Computer Vision-enhanced Human-robot Interaction Using a 7 DoF Manipulator with Application to Rehabilitation

By

Saeid Parvanesekam

A Thesis submitted to the Faculty of Graduate and Postdoctoral Studies in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Price Faculty of Engineering

University of Manitoba

Winnipeg, Manitoba, Canada

Abstract

Human-robot Interaction (HRI) plays a crucial role in enabling robots to operate safely alongside a human's upper extremity in a rehabilitation environment. This thesis focuses on the design and implementation of a robust HRI system using a 7 degree-of-freedom (DoF) robot for potential use in upper extremity rehabilitation and motor recovery. The system is designed to support multiple interaction modes, including passive, resistive, and assistive modes, tailored to the potential participant's physical capabilities and rehabilitation needs. A real-time vision system enables the 3D visualization of the human body movement and simulation of the robot in a shared simulation environment. Utilizing a camera and pre-trained models, the human body keypoints and the robot's base frame are detected. This visualization enables precise, real-time monitoring of human body movements, primarily in the robot's proximity.

To boost participants' engagement and therapeutic outcomes, a custom 2D interactive game was created where participants control a paddle to catch a moving ball, emphasizing shoulder joint movement. The paddle is connected to the robot's end-effector, enabling control through physical movement. The robot provides dynamic assistance or resistance based on the selected mode, making the system adaptable for patients with different levels of motor ability. The game-based interface is designed to motivate participation and support rehabilitation through repeated, goal-focused motor exercises for the shoulder joint.

The safe and real-time integration of vision-based human motion capture, multimodal robotic interaction, and gamified therapy creates a versatile platform for potential neurological rehabilitation. This work contributes to the growing field of assistive robotics by providing a scalable, interactive solution that can support physical therapy while maintaining user engagement.

Acknowledgments

First and foremost, I would like to thank God for granting me the strength, patience, and perseverance to reach this stage of my journey.

I am deeply grateful to my supervisor, Professor Nariman Sepehri, for his constant support, insightful advice, and invaluable mentorship throughout the course of this research project. I would also like to express my heartfelt appreciation to Dr. Tony Szturm; his guidance has been instrumental in shaping the direction and quality of this work.

I would like to sincerely thank the late Dr. Witold Kinsner for his support and inspiration. His guidance during my studies was deeply appreciated, and his legacy continues to have a lasting impact on those he mentored.

I am also thankful to the faculty and staff of the Department of Price Engineering at the University of Manitoba for providing an enriching academic environment and the resources necessary to carry out this project. Special thanks to my lab mates and colleagues for their collaboration, technical assistance, and encouragement along the way.

Finally, I want to express my heartfelt appreciation to my family and friends for their unwavering support, understanding, and belief in me. Their encouragement and love have been a constant source of strength throughout every stage of this journey.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background and Motivation

Robots are being used in a wide range of fields, including agriculture, transportation, space exploration, construction, and healthcare. Their characteristics, like precision, repeatability, durability, and adaptability, have made them a necessity for revolutionizing most aspects of these fields. Furthermore, the robot's integration into human life is accelerating, and with each advancement, robots' applications and the ways humans and robots interact become more complex. Due to the complex tasks they must perform, robots should evolve and adapt, particularly in environments that involve human interaction.

Human-robot Interaction (HRI) was initially introduced to free humans from dangerous and routine tasks [1], and increase the effectiveness of the robots' performances in collaboration with humans. HRI can be accomplished through direct and indirect physical interactions, with a variety of options including force/torque sensors (direct physical) and vision (indirect). Despite significant advancements in robotics, achieving full autonomy in robotic systems remains a complex challenge [2]. Therefore, HRI is key to creating future technologies where humans and robots can work together seamlessly [3]. HRI is also a crucial aspect to consider in the development of rehabilitation robotics systems, where physical interaction plays a critical role [4].

Rehabilitation robotics is a growing, promising, and challenging field that has emerged due to various social and medical needs, such as aging populations, neuromuscular, and musculoskeletal disorders [5]. Rehabilitation robots can also be used to train an individual to improve their solo task performances more efficiently [6]. Both patients and clinical professionals have received robotic rehabilitation very well, which is an effective method for motor function therapy in motor impairment patients, such as those with stroke [7].

1.2 Problem Statement

Current HRI systems for rehabilitation face several limitations. Existing platforms often struggle to accurately capture full-body motion in real-time, particularly when patients perform rapid or complex upper-extremity movements near a robot [8]. Moreover, many rehabilitation systems fail to provide engaging interaction or clear movement targets, resulting in reduced patient

motivation and inconsistent training performance [9]. These limitations highlight the need for a more effective HRI system capable of precise motion tracking, safe interaction, and enhanced user engagement to support upper-extremity rehabilitation.

1.3 Objectives

The main objective of this research is to develop a system that can accurately capture participants' motions in real-time to analyze body configurations near a robot. The system must also enable effective interaction and support upper extremity rehabilitation. Furthermore, to boost engagement and improve rehab sessions, the system should provide participants with clear targets during interaction. Tackling these challenges is key to advancing HRI technologies that enable successful rehabilitation outcomes. The properties of the desired framework are as follows:

I. Ensure safe and robust interaction between human and robot in real-time.
II. Enable the framework to support broader applications.
III. Leverage direct physical HRI for rehabilitation purposes.
IV. Provide entertaining interactions by providing a game as a target for the participant.
V. Introduce different modes like assistive, resistive, and passive for tailored sessions.
VI. Record and track human game performances in different modes of interaction.

1.4 Thesis Structure

This thesis is organized as follows. Chapter 2 reviews related topics and concepts. Chapter 3 provides a detailed explanation of the system architecture, followed by Chapter 4, which describes how the robot is controlled. Chapter 5 discusses the vision system used in the interaction framework. The rehabilitation application of the framework is thoroughly explained in Chapter 6. Chapter 7 presents the results, followed by discussions. The conclusions and future work are covered in Chapter 8.

