purpose of the Ideal mode is to act as a starting point, a theoretical model where we first verify our solutions. It allows the flexibility to comprehend the impact of all potential parameters on the robot's performance. However, the ideal mode operates under perfect conditions, which are hard to replicate in real-world scenarios. This is the starting point for the Sim2Real mode. Sim2Real mode, on the other hand, is designed to bridge the gap between simulations and real-world applications. The name, Sim2Real, itself suggests the mode's purpose - to allow the findings and solutions from the simulation environment to be effectively transferred to a real robot operating in the real world. The Sim2Real mode operates with a more constrained set of observations, reflective of the data that can realistically be obtained from the robot's sensors in an actual environment. The balancing control with Robinion2S on the balance board includes IMU data (Euler angles, Quaternions, Gyro-based angular velocity, and upward projection with Quaternions) and encoder data for controlled joint positions. In tandem,

the Ideal and Sim2Real modes play a crucial role in developing efficient control systems for robotics. Starting with the Ideal mode helps to verify the effectiveness of theoretical models, while the Sim2Real mode ensures the practical feasibility and reliability of these models when implemented in the real world.

Table 5.3 presents the different control modes for the Robinion2S balancing control on the board. Each mode has a unique set of actions that the robot can perform and uses a different control strategy. In the PID & Inverse Kinematics mode, the robot utilizes both the PD controller and inverse kinematics to achieve balance. The simulation timestep is 1/100s, meaning the

Table 5.3: Simulation setup for the environments

Environment(Mode)	**Simulation *dt***	**Actions**
PID & Inverse Kinematics	1/100 *s*	Roll p gain Roll d gain Pitch p gain Pitch d gain Weight for Leg Roll PID output Weight for Leg Pitch PID output Weight for Left Arm Roll PID output Weight for Left Arm Pitch PID output Weight for Right Arm Roll PID output Weight for Right Arm Pitch PID output Low pass filter alpha for Roll PID output Low pass filter alpha for Pitch PID output
Inverse Kinematics	1/100 *s*	Leg Position *x* Leg Position *y* Leg Orientation Roll Leg Orientation Pitch Left Arm Orientation Roll Left Arm Orientation Pitch Right Arm Orientation Roll Right Arm Orientation Pitch Low pass filter alpha for Leg Position Low pass filter alpha for Leg Orientation Low pass filter alpha for Arm Orientation
Joint	1/100 *s*	Position of 14 Joints (Ankle Roll & Pitch, Knee Pitch, Hip Roll & Pitch, Shoulder Roll & Pitch) Low pass filter alpha for Leg Jointa Low pass filter alpha for Arm Joints

control decisions are updated every hundredth of a second. The actions in this mode include adjusting the PID gains for roll and pitch, determining the weight given to the outputs of PID controllers for legs, left arm, and right arm, and setting the low pass filter α value for the roll and pitch PID output. In the Inverse Kinematics mode, the robot uses inverse kinematics to control its movements. It means that it calculates the joint parameters that provide a desired torso position. The actions include setting the x and y positions and roll and pitch orientations of the legs and roll and pitch orientations of the shoulders, as well as the low pass filter alpha values for leg position, leg orientation, and arm orientation. In the Joint mode, it controls the joints directly. Each of the 14 joints, ankle roll & pitch, knee pitch, hip roll & pitch, shoulder roll & pitch, can be adjusted to a specific position by current observations. The Joint mode also includes setting the low pass filter alpha values for leg and arm joints. Table 5.3 provides an overview of the different strategies employed to maintain balance in the robot. By understanding the different modes and their associated actions, we design and interpret the results of reinforcement learning algorithms more effectively.

The rewards signal to the agent how well its actions are contributing to the achievement of the goal. The rewards reflect the quality of an action taken in a given state, guiding the agent towards actions that help achieve its goal more effectively. The reward function is designed so that Robinion2S maintains balance on the board. It is achieved by penalizing deviations from the desired state. The reward function includes terms that discourage deviation of the torso position from the center of the board, encourage the robot to maintain an upright orientation, and reward the agent for keeping the balance board as flat as possible. Each component, the observations, actions, and rewards, interact cyclically, forming the basis of the RL problem. Over time, through the PPO algorithm, the agent learns to navigate this complex, high-dimensional space, refining its policy to optimally control the Robinion2S balance on the balance board. The reward weights for the different modes of operation are shown in Table 5.4. These weights are

Table 5.4: Reward Weight

Mode	PID & IK	IK	Joint
Reward weight	-5	-5	-20

used to calculate the reward for each mode, as shown in the following equations.

$$
\begin{aligned}
r_t^{sum} &= r_t^{R_{pos}} + r_t^{R_{roll}} + r_t^{Rpitch} + r_t^{Ryaw} + r_t^{Rlfootpos} + r_t^{Rrfootpos} + r_t^{Bangle} \\
r_t &= r_t^{sum} \ / \ Number\ of\ Rewards
\end{aligned}
\tag{5.1}
$$

The reward function is calculated based on various state parameters, as shown in Equations 5.1 to 5.5. The total reward r_t^{sum} is the sum of all the individual rewards, normalized by the number of rewards. The individual rewards are defined as follows.

$$
\begin{aligned}
D_{torso} &= \sqrt{{x_{pos}}^2 \ + \ {y_{pos}}^2} \\
r_t^{R_{pos}} &= \exp\left(R_{weight} \ * \ {D_{torso}}^2\right)
\end{aligned}
\tag{5.2}
$$

The $r^{R_{pos}}t$ reward is calculated based on the distance $Dtorso$ of the robot's torso from its initial position, as shown in Equation 5.2. The reward decreases exponentially as the robot moves away from the initial position.

$$
\begin{aligned}
r_t^{R_{roll}} &= \exp\left(R_{weight} \ * \ {\theta_{roll}}^2\right) \\
r_t^{R_{pitch}} &= \exp\left(R_{weight} \ * \ {\theta_{pitch}}^2\right) \\
r_t^{R_{yaw}} &= \exp\left(R_{weight} \ * \ {\theta_{yaw}}^2\right)
\end{aligned}
\tag{5.3}
$$

The $r^{R_{roll}}t$, $r^{Rpitch}t$, and r_t^{Ryaw} rewards are calculated based on the roll, pitch, and yaw orientations of the robot, as shown in Equation 5.3. The reward decreases exponentially as the robot's orientation deviates from the upright position.

$$
\begin{aligned}
r_t^{Rlfootpos} &= \exp\left(R_{weight} \ * \ {D_{lfoot_{pos}}}^2\right) \\
r_t^{Rrfootpos} &= \exp\left(R_{weight} \ * \ {D_{rfoot_{pos}}}^2\right)
\end{aligned}
\tag{5.4}
$$

The $r^{Rlfootpos}t$ and $r^{Rrfootpos}t$ rewards are calculated based on the distance of the left and right feet from their initial positions, as shown in Equation 5.4. The reward decreases exponentially as the feet move away from the initial position.

$$\theta_{board} = \sqrt{{\theta_{board_{roll}}}^2 + {\theta_{board_{pitch}}}^2}$$
$$r_t^{Bangle} = \exp\left(R_{weight} * {\theta_{board}}^2\right) \tag{5.5}$$

The r_t^{Bangle} reward is calculated based on the roll and pitch orientation of the balance board, as shown in Equation 5.5. The reward decreases exponentially as the board's orientation deviates from the horizontal position. The use of an exponential function in defining the reward structure of our reinforcement learning model serves a very important purpose. The exponential function has a unique characteristic: when the input value is zero, the output is one, and as the input deviates from zero, the output diminishes rapidly. For the balance board, the exponential function represents a highly suitable reward strategy. This approach focuses on minimizing the balance error, the difference between the desired balance state and the current state. When Robinion2S is perfectly balanced, the balance error is zero, yielding a reward of one - the maximum possible reward. As the robot deviates from the perfect balance, the balance error increases, and the reward decreases exponentially. This steep fall in reward for more significant errors encourages the learning model to maintain balance as accurately as possible, as even small deviations can lead to significant reductions in reward.

5.2 Evaluation

This section aims to evaluate the training results and performance of the balancing control on the balance board within the framework of the Isaac Gym environment, utilizing the PPO algorithm. The proposed balance board problem, a cutting-edge endeavor in the field of robotics and artificial intelligence, necessitates the harmonious interaction of physics, control systems, and machine learning. Given the system's complexity, an evaluation is critical to ensure the effectiveness and readiness for real-world deployment. Training and performance analysis, as well as performance testing, are explained for evaluating the balance board with Robinion2S. The core of the Balance Board project is developing a control policy that allows a robot to maintain its balance on a board in both simulated and real-world environments. It necessitates the execution of numerous training iterations, optimization processes, and a multitude of trials aimed at discovering the most qualified observations and control parameters. In this pursuit, we conduct an impressive 115 training, as shown in Figure 5.1, primarily focusing on identifying

applicable observations for the Ideal and Sim2Real modes.