2 Literature Review

2.1 Human-robot Interaction

Human-robot Interaction (HRI) refers to scenarios where humans and robots work in the same environment or collaborate. HRI has gained significant attention recently and is used in key applications such as manufacturing, warehousing, agriculture, and healthcare. HRI strategies help address complex problems like ensuring security, reducing workload, enhancing comfort, and boosting productivity [10] [11]. This field explicitly focuses on designing and developing collaborative robotic systems that can work with humans to achieve a common goal. It combines artificial intelligence, robotics, ergonomics, engineering, computer science, and social science to enable the interaction of robots and humans with all the required competencies [1].

Robot capabilities are maturing across domains. Robots are no longer confined to safety-controlled industrial settings. Instead, they directly interact with the public [12]. For this reason, HRI has become an inseparable component of the development of intelligent robotics systems. It paves the way for bringing robots into daily life and various industries, such as transportation, manufacturing, healthcare, and rehabilitation. HRI has revolutionized human-robot collaboration by making interactions more intelligent, enabling a variety of uses across a wide range of applications.

HRI has been categorized in various ways across different studies. For instance, based on the human's role, the robot's role, the level of autonomy, types of communication between two, applications, or characteristics of the robots [13] [14] [15]. Two main categories of HRIs are collaborative HRI and physical HRI, which overlap in some areas. In collaborative HRI, humans act as a team with robots to achieve a goal, and it comes from the confluence of information exchange, autonomy, and optimal task shaping [16]. Physical HRI, on the other hand, involves systems in which the physical states of an automation interact with the physical states of a human [17]. There are various applications for these types of HRI, such as assembling components alongside human workers on an assembly line [18], or applications in the healthcare industry where robots are leveraged to provide rehabilitation sessions for patients [19].

Communication between the robot and the human is also vital for establishing a robust and effective HRI. A communication channel can send data and information from a human to a robot

or vice versa [20]. Different types of communication channels include hearing, sight, and touch. Hearing channels allow humans to control the robot through speech or for the robot to deliver information to humans via voice. For example, it can show its feelings to humans [21]. The sight channel enables the robot to capture data from humans and display required information through images, light, and motion. For example, some systems can detect emotion from facial expressions [22]. The touch channel gets information from direct physical contact with the human by various sensors like specific buttons, torque sensors, gyroscopes, artificial skin, and more [23] [24].

Growth in HRI has also increased the challenges of developing more advanced and robust HRI that can fulfill the requirements. For instance, despite advancements in robotics, the deployment of autonomous robots in our everyday lives is still an open challenge due to reasons such as interruptions or delayed responses. Real-time solutions are crucial for HRI challenges. Robots should be able to capture and interpret data and make decisions promptly to meet standards. Task dynamics analysis is also another challenge, which focuses on the allocation of tasks between humans and robots [25]. Other challenges are safety, effectiveness, task allocation, real-time feedback, and responsiveness. These challenges have led to the use of state-of-the-art sensors, such as cameras, and to the adoption of strict standards, such as real-time responses.

2.2 Robotics Applications in Rehabilitation

A wide range of robots has been developed to support clinicians, caregivers, and other stakeholders in healthcare contexts [26]. Various types of robots, including exoskeletons, soft robots, and manipulators, have been widely used for rehabilitation. For example, physically assistive robots, such as exoskeletons and robotic prostheses, can help individuals perform tasks like walking and reaching. Socially assistive robots can help individuals engage in positive health behaviors, such as exercise and wellness [27].

Robots, compared to humans, are more effective at accomplishing tasks that require repetitive movements. As Artificial Intelligence (AI) advances, robots are becoming increasingly intelligent, enabling them to perform more complex tasks independently. The robots have great potential to provide practical therapy sessions and are excellent candidates to become therapists or, at the very least, invaluable therapy tools. Robot-assisted rehabilitation therapy has been proven to successfully enhance the upper limb motor function of patients with motor dysfunction,

according to the motor relearning program theory, which suggests that repetitive training movements can restore motor function [28].

Exoskeletons are one of the most used types of robotics systems in rehabilitation. Exoskeletons are increasingly applied during overground gait and balance rehabilitation following neurological impairment [28]. They can be divided into four main groups: end-effectors, grounded exoskeletons, wearable exoskeletons, and soft exoskeletons [29]. Some diseases and conditions for which exoskeletons are used are Spinal Cord Injury (SCI), post-stroke impaired movement, and Multiple Sclerosis (MS) [30] [31]. Figure 2.1 shows two types of wearable exoskeletons.

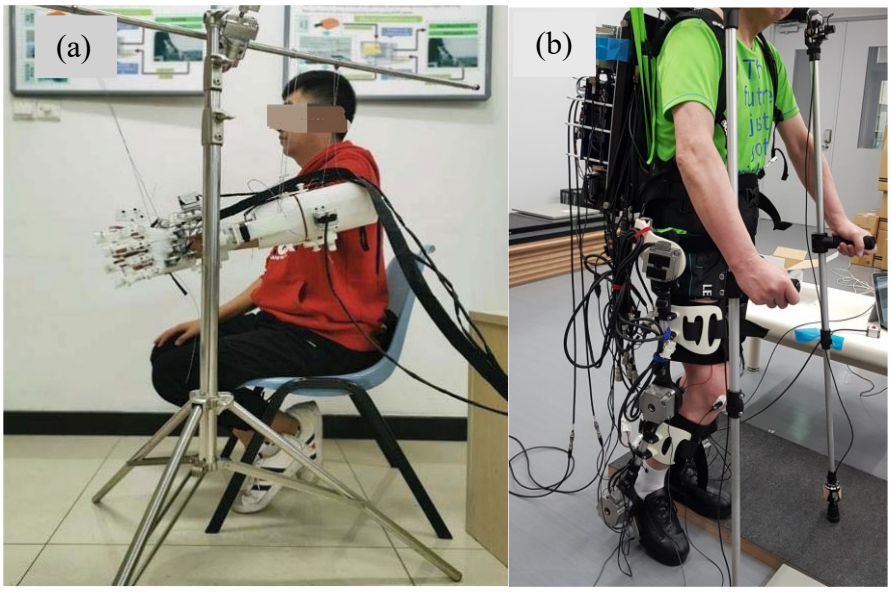

Figure 2.1: (a) Demonstration of a wearable exoskeleton device [32]; (b) Experimental setup for gait pattern following with TTI-Knuckle1 exoskeleton [33].

Soft robotics devices have been developed in recent years for the rehabilitation of major joints like knees, shoulders, wrists, and fingers [34] [35] [36] [37]. Developing soft robotics has its own challenges: designing a control algorithm due to the nature of the soft robots, choosing the material for durability, and pressure handling [38]. Soft robots are still being developed and, in the near future, can revolutionize rehabilitation for delicate body parts, such as fingers. Figure 2.2 shows examples of soft robotics systems.

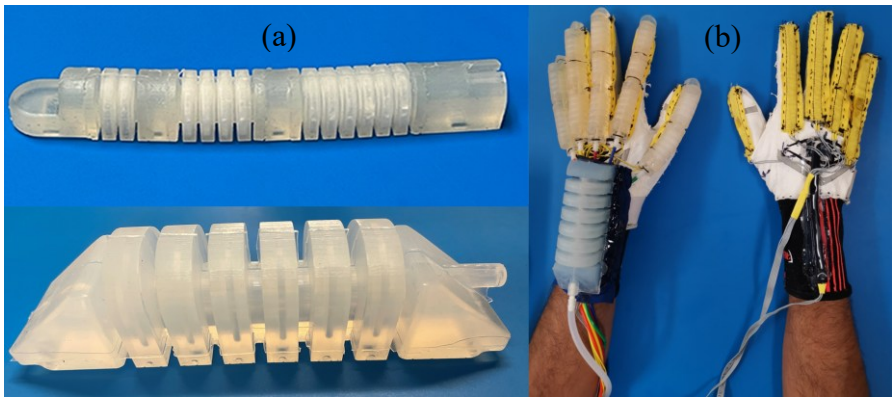

Figure 2.2: Soft robotic bilateral therapy system: (a) Finger and wrist actuators; (b) hand-worn soft glove (HWSG) and hand-worn exoskeleton (HWE) [39].

Manipulators are widely used in different industries and applications. Robotics manipulators are well-suited for various types of movements and are adaptable to body movements in rehabilitation, depending on their degree-of-Freedom (DoF) [28]. Using manipulators for rehabilitation sessions has some advantages. Robotics manipulators can be in different configurations and positions relative to the patients (see Figure 2.3). Depending on their configuration, robotic manipulators can perform different types of movements, making them a good candidate for use in various rehabilitation sessions. In addition, manipulators can be used for the assessment of the rehabilitation [40]. Using movement/force sensors, they can record patients' performance and track it over time. This can provide insights for rehabilitation specialists.

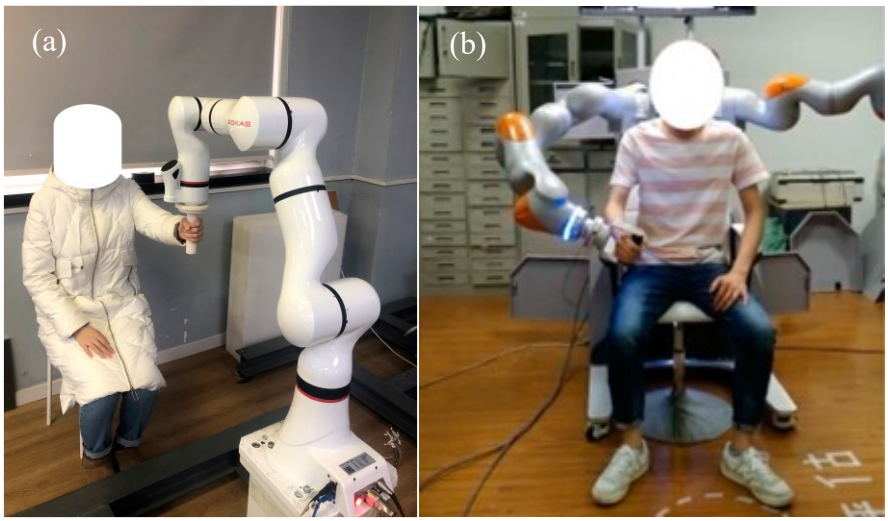

Figure 2.3: Upper-limb rehabilitation session involving physical HRI: (a) using a single collaborative robotic manipulator [41]; (b) using a multi-DoF robotic system [28].

Safety and stability, patient-dependent variability, controller design, and noise are some challenges [41]. For example, in settings where rehabilitation robots assist individuals with motion, the physical safety of the patient must be considered, making the robot prioritize human safety [12]. Adapting and optimizing manipulators' movements to the target joint and adjusting their end-position configuration is another challenge.

2.3 Computer Vision in Robotics

Computer vision is a revolutionary technology taking robotics to the next level. It enables robots to perceive their environment, interact with the real world, and take relevant actions and decisions based on visual information [42]. The most common sensor in computer vision is RGB (red, green, and blue) cameras, which provide detailed information by capturing light in red, green, and blue wavelengths and creating a color representation of the environment around [43].

Computer vision has been used in robotics for various purposes, such as simultaneous localization and mapping (SLAM), object detection (see Figure 2.4), and collaborative actions [43]. For these complex application purposes and building a robust computer vision system that can perform well, besides the RGB camera, different types of sensors such as LiDAR (Light Detection and Ranging), stereo vision cameras, IR (infrared) sensors, thermal cameras, and combined sensors like RGB-D (Red, Green, Blue and Depth) cameras can also be used.