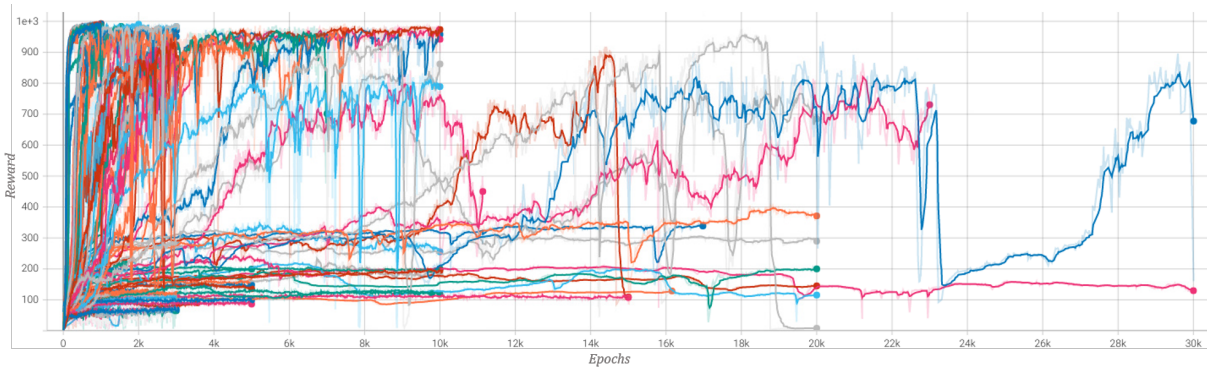


Figure 5.1: Result of training to find the optimal observation and Reward function of Ideal mode and Sim2Real mode in a total of 115 times

Although most training iterations are concluded within 10k epochs, in 11 instances, the process is extended to 20k epochs, and in 2 cases, it is trained until 30k epochs. Despite the extended training duration, the result does not guarantee satisfactory results, underlining the demanding nature of this project. However, they significantly contribute to the objective of deciding observation and reward functions that can realistically be implemented in Robinion2S, thus facilitating Sim2Real capabilities. After carefully analyzing and refining various observations, we set the optimal observations and reward function. We train the systems with the functions, which contain different combinations of Ideal and Sim2Real modes in PID & IK, IK, and Joint modes. To ensure the robustness of the results, we set different seeds for the training of each mode, resulting in three separate training seeds for each mode. The training results are illustrated in the graph shown in Figure 5.2. The graph suggests that, with the exception of PID

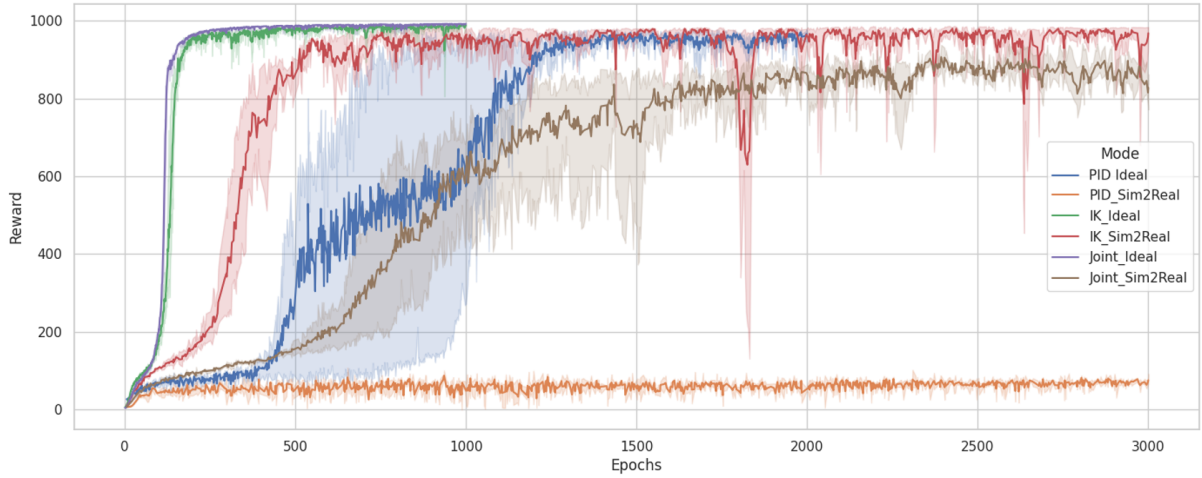


Figure 5.2: The optimized result of different combinations of Ideal and Sim2Real modes in PID & IK, IK, and Joint modes

& IK with Sim2Real mode, the models provide adequate results across the board.

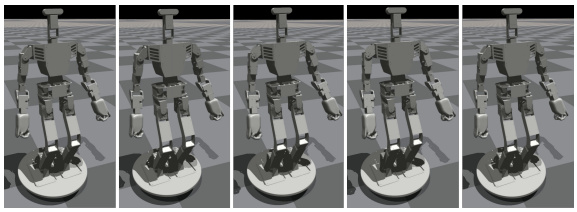
Table 5.5: Average value of reward and steps obtained from 30 trials of each mode

Mode	Reward	Steps
PID & Inverse Kinematics Ideal	972.09	1000
PID & Inverse Kinematics Sim2Real	53.83	58
Inverse Kinematics Ideal	987.12	1000
Inverse Kinematics Sim2Real	981.33	1000
Joint Ideal	989.00	1000
Joint Sim2Real	957.71	1000

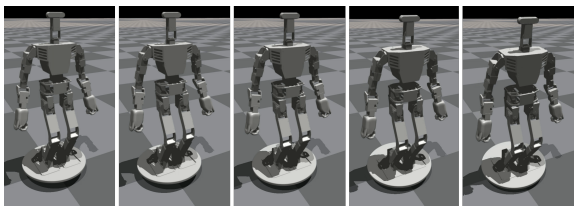
Table 5.5 indicates the evaluation results for each mode tested. It comprises the average reward and steps obtained from 30 trials of each mode. The result reveals that the PID & IK Ideal mode performs exceptionally well, achieving an average reward of 972.09 and consistently reaching the maximum episode length of 1000 steps. It indicates that the PID controller and inverse kinematics, when used in conjunction, generate an effective policy that enables the agent in the simulation to maintain balance on the balance board for a prolonged period of time. The result, however, is slightly different when it comes to the PID & IK Sim2Real mode. This mode attains an average reward of 53.83 and an average episode length of only 58 steps. This significant drop in performance in the Sim2Real mode compared to the Ideal mode suggests that the transition from simulation to the real world presents unique challenges that the PID & IK model struggles to overcome. In contrast, the Inverse Kinematics and Joint modes perform strongly in both the Ideal and Sim2Real environments, achieving high average rewards and consistently reaching the maximum episode length. This performance underscores the robustness and adaptability of these models, regardless of the environment in which they are deployed.

The comprehensive assessment of our reinforcement learning methodology is visually presented in Figure 5.3, which captures the simulation screenshots of balancing in six different modes. These dynamic demonstrations allow for the direct comparison of the different modes and their respective performances, emphasizing the effectiveness of the proposed approach.

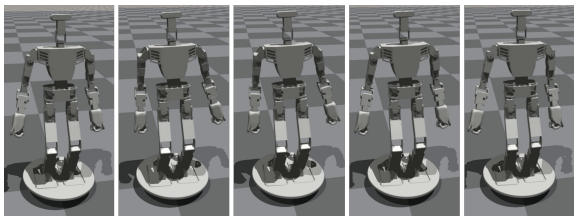
Ideal mode, a method in which we assume perfect information about the system, shows exemplary performance across all three modes, which are PID & IK, IK, and Joint (see Figures