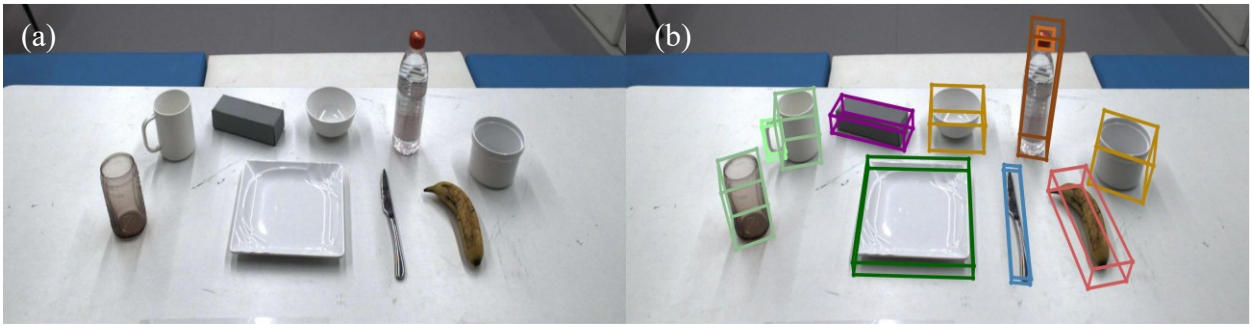

Figure 2.4: (a) Input stereo images with various object surface properties (diffuse, specular, transparent, and mixed); (b) Estimated object categories, 6D poses, and sizes [44].

Other techniques, include gesture and face detection, 2D and 3D human body pose estimation, motion tracking, classification of the image of interest [42] [45] [46] [47] [48]. These features allow robots to interact securely and effectively with humans. Interactions often include various types of social signals, such as facial expressions, speech, and body language from both

humans and robots, which, if better understood, can be used as cues to promote more natural interactions [49].

Real-time 3D human pose estimation allows robots to detect nearby humans and adapt their behavior instantly. This method is aided by AI and machine learning, where AI models are trained on a dataset of human bodies, enabling the creation of models that can detect human body keypoints from RGB images. Simultaneous access to depth and RGB frames, enables the construction of body configurations in 3D. RGB-D cameras provide both types of data at high frequency, and with advanced computational devices, this approach facilitates real-time human pose detection. Figure 2.5 demonstrates an example of human visualization in the proximity of a robot through 6 cameras. Figure 2.6 shows an example of a human body model.

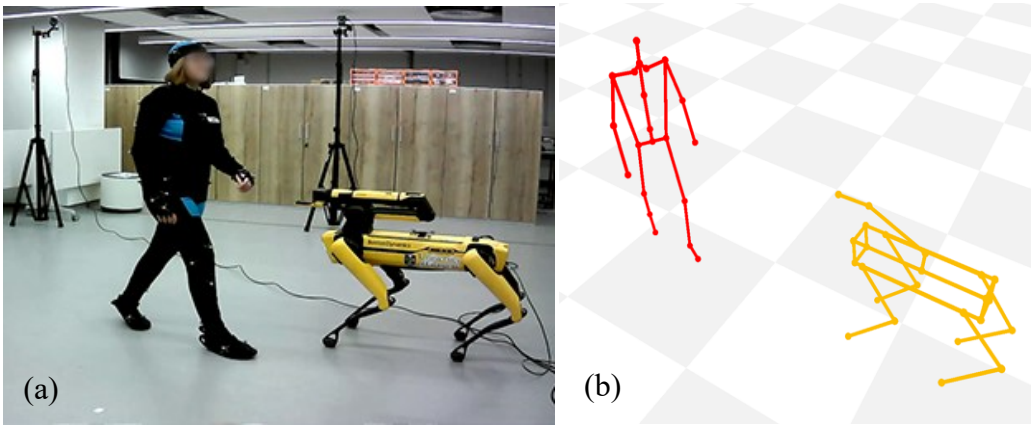

Figure 2.5: (a) A human and a robot share an environment-the scene captured via six OptiTrack cameras; (b) 3D human poses represented with 21 joints, and the robot itself [50] [51].

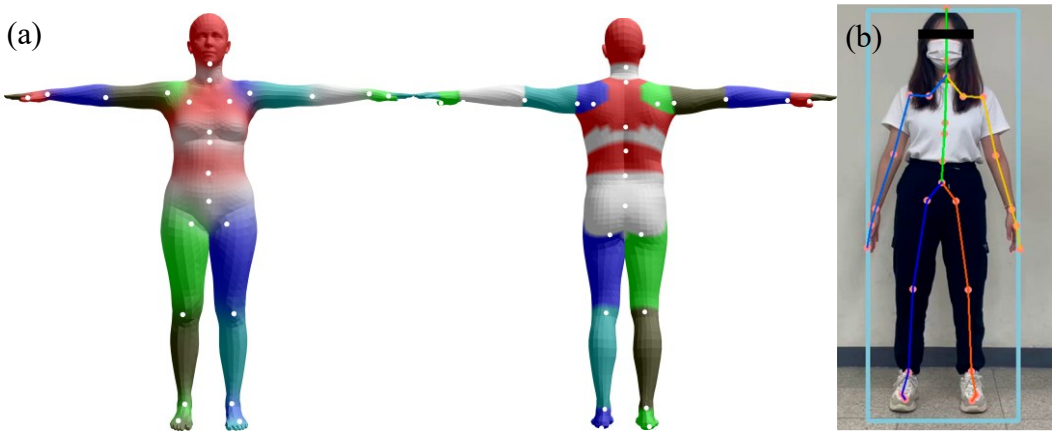

Figure 2.6: (a) Skinned Multi-Person Linear (SMPL) model [52]; (b) Human body keypoints detected by a pre-trained model [51].

2.4 Summary

This chapter provided a review of the literature related to HRI and its relevance to rehabilitation. It began by defining HRI and emphasizing its interdisciplinary nature, involving robotics, computer science, ergonomics, and healthcare. Various types of HRI were introduced, including collaborative, assistive, and social interactions, along with common communication modalities such as verbal, non-verbal, and physical interfaces. A detailed exploration of robotics applications in rehabilitation highlighted the use of exoskeletons, robotic limbs to assist patients with neurological or motor impairments. The section also discussed the role of computer vision in enhancing real-time human modeling and interaction, which is crucial for tasks such as motion tracking and pose estimation. By analyzing current technologies and identifying gaps, such as real-time performance and adaptability, this chapter lays the foundation for this thesis, which aims to improve robotic rehabilitation through enhanced Human-robot Interaction.

3 System Architecture

3.1 Proposed System Overview

A system is proposed whose schematic is depicted in Figure 3.1. The system consists of four main components: (a) a 7 degree-of-Freedom (DoF) manipulator robot, (b) a PC, (c) an RGB-D (Red, Green, Blue, and Depth) camera, and (d) a game module.

With reference to Figure 3.1, the 7 DoF robot is controlled via direct joint torque control and PD control of end-effector position by a dedicated algorithm embedded on the PC. To ease the interaction, both the robot and the human grasp the same handle, allowing them to exchange force/motion. The RGB-D camera captures RGB and depth data, sends it to the PC, and updates the human kinematic model. Simultaneously, robot sensors update the robot's visualization, ensuring an accurate representation of the interaction.

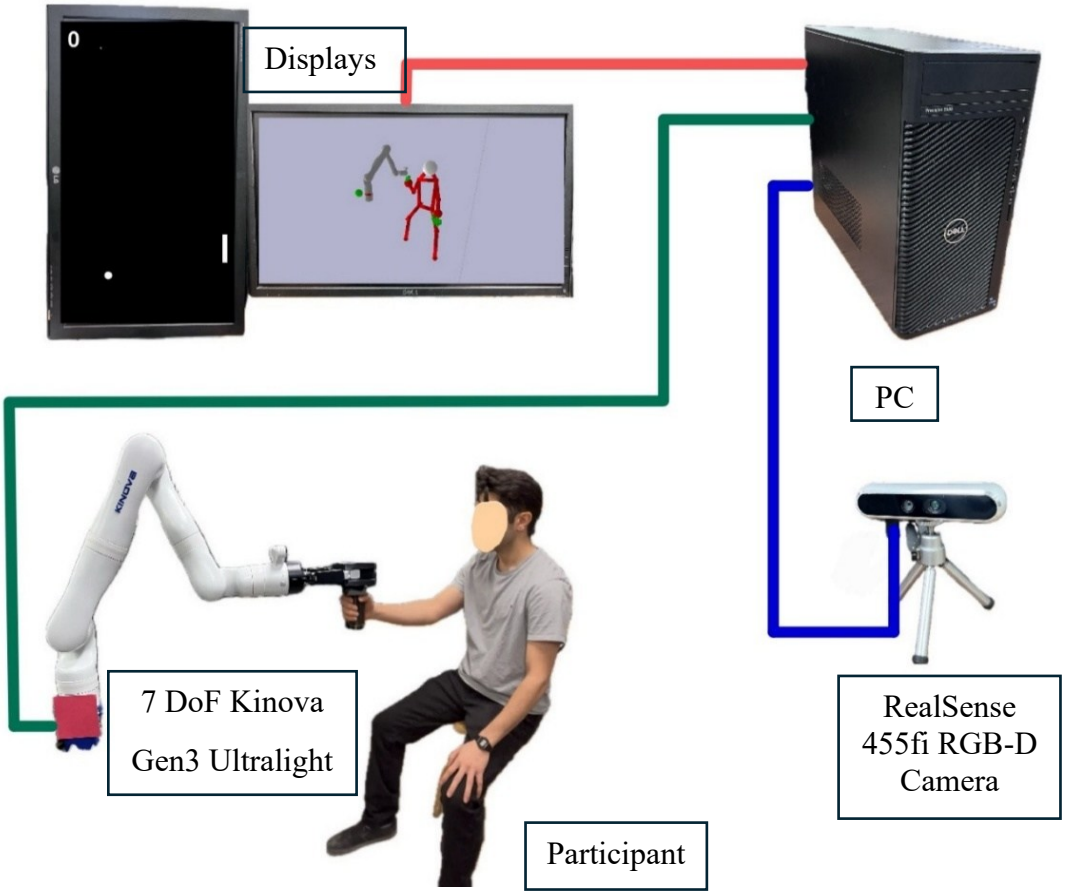

Figure 3.1: Schematic of the proposed system.

For rehabilitation purposes, a game is presented to the user, who interacts with it by moving the robot's end-effector. During gameplay, the robot adjusts the difficulty based on the selected game mode, either simplifying or challenging the task for the human participant. This system facilitates seamless and robust interaction between the human and the robot, while also enabling their coexistence within a shared environment. Furthermore, it provides a rehabilitative gaming platform that supports the assessment of Human-robot Interaction (HRI) strategies and therapeutic interventions.

3.2 Hardware Components

The 7 DoF manipulator, the RGB-D camera, and the computation and interfacing unit are discussed, highlighting their specifications, operational principles, and integration within the system architecture.

3.2.1 Robotics Manipulator

The robot used in this system is a 7 DoF Kinova Gen3 Ultralight equipped with a 2F-85 Robotiq gripper [53] [54]. With its seven revolute joints, depicted in Figure 3.2, and a variety of integrated sensors, the arm supports a wide range of research and assistive applications. The Gen3 model can exert up to 20 N of force at the end-effector in any direction, making it suitable for various manipulation tasks. The Kinova Gen3 robotic arm is actuated by brushless DC (BLDC) servo motors equipped with high-resolution encoders, enabling precise position and torque control. The robot and its gripper, while holding a handle, are shown in Figure 3.3.