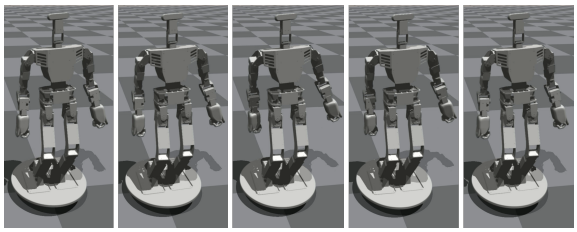
(a) PID & IK Ideal Mode



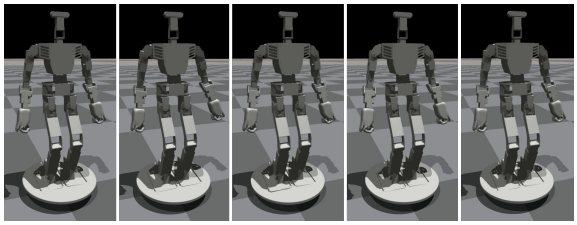
(b) PID & IK Sim2Real Mode



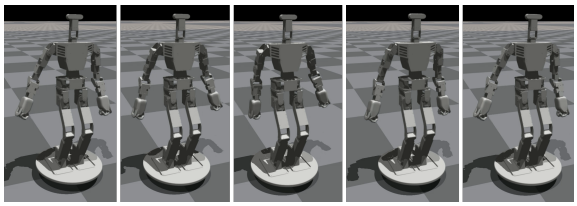
(c) IK Ideal Mode



(d) IK Sim2Real Mode



(e) Joint Ideal Mode



(f) Joint Sim2Real Mode

Figure 5.3: Simulation screenshots of balancing in six different modes

5.3a, 5.3c, 5.3e). The robots in these modes maintain balance without significant movement of their limbs, reflecting the efficiency of the control approach in Ideal modes. On the contrary, the performance in the Sim2Real modes, which aim to simulate real-world conditions, is more varied. The IK Sim2Real mode displays dynamic movements, with rapid arm movements and synchronized changes in the legs (see Figure 5.3d). This intricate interaction between the body parts exhibits the model's adaptability to maintain balance under challenging conditions. In the Joint Sim2Real mode, the robot achieves balance with only minor movements of the legs and more natural arm movements (see Figure 5.3f). This minimalistic approach mirrors human behavior, suggesting that the Joint Sim2Real mode may provide the most anthropomorphic movements when deployed on a physical robot. The results for the PID & IK Sim2Real mode in Figure 5.3b, however, are a remarkable revelation. Instead of dynamically adjusting its posture, the robot maintains its initial position, resembling a state of static equilibrium. Intriguingly, this result is quite similar to the optimal result achieved through the random search algorithm presented in Chapter 4 (see Figure 4.13).

5.3 Discussion

In conclusion, the exploration of reinforcement learning as a method for achieving balance control in Robinion2S presented in this chapter has provided valuable insights. Through experimentation and evaluation across varying modes—PID & IK, IK only, and Joint control—we observed the power and efficiency of reinforcement learning in tackling this intricate control problem. A fascinating observation was the striking resemblance in results between the reinforcement learning approach and the random search algorithm, especially in the PID & IK Sim2Real mode. Despite this mode delivering the least satisfactory results in terms of achieving balance, the fact that the optimal solution in both experiments was the robot maintaining its initial position. Implementing and testing proposed solutions in the Ideal and Sim2Real modes brought to emphasize the importance of accurate observation and the design of effective reward functions. Importantly, the trained models from IK Sim2Real and Joint Sim2Real modes were successfully deployed in the real robot, Robinion2S, demonstrating the feasibility of the Sim2Real approach. In this chapter, reinforcement learning for balance control with Robinion2S, while challenging, offered a wealth of knowledge and direction for future research. The balancing control on the balance board, through reinforcement learning, proposed potential in not only solving complex control problems but also unexpected strategies and solutions.

Chapter 6. Sim2Real

In chapter 5, we researched the simulation environments of reinforcement learning for balance control in robots. Through the experiments and evaluation, we gained valuable results on the efficiency of this approach in solving the complicated challenge of balancing control. Building upon the details and results from Chapter 5, we evaluate the application of the trained models to the real-world system of the Robinion2S robot in this chapter. This chapter focuses on the Sim2Real framework, a step toward bridging the gap between simulated environments and the physical world. The Sim2Real approach is available to bring the benefits of simulated training to real-world robotic systems. Specifically, we apply the IK Sim2Real and Joint Sim2Real models, which we trained and fine-tuned with possible observations in the real world in the previous chapter, to our physical robot, the Robinion2S. By implementing the Sim2Real models on the Robinion2S, we aim to validate the effectiveness and practical applicability of the reinforcement learning-based balance control solutions in the highly dynamic balance board. Applying simulated training models to physical platforms plays an important role in robotics research as it allows evaluation of the performance of learned policies in real-world environments. By applying IK Sim2Real and Joint Sim2Real models on the Robinion2S, we prove the ability of the trained models to adapt and maintain balance in a physical environment. Furthermore, we aim to show the potential of the Sim2Real approach in achieving a seamless transfer of control policies from simulation to real environment. The deployment and evaluation of our trained models on the Robinion2S not only validate the effectiveness of the reinforcement learning approach but also demonstrates the applicability of various tasks for Sim2Real applications in various robotics domains. The following sections of this chapter explain the methodology of the implementation process, the challenges encountered during the Sim2Real transfer, and the evaluation of Sim2Real on the Robinion2S robot.

6.1 Methodology

This chapter aims to apply the Sim2Real approach to the humanoid robot platform, Robinion2S, by employing the trained IK Sim2Real and Joint Sim2Real models obtained in Chapter 5. It enables investigating whether the trained models can be effectively executed on the physical

robot, validating the transferability of the control strategies from simulation to the real world. As discussed in Chapter 2, Robinion2S consists of two controllers: the main controller and the sub-controller. The block diagram depicted in Figure 6.1 illustrates the configuration of the sub-controller and main controller for the Sim2Real implementation.

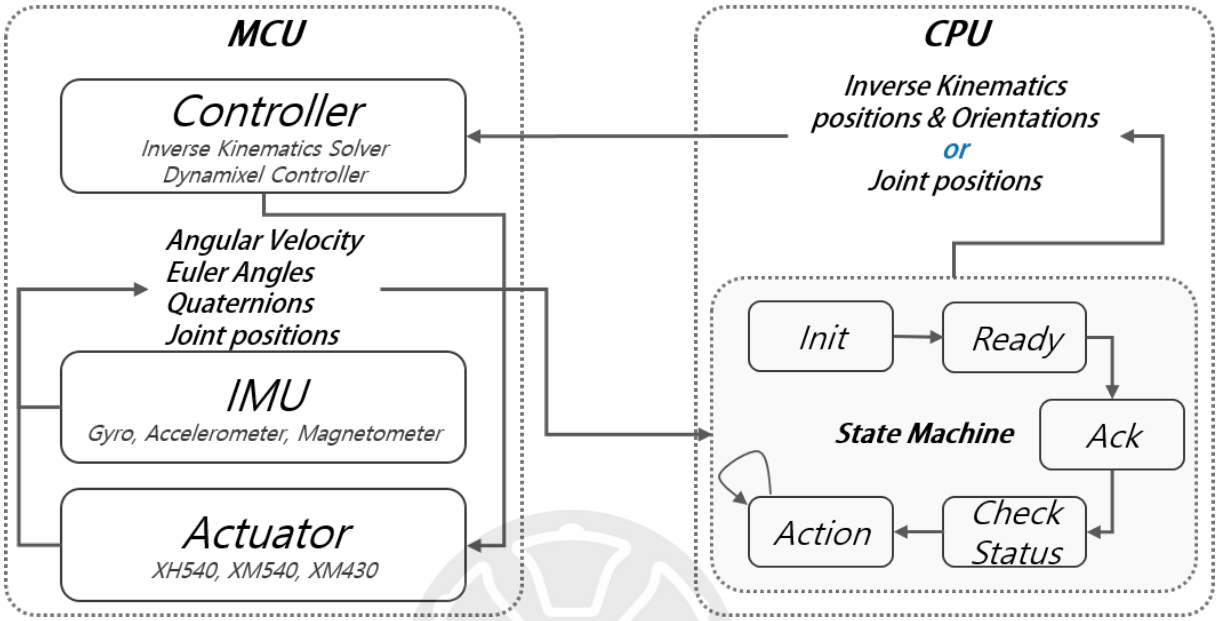


Figure 6.1: Block diagram of the sub-controller and main controller for the Sim2Real

The main and sub-controllers operate collaboratively to achieve balance control and execute complex movements on the balance board. The sub-controller is vital in gathering sensor data and performing low-level computations. It receives inputs from the 9-axis IMU sensor in the sub-controller, which provides low-level data from the gyroscope, accelerometer, and magnetometer. Using IMU sensor data, the sub-controller calculates Euler angles and quaternions to represent the robot's orientation in space. Additionally, the sub-controller receives the encoder value from the smart actuators installed on Robinion2S to recognize the position values of each joint of the humanoid robot. Building upon the observations made in Chapter 5, we select specific measurements as inputs for the Sim2Real implementation. The inputs include Euler angles, quaternions, angular velocities, upward projections, joint positions, and previous actions. The sub-controller obtains these values from the IMU sensor and the encoder values of the smart actuators. By using the inputs from the sensors into the Sim2Real approach, the method aims to solve the gap between simulation and the real world, enabling the robot to make informed control decisions based on real sensor data from Robinion2S. The main controller is the central

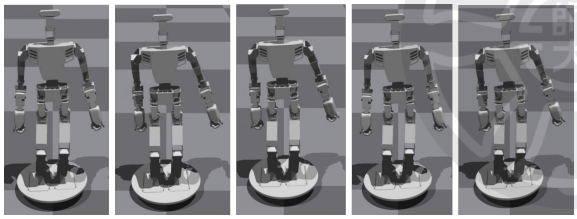
processing unit responsible for coordinating the control strategy and executing high-level decisions. It applies the sensor inputs received from the sub-controller to the trained IK Sim2Real and Joint Sim2Real models developed in Chapter 5. The trained models are optimized through the reinforcement learning algorithm to generate control signals that are the desired behaviors observed in the simulation. It employs a state machine to execute the Sim2Real models to the physical robot. The state machine consists of five states: Init, Ready, Ack, Check Status, and Action. In the Init state, the state machine performs initialization tasks, such as establishing serial communication, loading the trained models, and controlling the initial position of Robinion2S to prepare to balance control on the board. The Ready and Ack states verify the successful serial communication setting between the sub-controller and the main controller. These states ensure that the necessary data exchange between the two controllers is reliable and synchronized, guaranteeing precise communication during the execution of Sim2Real motions. The Check Status state is a preparation step before initiating action for Sim2Real. It checks the integrity of the input/output data between the sub-controller and the main controller, ensuring that all required information for Sim2Real execution is current. Once the data communication is successful, the state machine proceeds to the Action state. In the Action state, the main controller applies the sensor inputs received from the sub-controller to the trained IK Sim2Real and Joint Sim2Real models. The models generate corresponding control outputs, which are then transferred back to the sub-controller for execution. The sub-controller uses the outputs to control each joint of the Robinion2S robot, enabling the execution of the desired motions according to the Sim2Real approach. The proposed methodology seeks to validate the potential of the Sim2Real approach in enabling the transfer of control strategies from simulation to the real world. By successfully executing simulated motions on the physical robot, we can demonstrate the practical applicability and effectiveness of our approach in the real environment.

6.2 Evaluation

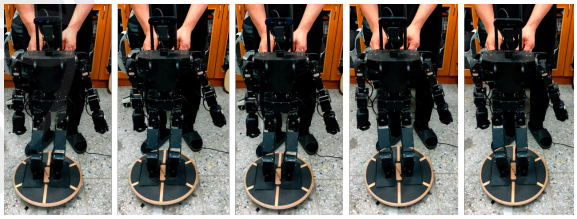
The evaluation of the Sim2Real approach presented in this chapter aims to assess the effectiveness and robustness of the developed balance control with Robinion2S on the board when applied to the real world. Through experiments and analysis, we aim to gain valuable results in the performance of the system and its potential for real-world applications. The evaluation

consists of two experiments designed with the Sim2Real approach of IK & Joint modes. The objective of the first experiment was to capture and analyze the motions of both the simulated and the real agent, Robinion2S. By comparing and contrasting each motion, it gives meaningful results to the system's ability to transfer simulated behaviors into real-world actions. The analysis provides a more in-depth understanding of the actuality and effectiveness of the Sim2Real approach in achieving balance control and performing complex tasks. The second experiment focuses on assessing the robustness of the Robinion2S robot by systematically adjusting the angles of pitch and roll for the initial pose of the balance board. The experimentation aims to evaluate the performance and stability when initializing the balance board at varying angles. By setting the various angles and observing the response of the robot's balance, it is possible to assess its ability to maintain balance under challenging circumstances.

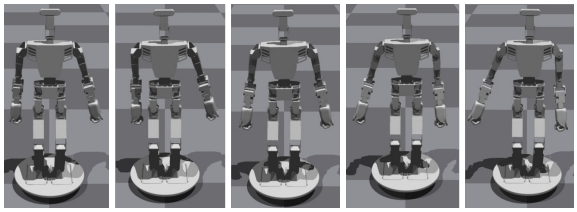
Figure 6.2 analyzes the actions observed when applying the trained IK and Joint modes model in the simulation environment and compares simulated agents to the real robot, Robinion2S.



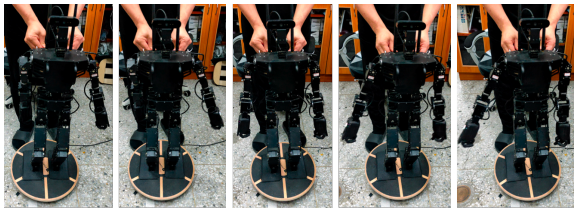
(a) IK Mode in Simulation



(b) IK Mode int Real world



(c) Joint Mode in Simulation



(d) Joint Mode in Real world

Figure 6.2: Transfer trained IK & Joint Modes policy from simulation to real world

This experiment aims to analyze the potential of the Sim2Real approach, specifically in the balance control on a balance board. However, it is observed that Robinion2S, the real robot, could not achieve the same level of balance as shown in the simulation. The primary reason for this discrepancy is that the experiment focused on researching the feasibility of Sim2Real and did not include domain randomization during the training process in the simulation environ-

ment. The absence of domain randomization can affect the robot's ability to adapt to real-world conditions, leading to variations between simulation and reality.

In the IK mode, balance control on the balance board is performed by controlling both the position and orientation of the shoulders and legs. Conversely, in the Joint mode, where there is minimal leg movement in the simulation environment, the experiment produces the same results even when the legs are fixed in their initial positions. This result shows the importance of arm movement in achieving balance for the humanoid robot, distinguishing it from the whole-body balance control employed by humans when balancing on the board. Despite the challenges encountered and the differences between simulation and reality, it is noteworthy that the simulated and real environments demonstrated similar actions in the same input state. This result indicates the potential applicability of the developed Sim2Real application. The results of this first experiment present the need for further exploration and refinement in the domain of Sim2Real balance control.

The second experiment aims to assess the performance of the IK and Joint modes in the simulation environment under different initial board orientations. A total of 30 experiments for each degree are conducted, with each experiment involving an increase of 1 degree in the pitch and roll directions for the initial board orientation. It is important to note that in the actual robot, it is not possible to precisely set the initial angle of the board, making it challenging to provide precise experimental results. However, by adjusting the initial values of the board for the real robot, it is possible to observe initial actions that showed similarities to movements observed in the simulation. This suggests potential possibilities for applying the simulated control strategies to various real-world tasks. To ensure consistency, every experiment followed the reset condition outlined in Chapter 5, which defined the standards for completing each experiment. The evaluation of the IK and Joint modes is based on two results: Rewards and Steps. The results provide the robustness of the balance control algorithms in maintaining stability on the balance board. The average values of Rewards and Steps for the experiments are presented in Table 6.1 and 6.2 for the IK mode and Joint mode.

Table 6.1 presents the results obtained in the IK Mode simulation environment. It indicates that the pitch axis was capable of maintaining balance for initial board angles up to 11 degrees.

Table 6.1: Results regarding Reward & Steps in Different Initial Angles of IK

Direction	Board Angle(°)	Reward	Steps
Pitch	1	981.43	1000
	2	981.47	1000
	3	981.69	1000
	4	981.63	1000
	5	901.71	920.37
	6	872.81	890.63
	7	710.08	725.73
	8	648.01	662.47
	9	536.58	549.73
	10	407.48	418.03
	11	234.15	242.33
Roll	1	981.03	1000
	2	980.61	1000
	3	980.16	1000
	4	979.66	1000
	5	978.78	1000
	6	699.65	717.83
	7	129.99	136.27

Within the 1 to 4 degrees range, the balance was achieved perfectly regardless of the initial angle. However, as the angle increases beyond this range, it becomes increasingly challenging to maintain balance. Similarly, when analyzing the initial pose of the board in the roll axis, the IK Mode perfectly demonstrates balance control up to 5 degrees, regardless of the initial angle. However, beyond 7 degrees, it becomes that maintaining balance is not possible.

Table 6.2 shows the Joint mode simulation environment results. In this case, the pitch axis was capable of balancing the initial board angles up to 10 degrees. Similar to the IK Mode, balance control was achieved within the 1 to 5 degrees range, regardless of the initial angle. However, as the angle increased beyond this range, the difficulty of maintaining balance rapidly intensified. When considering the roll axis of the board in the Joint Mode, balance control was achieved up to 2 degrees, irrespective of the initial angle. The detailed information provided in these tables offers results in the performance of the IK and Joint Modes under varying initial board orientations.