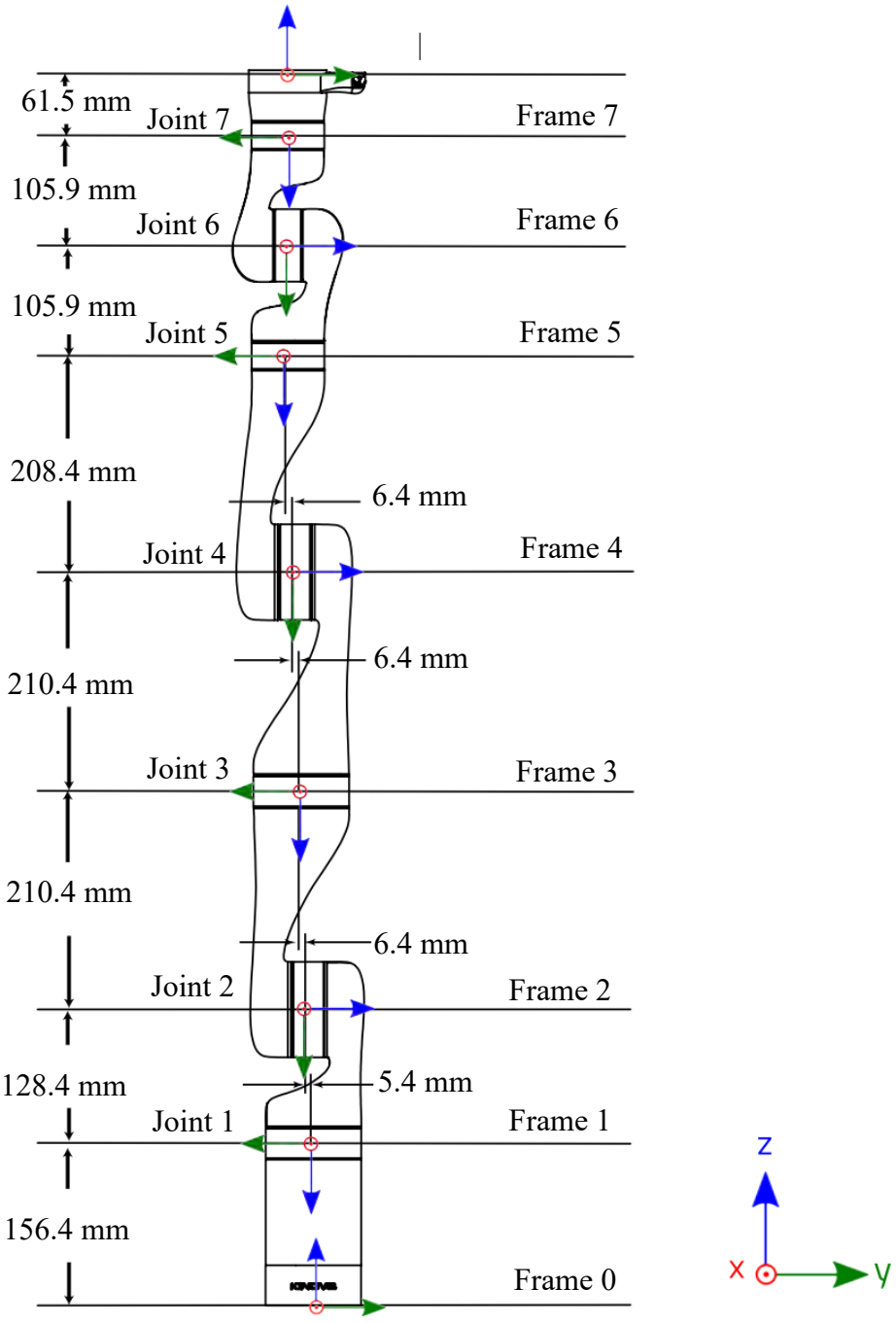

Figure 3.2: Kinova Gen3 7 DoF robot schematic, joints, coordinate frames, and dimensions [53].

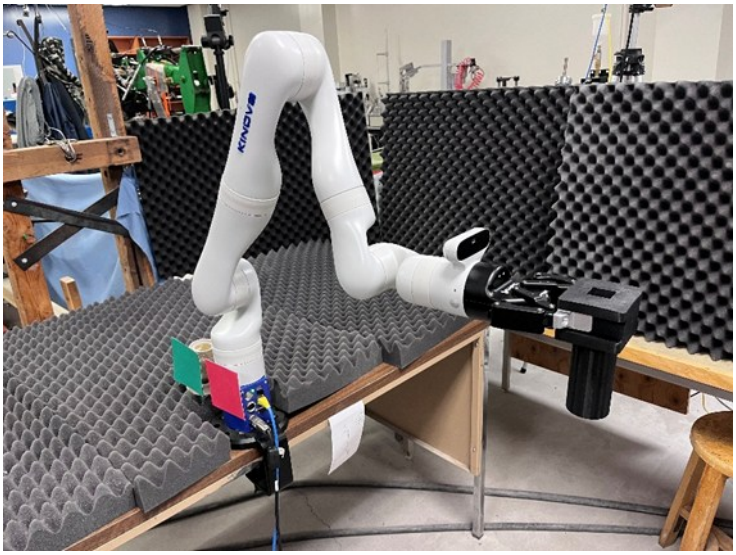

Figure 3.3: Kinova Gen3 Ultralight equipped with a 2F-85 Robotiq gripper, is holding the designed handle for interaction with human.

The robot can be controlled at a frequency of 1000 Hz. Motor drivers continuously receive feedback from a network of embedded sensors, including high-resolution joint encoders for precise position tracking, torque sensors for measuring the applied motor torque, and integrated current sensors for monitoring motor health and performance. This high-frequency data acquisition system meets the requirements of an HRI system, enabling the manipulator to respond in real-time to human actions in its proximity. For instance, the robot can react quickly to forces applied to its end-effector by the user or avoid collisions, ensuring safe and compliant interactions.

3.2.2 Vision System

The RGB-D camera is an Intel RealSense 455f [55], capable of streaming both RGB and depth frames at up to 60 Hz. This high-frequency data stream is well-suited for real-time applications that require spatial awareness and perception. With a practical depth range of 0.2 to 3 meters and an accuracy of 1 to 2 percent, it delivers reliable performance in scenarios where precise detection of user movements and scene geometry is crucial. The camera connects to the main PC via USB.

Within the overall system, the camera captures the entire interaction scene, including the robot and the participant. From these data streams, the system identifies the participant's body keypoints positions. In parallel, the camera data are fused with the manipulator’s angular position sensors and torque sensors readings to create an integrated view of the environment. This feedback

loop ultimately supports rehabilitation, gaming experiences, and reliable HRI by allowing the system to track user actions in real-time and dynamically adapt the robot's movements.