Figure 6.3 provides a visual representation of the experimental results, which are supported

Table 6.2: Results regarding Reward & Steps in Different Initial Angles of Joint Mode

Direction	Board Angle(°)	Reward	Steps
Pitch	1	956.40	1000
	2	958.98	1000
	3	957.50	1000
	4	958.04	1000
	5	957.28	1000
	6	926.82	970.70
	7	596.53	623.47
	8	531.69	555.53
	9	300.54	315.90
	10	160.87	171.67
Roll	1	952.58	1000
	2	947.62	1000
	3	795.47	846.00
	4	583.39	626.13
	5	285.58	311.03
	6	135.94	1.47
	7	77.93	88.20

by the data presented in table 6.1 and table 6.2. This graph offers an overview of the performance of the IK and Joint modes across different board angles, enabling comparison and analysis of the balance control capabilities of the initial orientation of the balance board. Analyzing the results shows that the ability to maintain balance diminishes as the initial angles deviate further from the trained pose. The results contribute to apprehending the robustness of the balance control algorithms. In addition, as demonstrated in the results of both modes, it is evident that the angle

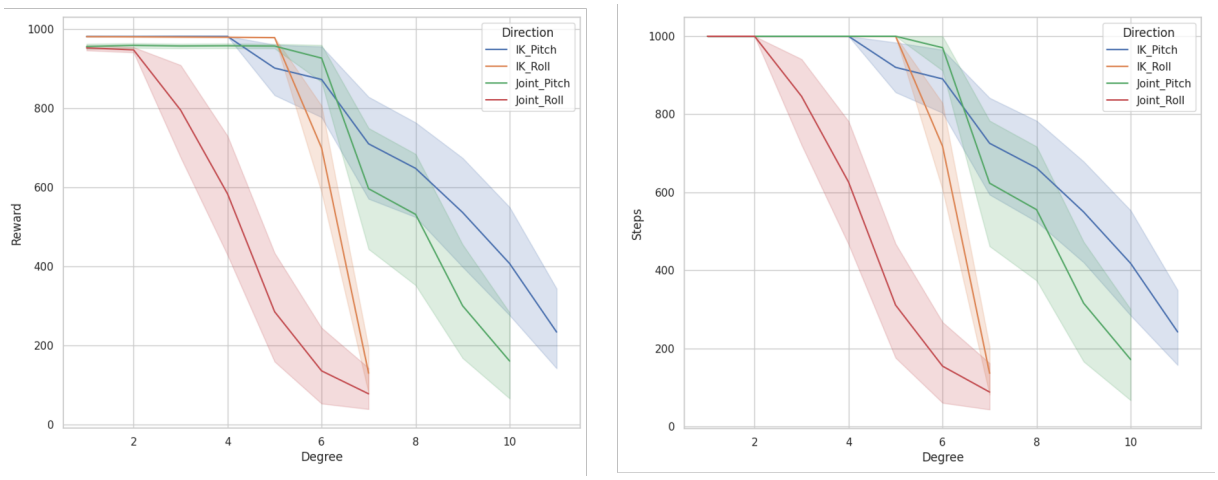
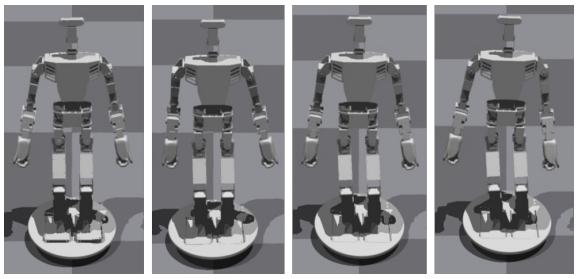
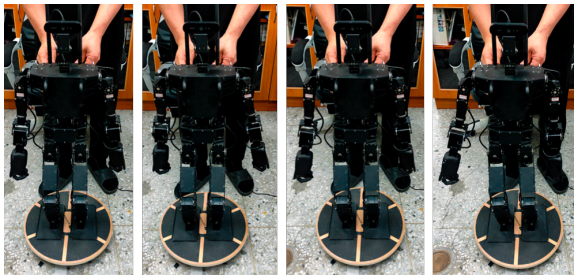


Figure 6.3: The results of the IK and Joint Modes under different initial board orientations

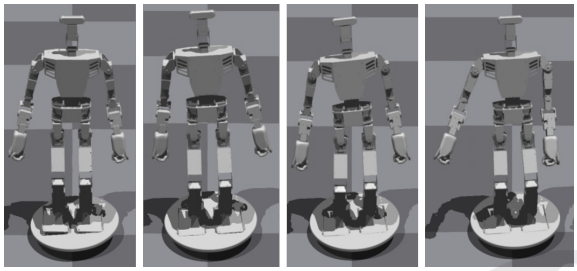
of the roll axis of the initial board orientation is important for balance control. The experiments showed that changes in the roll axis angle of the board impact the balance more than changes in the pitch axis.



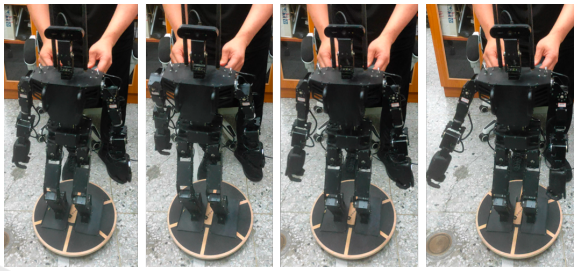
(a) Pitch axis of the IK mode in simulation: 11°



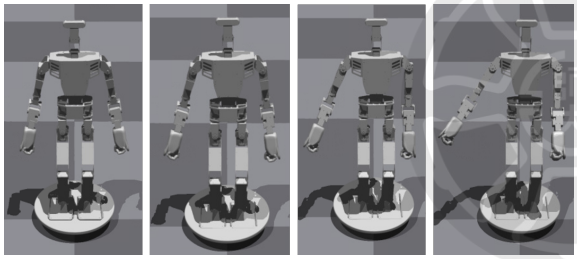
(b) Pitch axis of the IK mode in real world



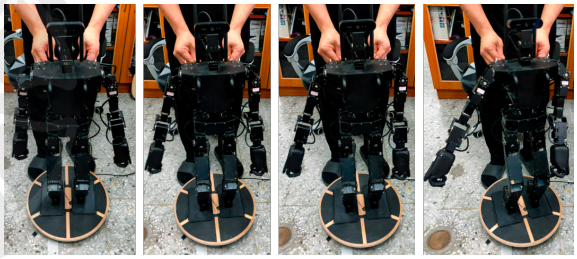
(c) Roll axis of the IK mode in simulation: 7°



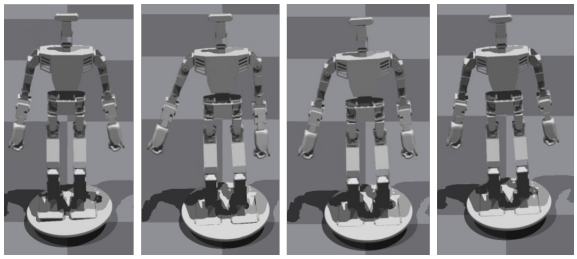
(d) Roll axis of the IK mode in real world



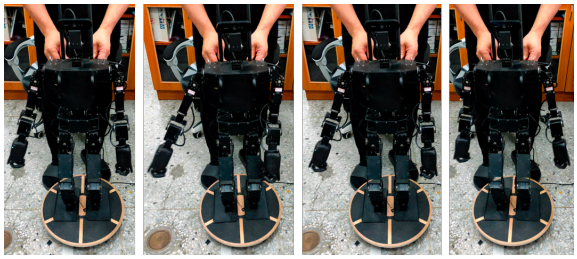
(e) Pitch axis of the Joint mode in simulation: 10°



(f) Pitch axis of the Joint mode in real world



(g) Roll axis of the Joint mode in simulation: 7°



(h) Roll axis of the Joint mode in real world

Figure 6.4: Initial behavior of Robinion2S for different board initial angles of roll and pitch axis in simulation and real world

Figure 6.4 shows a visual comparison of the motion of Robinion2S about the results obtained from different initial angles of the balance board in simulation and the real environment. The experiments are conducted by incrementing the angles by 1 degree in both the roll and pitch axes. Due to the similarity of the initial movements at different angles within the same axis in the simulation, a representative picture is selected to represent the results of each set of experi-

ments. In the real environment, the experiments are repeated to confirm similar movements due to the inability to precisely set the exact initial angle of the board and the robot program starting from the same position. The comparison between the simulation results and the real robot's movements indicates a high similarity, proving the effectiveness of the Sim2Real approach. It is important to note that while there are limitations due to human intervention, the similarity between the simulated and real-world movements further confirms the potential and practical applicability of the Sim2Real.