3.2.3 Computation and Interfacing Unit

The PC is the system's central hub and serves as the computational and interfacing unit. It facilitates data flow between the robot, camera, and other system elements. The PC leveraged in this system has a Core i9 multi-core CPU and a high-performance GPU (RTX 4090). This hardware setup provides sufficient processing power to handle simultaneous real-time vision tasks, robotic control algorithms, and user-interface applications.

The PC communicates with the robotic manipulator at 1000 Hz over a wired local Ethernet connection. This communication enables the PC to send up to 1000 torque or position commands per second while receiving joint position and torque sensor feedback at the same rate. In parallel, the PC connects to the RGB-D camera via USB 3.0, capturing color and depth frames at 30 to 60 Hz, depending on the operational mode. Simultaneously, the PC runs a rehabilitation game, along with other algorithms and visualization processes, ensuring that all critical functions integrate smoothly into HRI.

3.3 Software Components

In addition to hardware components, essential software modules manage functions such as computer vision, robotic simulation, low-level robot control, and game-based user interaction. A detailed explanation of these modules is provided in this section.

3.3.1 Computer Vision Modules

The computer vision modules integrate multiple libraries to process RGB and depth data, deriving valuable information for real-time visualization. The primary modules include OpenCV, YOLOv8, MediaPipe, RealSense, NumPy, and additional filtering and denoising modules. OpenCV handles image preprocessing, object detection, and contour detection. RealSense SDK captures and filters RGB and depth data. NumPy is used for efficient array operations and numerical computations. Kalman filter and k-means clustering are applied for depth refinement and noise reduction. YOLOv8 is utilized for human body pose detection, while MediaPipe Hand detects and tracks hand keypoints.

3.3.2 Visualization Environment

PyBullet is used to build a shared visualization environment. PyBullet is an open-source, powerful, real-time physics engine that models and visualizes robotic systems, human body, and the interaction between humans and robots. PyBullet can integrate with robots and can be used to test motion planning and control algorithms before deploying them on real hardware. It supports rigid-body dynamics and collision detection and is compatible with Unified Robot Description Format (URDF).

3.3.3 Control

The robot operates in low-level mode, enabling precise control of its actuators through torque inputs while continuously accessing sensor data at 1000 Hz. This approach allows for significant flexibility in designing custom algorithms and supports the development of robust, real-time HRI applications.

In this study, the robot is controlled by an algorithm capable of functioning in both direct joint torque control and PD end-effector position control modes, either separately or simultaneously. The control gains are adjustable, offering adaptability in HRI modes. The algorithm constantly receives feedback from the robot's sensors and the rehabilitation game, and using this information, computes and delivers appropriate torque commands. This control strategy ensures precise, responsive, and customizable interactions, facilitating a highly adaptive and engaging rehabilitation experience.

3.3.4 Game

Incorporating a game into the system can enhance participant engagement and serve as a valuable performance metric, particularly for research. The game fosters active involvement by providing clear goals and interactive elements, thereby motivating participants to participate actively. Additionally, the data collected during gameplay can offer insights into participant performance, enabling effective assessment and analysis. The designed game is simple, featuring an easy mechanism with a paddle that must catch a moving ball. A screenshot of the game is shown in Figure 3.4.

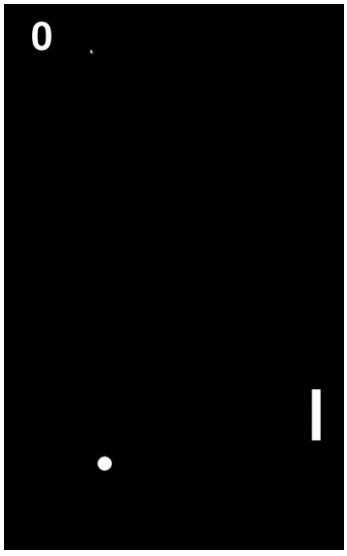

Figure 3.4: The Game screenshot.

3.4 Data Processing

The primary data sources in the current system are camera data, the robot's actuators, torque sensors, and the robot's angular position sensors. As mentioned earlier, the camera captures the interaction scene at 30 to 60 Hz. It provides sufficient data for vision algorithms running on the PC to detect human body joints and update the visualization. The torque sensors, on the other hand, enable calculating and classifying the applied torques on the robot's actuators at 1000 Hz, while the angular position sensors provide data to update the robot's configuration and positions, also at 1000 Hz.

3.4.1 Human Body Joints Detection

To detect human body keypoints using RGB and depth frames, and extract corresponding 3D coordinates, two primary pre-trained models are used: YOLOv8n-pose and MediaPipe Hands. Yolov8n-pose detects 17 keypoints of the human body, including the head, shoulder joints, elbows, wrists, hips, knees, and ankles, at approximately 60 Hz shown in Figure 3.5. MediaPipe Hands detection module detects 21 keypoints per hand, capturing finger movements at around 20 Hz shown in Figure 3.6.