6.3 Discussion

This chapter analyzed the Sim2Real approach for balance control with the humanoid robot platform, Robinion2S. By implementing the trained IK Sim2Real and Joint Sim2Real models on the physical robot, we aimed to validate the effectiveness and practical applicability of the reinforcement learning-based balance control solutions in a real-world, highly dynamic environment. Through experiments and evaluation, we gained results on the performance and limitations of the Sim2Real approach. The methodology involved the collaborative system of the main and sub-controllers in enabling balance control and complex movements on the balance board, with the sub-controller collecting sensor data and the main controller utilizing trained Sim2Real models.

The evaluation of the Sim2Real approach included two experiments. The first experiment focused on comparing the motions of the simulated agent and the real robot to analyze the transferability of control strategies from simulation to the real world. The results showed similarities in the actions between simulation and reality, indicating the potential applicability of the Sim2Real approach. However, challenges were encountered due to the absence of domain randomization during training, leading to discrepancies between simulation and reality. The second experiment aimed to assess the robustness of the balance control algorithms by adjusting the initial angles of the balance board. The results demonstrated that the IK and Joint modes could maintain balance up to certain angles in both the pitch and roll axes. However, as the initial angles deviated further from the trained pose, the ability to maintain balance diminished. Changes in the roll axis angle had a greater impact on balance control compared to changes in the pitch axis.

In conclusion, while the Sim2Real approach did not achieve ideal results, it represents a significant step forward in demonstrating the potential for real-world applications. The implementation of the trained models on the physical robot, Robinion2S, validated the effectiveness of the reinforcement learning approach and showcased the potential of the Sim2Real framework in achieving a seamless transfer of control policies from simulation to the real environment.

Chapter 7. Conclusion and Future work

This thesis analyzed balance control in a highly dynamic environment with its own implemented humanoid robot platform, Robinion2S. Starting with an analysis of humanoid robot platforms in Chapter 2, the research was based on an understanding of the systems and components contributing to the capabilities of presented humanoid robot platforms. It provided a detailed overview of the humanoid robot platforms, focusing on the Robinion series. An exploration of the mechatronic system, walking gait algorithm, and the perception system of the platform showed the robustness and cost-effectiveness of the Robinion series. Experiments conducted on artificial turf validated the capability to maintain stability with different step sizes, validating the efficiency of the walking gait algorithm in challenging environments. The effectiveness of the deep learning-based perception system was also demonstrated, proving the advantages of advanced models like our own customed model based on Tiny YOLO-v3 and MobileNet over traditional image processing techniques. With an advanced humanoid robot platform named Robinio2S, we implemented balancing control in the highly dynamic environment, which was the balance board.

To solve the balance task, it presented a PD-based balance control approach with the Robinion2S robot on different balance boards in chapter 4. The experiments explained the methods of achieving balance control with traditional methods for the balance board. However, as the study evolved, the balance control limitations of traditional PD-based methods were identified in Chapter 4. While the PD-based balance control approach allowed simplistic handling of the system, the inability to adapt and respond to dynamic environmental changes emerged as a fundamental weakness. In addition, while the random search algorithm in the Isaac Gym environment proved valuable in determining optimal PD gains, the results indicated the limitations of PD-based control and presented the necessity for more advanced, adaptive techniques.

In response to the result of chapter 4, Chapter 5 introduced the innovative application of reinforcement learning to balance control. Various control modes were evaluated, demonstrating the potential of reinforcement learning in solving the dynamic balance control problem. The result of the PID & IK mode for Sim2Real were less satisfactory, yet the results from the

IK Sim2Real and Joint Sim2Real modes presented valuable solutions for the real robot. Reinforcement learning techniques demonstrated their robustness by adapting to a highly intricate control problem, thus offering an alternative strategy for balance control. The chapter also underscored the importance of designing effective reward functions and validating the Sim2Real approach. Furthering the advanced exploration of reinforcement learning, chapter 6 presented the Sim2Real approach by scrutinizing the Sim2Real method for balance control. The deployment of trained models on the real-world robot Robinion2S validated the viability of reinforcement learning-based balance control in the real world. It provided a practical examination of the effectiveness of reinforcement learning-based balance control strategies within a dynamic environment. Although challenges were encountered in the form of discrepancies between simulation and reality, the experiments demonstrated the potential of the Sim2Real framework in achieving a seamless transfer of control policies from simulation to reality. The research within this thesis offered an important understanding of the challenging subject of balance control for the humanoid robot in the highly dynamic environment, which is the balance board. It has highlighted the limitations of the traditional control method and proposed a shift towards a more advanced, robust, and versatile reinforcement learning technique. The performed work represents a considerable contribution to the robotics field, though it is clear that additional research and development are necessary to realize the potential of these new approaches.

This research shows several avenues for further research and development. The application of advanced artificial intelligence techniques, particularly reinforcement learning, shows potential. A key solution lies in enhancing the Sim2Real transfer methodology. Although the results showed potential to solve the balancing problem, the challenge of discrepancies between simulated and real environments persisted. Future work focuses on training domain randomization for deploying the real robot in the simulator, allowing the reinforcement learning algorithm to generalize better across a range of environmental parameters. Additionally, while this research used the balance board as the research environment, future research could extend to more complex and dynamic tasks. The interactions with different types of robots, different terrains, and unpredictable disturbances are available to provide better robustness for the control strategy. Finally, as the ultimate goal of humanoid robots is to function in human environments, developing more advanced control and perception systems for better environment understanding and

interaction will be vital. This research is one of the significant steps forward toward achieving dynamic balance control for humanoid robots. By building on the work presented in this thesis, it is hoped that the dream of highly autonomous, balanced, and interactive humanoid robots can soon become a reality.

References

[1] S. Saeedvand, H. Mandala, and J. Baltes, "Hierarchical deep reinforcement learning to drag heavy objects by adult-sized humanoid robot," *Applied Soft Computing*, vol. 110, p. 107601, 2021.

[2] D. Rodriguez and S. Behnke, "Deepwalk: Omnidirectional bipedal gait by deep reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, (Xi' an, China), pp. 3033–3039, IEEE, June 2021.

[3] M. F. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, "Continuous humanoid locomotion over uneven terrain using stereo fusion," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, (Seoul, South Korea), pp. 881–888, IEEE, November 2015.

[4] H.-M. Joe and J.-H. Oh, "A robust balance-control framework for the terrain-blind bipedal walking of a humanoid robot on unknown and uneven terrain," *Sensors*, vol. 19, no. 19, p. 4194, 2019.

[5] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2, (Washington DC, USA), pp. 1398–1403, IEEE, May 2002.

[6] H.-M. Joe and J.-H. Oh, "A robust balance-control framework for the terrain-blind bipedal walking of a humanoid robot on unknown and uneven terrain," *Sensors*, vol. 19, no. 19, p. 4194, 2019.

[7] C. J. Iverach-Brereton, "Rocking the bongo board: Humanoid robotic balancing on dynamic terrain," Master's thesis, University of Manitoba, 2015.

[8] L. Liu and J. Hodgins, "Learning to schedule control fragments for physics-based characters using deep q-learning," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 3, pp. 1–14, 2017.

[9] J.-T. Song, G. Christmann, J. Jeong, and J. Baltes, *Reinforcement Learning and Action Space Shaping for a Humanoid Agent in a Highly Dynamic Environment*, pp. 29–42. Springer International Publishing, 2023.

[10] I. Gori, U. Pattacini, F. Nori, G. Metta, and G. Sandini, "Dforc: A real-time method for reaching, tracking and obstacle avoidance in humanoid robots," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, (Osaka, Japan), pp. 544–551, IEEE, November 2012.

[11] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model predictive control of humanoid robots," in *2013 13th IEEE-RAS International conference on humanoid robots (Humanoids)*, (Atlanta, Georgia, USA), pp. 292–299, IEEE, October 2013.

[12] A. K. Kashyap and D. R. Parhi, "Particle swarm optimization aided pid gait controller design for a humanoid robot," *ISA transactions*, vol. 114, pp. 306–330, 2021.

[13] B. Henze, C. Ott, and M. A. Roa, "Posture and balance control for humanoid robots in multi-contact scenarios based on model predictive control," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Chicago, Illinois, USA), pp. 3253–3258, IEEE, September 2014.

[14] A. Hosseinmemar, J. Anderson, J. Baltes, M. C. Lau, and Z. Wang, "Push recovery and active balancing for inexpensive humanoid robots using rl and drl," in *Trends in Artificial Intelligence Theory and Applications. Artificial Intelligence Practices: 33rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2020*, (Kitakyushu, Japan), pp. 63–74, Springer, September 2020.

[15] D. A. Rodriguez Vargas, *Learning Grasping and Walking Motion Generation for Humanoid Robots*. PhD thesis, Universitäts-und Landesbibliothek Bonn, 2021.

[16] J. Jeong, J. Yang, G. H. G. Christmann, and J. Baltes, "Lightweight mechatronic system for humanoid robot," *The Knowledge Engineering Review*, vol. 38, p. e5, 2023.

[17] J. Jeong, J. Yang, and J. Baltes, “Robot magic show as testbed for humanoid robot interaction,” *Entertainment Computing*, vol. 40, p. 100456, 2022.

[18] R. Gerndt, D. Seifert, J. H. Baltes, S. Sadeghnejad, and S. Behnke, “Humanoid robots in soccer: Robots versus humans in robocup 2050,” *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 147–154, 2015.

[19] J. Baltes, S. Sadeghnejad, D. Seifert, and S. Behnke, “Robocup humanoid league rule developments 2002–2014 and future perspectives,” in *Robot Soccer World Cup*, pp. 649–660, Springer, 2014.

[20] M. Paetzel and L. Hofer, “The robocup humanoid league on the road to 2050 [competitions],” *IEEE Robotics & Automation Magazine*, vol. 26, no. 4, pp. 14–16, 2019.

[21] J. Baltes, K.-Y. Tu, S. Sadeghnejad, and J. Anderson, “Hurocup: competition for multi-event humanoid robot athletes,” *The Knowledge Engineering Review*, vol. 32, 2017.

[22] H. Moon, Y. Sun, J. Baltes, and S. J. Kim, “The iros 2016 competitions [competitions],” *IEEE Robotics and Automation Magazine*, vol. 24, no. 1, pp. 20–29, 2017.

[23] J. Baltes, Y. Sun, and H. Moon, “2017 competitions: Magical, manipulating, mercurial robots [competitions],” *IEEE Robotics and Automation Magazine*, vol. 25, no. 2, pp. 8–15, 2018.

[24] J. Van Dingenen, “High performance dyneema fibres in composites,” *Materials & Design*, vol. 10, no. 2, pp. 101–104, 1989.

[25] K. Seshadri, B. Akin, J. Laudon, R. Narayanaswami, and A. Yazdanbakhsh, “An evaluation of edge tpu accelerators for convolutional neural networks,” in *2022 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 79–91, IEEE, 2022.

[26] Y. Sun and A. M. Kist, “Deep learning on edge tpus,” *arXiv preprint arXiv:2108.13732*, 2021.

[27] P. Adarsh, P. Rathi, and M. Kumar, “Yolo v3-tiny: Object detection and recognition using one stage improved model,” in *2020 6th international conference on advanced computing*

and communication systems (ICACCS), (Tamil Nadu, India), pp. 687–694, IEEE, March 2020.

[28] A. S. Aguiar, F. N. Dos Santos, A. J. M. De Sousa, P. M. Oliveira, and L. C. Santos, "Visual trunk detection using transfer learning and a deep learning-based coprocessor," *IEEE Access*, vol. 8, pp. 77308–77320, 2020.

[29] S. Junk, B. Klerch, and U. Hochberg, "Structural optimization in lightweight design for additive manufacturing," *Procedia CIRP*, vol. 84, pp. 277–282, 2019.

[30] G. Ficht, H. Farazi, D. Rodriguez, D. Pavlichenko, P. Allgeuer, A. Brandenburger, and S. Behnke, "Nimbro-op2x: affordable adult-sized 3d-printed open-source humanoid robot for research," *arXiv preprint arXiv:2010.09308*, 2020.

[31] J. Jeong, J. Yang, and J. Baltes, "Robot magic show: human–robot interaction," *The Knowledge Engineering Review*, vol. 35, 2020.

[32] P. Allgeuer, H. Farazi, G. Ficht, M. Schreiber, and S. Behnke, "The igus humanoid open platform," *KI-Künstliche Intelligenz*, vol. 30, no. 3, pp. 315–319, 2016.

[33] M. Bestmann, J. Güldenstein, F. Vahl, and J. Zhang, "Wolfgang-op: A robust humanoid robot platform for research and competitions," in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, (Munich, Germany), pp. 90–97, IEEE, July 2021.

[34] P. Allgeuer, M. Schwarz, J. Pastrana, S. Schueller, M. Missura, and S. Behnke, "A ros-based software framework for the nimbro-op humanoid open platform," *arXiv preprint arXiv:1809.11051*, 2018.

[35] J. Jeong, J. Yang, and J. Baltes, "Humanoid robot platform: Robinion-2," in *2020 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*, (Taipei, Taiwan), August 2020.

[36] A. M. Abate, *Mechanical design for robot locomotion*. PhD thesis, Oregon State University, 2018.

[37] H. Shin, T. Ishikawa, T. Kamioka, K. Hosoda, and T. Yoshiike, "Mechanistic properties of five-bar parallel mechanism for leg structure based on spring loaded inverted pendulum," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, (Toronto, Canada), pp. 320–327, IEEE, October 2019.

[38] S. Shigemi, A. Goswami, and P. Vadakkepat, "Asimo and humanoid robot research at honda," *Humanoid robotics: A reference*, pp. 55–90, 2018.

[39] K. Yokoi, F. Kanehiro, K. Kaneko, S. Kajita, K. Fujiwara, and H. Hirukawa, "Experimental study of humanoid robot hrp-1s," *The International Journal of Robotics Research*, vol. 23, no. 4-5, pp. 351–362, 2004.

[40] H. Hirukawa, F. Kanehiro, K. Kaneko, S. Kajita, K. Fujiwara, Y. Kawai, F. Tomita, S. Hirai, K. Tanie, T. Isozumi, *et al.*, "Humanoid robotics platforms developed in hrp," *Robotics and Autonomous Systems*, vol. 48, no. 4, pp. 165–175, 2004.

[41] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, *et al.*, "Talos: A new humanoid research platform targeted for industrial applications," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, (Birmingham, UK), pp. 689–695, IEEE, November 2017.

[42] M. Vukobratović and B. Borovac, "Zero-moment point—thirty five years of its life," *International journal of humanoid robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[43] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural networks*, vol. 21, no. 4, pp. 642–653, 2008.

[44] G. Ficht and S. Behnke, "Bipedal humanoid hardware design: A technology review," *Current Robotics Reports*, vol. 2, pp. 201–210, 2021.

[45] K. Khokar, P. Beeson, and R. Burridge, "Implementation of kdl inverse kinematics routine on the atlas humanoid robot," *Procedia Computer Science*, vol. 46, pp. 1441–1448, 2015.

[46] G. Tevatia and S. Schaal, "Inverse kinematics for humanoid robots," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automa-*

tion. Symposia Proceedings (Cat. No. 00CH37065), vol. 1, (San Francisco, CA, USA), pp. 294–299, IEEE, April 2000.

[47] K. Jolly, R. S. Kumar, and R. Vijayakumar, "A bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits," *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 23–33, 2009.

[48] T.-Y. Li, P.-F. Chen, and P.-Z. Huang, "Motion planning for humanoid walking in a layered environment," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 3, (Taipei, Taiwan), pp. 3421–3427, IEEE, September 2003.

[49] J. García and D. Shafie, "Teaching a humanoid robot to walk faster through safe reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 88, p. 103360, 2020.

[50] A. F. Muzio, M. R. Maximo, and T. Yoneyama, "Deep reinforcement learning for humanoid robot behaviors," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 1, p. 12, 2022.

[51] L. C. Melo and M. R. O. A. Máximo, "Learning humanoid robot running skills through proximal policy optimization," in *2019 Latin american robotics symposium (LARS), 2019 Brazilian symposium on robotics (SBR) and 2019 workshop on robotics in education (WRE)*, pp. 37–42, IEEE, 2019.

[52] K. Harada, S. Kajita, K. Kaneko, and H. Hirukawa, "An analytical method for real-time gait planning for humanoid robots," *International Journal of Humanoid Robotics*, vol. 3, no. 01, pp. 1–19, 2006.

[53] D. J. Bora, A. K. Gupta, and F. A. Khan, "Comparing the performance of l* a* b* and hsv color spaces with respect to color image segmentation," *arXiv preprint arXiv:1506.01472*, 2015.

[54] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055*, 2019.

[55] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[56] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[57] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "Deeplabcut: markerless pose estimation of user-defined body parts with deep learning," *Nature neuroscience*, vol. 21, no. 9, pp. 1281–1289, 2018.

[58] H. Yuen, J. Princen, J. Illingworth, and J. Kittler, "Comparative study of hough transform methods for circle finding," *Image and vision computing*, vol. 8, no. 1, pp. 71–77, 1990.

[59] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[60] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, (Honolulu, HI, USA), pp. 5918–5926, July 2017.

[61] X. Fan, M. Jiang, and H. Yan, "A deep learning based light-weight face mask detector with residual context attention and gaussian heatmap to fight against covid-19," *Ieee Access*, vol. 9, pp. 96964–96974, 2021.

[62] B. A. Robson, T. Bolch, S. MacDonell, D. Hölbling, P. Rastner, and N. Schaffer, "Automated detection of rock glaciers using deep learning and object-based image analysis," *Remote sensing of environment*, vol. 250, p. 112033, 2020.

[63] J. Redmon, "Darknet: Open source neural networks in c." http://pjreddie.com/darknet/, 2013–2016.

[64] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, (Santiago, Chile), pp. 1440–1448, December 2015.

[65] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, (Salt Lake City, UT, USA), pp. 4510–4520, June 2018.

[66] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," *Advances in neural information processing systems*, vol. 31, 2018.

[67] M. A. Johnson and M. H. Moradi, *PID control*. Springer, 2005.

[68] K. J. Åström and T. Hägglund, "The future of pid control," *Control engineering practice*, vol. 9, no. 11, pp. 1163–1175, 2001.

[69] K. H. Ang, G. Chong, and Y. Li, "Pid control system analysis, design, and technology," *IEEE transactions on control systems technology*, vol. 13, no. 4, pp. 559–576, 2005.

[70] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[71] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[72] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, *et al.*, "Sim2real in robotics and automation: Applications and challenges," *IEEE transactions on automation science and engineering*, vol. 18, no. 2, pp. 398–400, 2021.

[73] R. K. Mandava and P. R. Vundavilli, "An adaptive pid control algorithm for the two-legged robot walking on a slope," *Neural Computing and Applications*, vol. 32, pp. 3407–3421, 2020.

[74] S. Kalouche, D. Rollinson, and H. Choset, "Modularity for maximum mobility and manipulation: Control of a reconfigurable legged robot with series-elastic actuators," in *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–8, IEEE, 2015.

[75] S. Seok, A. Wang, D. Otten, and S. Kim, "Actuator design for high force proprioceptive control in fast legged locomotion," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Vilamoura, Algarve, Portugal), pp. 1970–1975, IEEE, October 2012.

[76] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Blösch, *et al.*, "Anymal-toward legged robots for harsh environments," *Advanced Robotics*, vol. 31, no. 17, pp. 918–931, 2017.

[77] Y. H. Lee, Y. H. Lee, H. Lee, H. Kang, J. H. Lee, J. M. Park, Y. B. Kim, H. Moon, J. C. Koo, and H. R. Choi, "Whole-body control and angular momentum regulation using torque sensors for quadrupedal robots," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 3, p. 66, 2021.

[78] D. Belter, P. Łabecki, P. Fankhauser, and R. Siegwart, "Rgb–d terrain perception and dense mapping for legged robots," *International Journal of Applied Mathematics and Computer Science*, vol. 26, no. 1, pp. 81–97, 2016.

[79] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, and V. Kasi, "Reviews on various inertial measurement unit (imu) sensor applications," *International Journal of Signal Processing Systems*, vol. 1, no. 2, pp. 256–262, 2013.

[80] D. I. H. Putri, C. Machbub, *et al.*, "Gait controllers on humanoid robot using kalman filter and pd controller," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, (Singapore), pp. 36–41, IEEE, November 2018.

[81] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[82] J. B. Rawlings, "Tutorial overview of model predictive control," *IEEE control systems magazine*, vol. 20, no. 3, pp. 38–52, 2000.

[83] M. A. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, no. 3, p. 729, 2012.

[84] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[85] X. A. Wu, T. M. Huh, R. Mukherjee, and M. Cutkosky, "Integrated ground reaction force sensing and terrain classification for small legged robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1125–1132, 2016.

[86] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[87] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *International Conference on Machine Learning*, pp. 1282–1289, PMLR, 2019.

[88] L. Gan, J. W. Grizzle, R. M. Eustice, and M. Ghaffari, "Energy-based legged robots terrain traversability modeling via deep inverse reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8807–8814, 2022.

[89] R. Schoknecht, "Optimality of reinforcement learning algorithms with linear function approximation," *Advances in neural information processing systems*, vol. 15, 2002.

[90] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[91] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[92] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[93] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[94] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[95] K. Dimitropoulos, I. Hatzilygeroudis, and K. Chatzilygeroudis, "A brief survey of sim2real methods for robot learning," in *International Conference on Robotics in Alpe-Adria Danube Region*, (Klagenfurt, Austria), pp. 133–140, Springer, June 2022.

[96] B. Balaji, S. Mallya, S. Genc, S. Gupta, L. Dirac, V. Khare, G. Roy, T. Sun, Y. Tao, B. Townsend, *et al.*, "Deepracer: Autonomous racing platform for experimentation with sim2real reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, (Virtual), pp. 2746–2754, IEEE, June 2020.

[97] D. Rodriguez and S. Behnke, "Deepwalk: Omnidirectional bipedal gait by deep reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, (Virtual), pp. 3033–3039, IEEE, June 2021.

[98] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning.(2016)," *URL http://pybullet. org*, 2016.

[99] D. Hahn, P. Banzet, J. M. Bern, and S. Coros, "Real2sim: Visco-elastic parameter estimation from dynamic motion," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–13, 2019.

[100] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

Academic Achievements

1. SCIE Journal Papers

    - **Jaesik Jeong**, Jeehyun Yang, Guilherme Henrique Galelli Christmann, and Jacky Baltes. "Lightweight Mechatronic System for Humanoid Robot." *The Knowledge Engineering Review*, 2023, 38, e5.
    - Jeehyun Yang, **Jaesik Jeong**, and Jacky Baltes. "An Analysis and Review of Robot Magic Show." *The Knowledge Engineering Review*, 2023, 38, e6.
    - Jeehyun Yang, **Jaesik Jeong**, Eko Rudiawan Jamzuri, and Jacky Baltes. "Humanoid Robot Magic Show Performance." *Multimedia Tools and Applications*, 2023, 1-22.
    - **Jaesik Jeong**, Jeehyun Yang, and Jacky Baltes. "Robot Magic Show as Testbed for Humanoid Robot Interaction." *Entertainment Computing*, 2022, 40, 100456.
    - **Jaesik Jeong**, Jeehyun Yang, and Jacky Baltes. "Robot magic show: human–robot interaction." *The Knowledge Engineering Review*, 2020, 35, e15.

2. International Conference Papers

    - Jyun-Ting Song, Guilherme Christmann, **Jaesik Jeong** and Jacky Baltes. "Reinforcement Learning and Action Space Shaping for a Humanoid Agent in a Highly Dynamic Environment" *IEEE/ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, Springer International Publishing, 2023. 29-42.
    - **Jaesik Jeong**, Jeehyun Yang, and Jacky Baltes. "*Robinion Sr*: Lightweight Humanoid Robot Platform." *International Conference on Advanced Robotics and Intelligent Systems(ARIS)*, 2021.
    - **Jaesik Jeong**, Jeehyun Yang, and Jacky Baltes. "Humanoid robot platform Robinion-2." *International Conference on Advanced Robotics and Intelligent Systems(ARIS)*, 2020.

- Jeehyun Yang, **Jaesik Jeong**, and Jacky Baltes. “Humanoid Robot Magic: various responses and Communication.” *In Proceedings of the 25th International Symposium on Electronic Art –ISEA 2019 ‘Lux Aeterna’*, 2019, 73-78.

3. Awards

- FIRA SimulCup Hurocup Adult-size All-round 1st place (2022).
- Humanoid Taiwan Hurocup Adult-size All-round 1st place (2022).
- FIRA SimulCup Hurocup Adult-size All-round 1st place (2021).
- Humanoid Taiwan Hurocup Adult-size All-round 1st place (2020).
- FIRA Roboworld Cup Hurocup Adult-size All-round 1st place (2019).
- IROS-HAC Robot magic 3rd place (2019).
- Robocup Humanoid league Adult-size Main game 4th place, Technical Challenge 3rd place (2019).
- Humanoid Taiwan Hurocup Adult-size All-round 1st place (2019).
- AUTCUP Hurocup Adult-size All-round 1st place (2019).
- IROS-HAC Robot magic 1st place (2018).
- FIRA Roboworld Cup Hurocup Adult-size All-round 2nd place (2018).
- Robocup Humanoid league Adult-size Technical Challenge 3rd place (2018).
- Humanoid Taiwan Hurocup Adult-size All-round 1st place (2018).
- AUTCUP Hurocup Adult-size All-round 3rd place (2018).
- IROS-HAC Robot magic 2nd place (2017).
- FIRA Roboworld Cup Marathon 2nd place, Weightlifting 3rd place (2017).
- Humanoid Taiwan Hurocup Adult-size All-round 2nd place (2017).
- Robocup Humanoid league Adult-size Main game 4th place (2017).