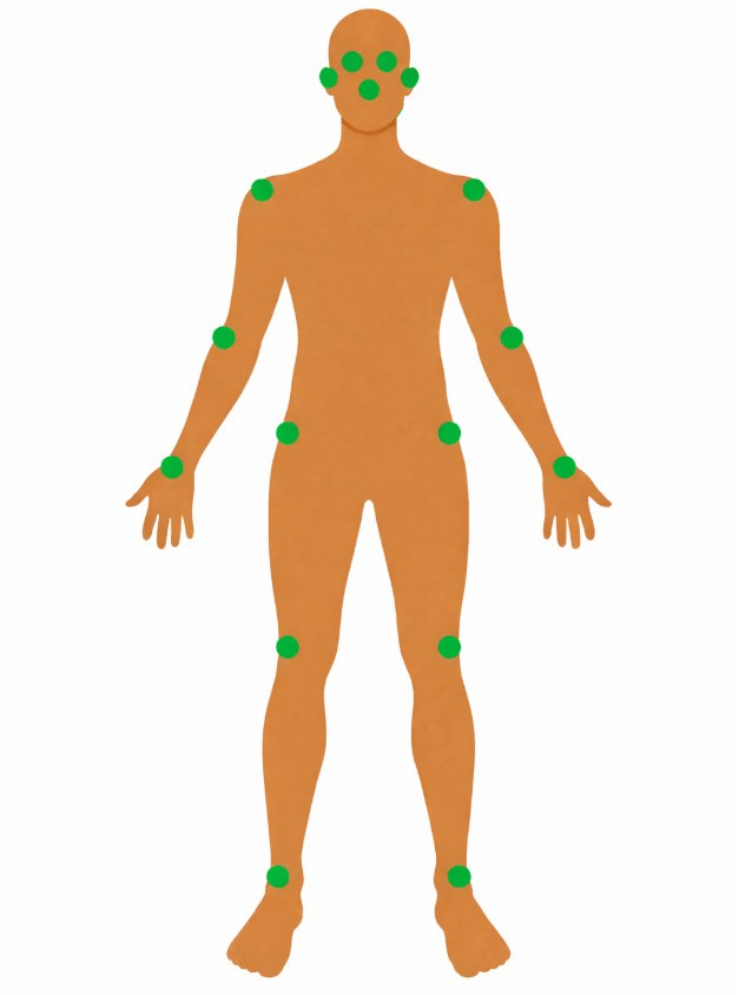

Figure 3.5: Body keypoints detected by YOLOv8n-pose.

Figure 3.6: Hand keypoints detected by MediaPipe Hands.

3.4.2 Real-time Data Processing and Transfer

After obtaining the 2D coordinates of the keypoints from the color frame, their respective depth values are then extracted from the depth frame. Using this information, the 3D spatial coordinates for each keypoint are calculated, providing a more comprehensive understanding of their position in 3D space.

One challenge is in reading depth values, as the depth frame contains noise and misalignment with RGB images [54] [55]. To improve the accuracy and stability of depth readings, different filters are applied to the depth frame, including hole-filling, spatial, and temporal filters, to reduce noise. Before updating the human kinematic model in the visualization environment, the keypoint data are filtered using a custom filter that applies k-means clustering to remove false depth data and a Kalman filter to reduce noise. The whole process, from capturing the frames to deriving 3D coordinates of keypoints, runs at around 20 Hz when detecting whole-body keypoints and 30 Hz when detecting hand keypoints are disabled.

In parallel, the robot control algorithm runs at 1000 Hz and processes user inputs and sensor feedback to control the manipulator's response in real-time in a separate thread. Simultaneously, in another thread, a stream continuously provides a visual representation of the interface for potential logging or analysis. The game, displayed alongside these processes, responds to the human's movements via the robot's end-effector, making gameplay part of the interaction and feedback loop. All components, including vision processing, robot control, game logic, and data streaming, operate in a multithreaded architecture, ensuring that each module runs efficiently and communicates seamlessly with the others. This robust, real-time data processing pipeline enables smooth and reactive Human-robot Interaction, critical for both functional assistance and engaging rehabilitation experiences.

3.5 Summary

Overall, the proposed system integrates a 7 DoF Kinova Gen3 manipulator, an Intel RealSense RGB-D camera, and a high-performance computing unit to support real-time HRI in a shared environment. The system’s software architecture combines advanced computer vision, low-level robot control, real-time visualization, and gamified HRI, all operating in a multithreaded framework to ensure responsiveness and stability, suitable for potential use in rehabilitation

applications. Real-time data acquisition, filtering, and processing enable accurate synchronization between the human and robot models in both physical and virtual environments.

4 Robot Control

4.1 Robot Kinematics and Dynamics

In controlling robotics systems, two fundamental concepts should be considered: Kinematics and Dynamics. Kinematics deals with robots' motion without considering the forces. It analyzes motion based on velocity, position, and acceleration, without considering force. In Forward Kinematics, the approach is to calculate the robot's end-effector position and orientation from the joint angles. Conversely, in Inverse Kinematics, the joint angles are calculated for a desired end-effector position and orientation.

4.1.1 Forward Kinematics

In this research, the primary purposes of the Forward Kinematics are to synchronize the paddle position in the game with the robotic manipulator's end-effector position and to simulate the robot in the shared simulation environment in real-time. The Forward Kinematics of the robotic manipulator is computed using PyBullet using the robot's URDF (Unified Robot Description Format) model. PyBullet provides an efficient built-in function for calculating Forward Kinematics, which determines the end-effector position and orientation based on the robot's joint angles. The Forward Kinematics calculations are performed at 60 Hz, enabling real-time updates to the paddle position in the game and ensuring smooth, responsive gameplay.

4.1.2 Dynamics

Dynamics analyzes a robot's motion based on the forces and torques that influence it. It incorporates how forces such as gravity, inertia, friction, and external forces affect the robot's motion. This is crucial for real-time robot control, where accurate force estimation is vital for stable, responsive behavior.

To derive the dynamic equation, there are two approaches, Lagrangian and Newton-Euler. The Lagrangian formulation relies on energy principles and uses kinetic energy and potential energy to derive equations. On the other hand, the Newton-Euler approach is based on Newton's second law for linear motion and Euler's equations for rotational motions to derive the equations of motion. The Newton-Euler approach has a lower computational cost in comparison to

Lagrangian approach, which makes it more appropriate for real-time applications, for instance, a robot with 6 links, Newton-Euler is 100 times more efficient [56].

To specifically evaluate computational efficiency, the total number of multiplications and additions is counted for each approach, based on the number of links in a robotics manipulator. The Newton-Euler equation, used for dynamic calculation, is sufficiently simple to warrant easy real-time implementation [57]. For a typical case of a 6-link manipulator, Newton-Euler equations are 100 times more efficient than Lagrangian [56]. This advantage increases with the number of links, ensuring Newton-Euler's suitability for real-time robotic control applications.

Lagrangian general dynamic equation:

$$\tau_i = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + \tau_{ext} \quad (1)$$

where, τ_i is the torque applied on the i_{th} joint by gravity, inertia, external, and Coriolis forces, q is a joint position, $\dot{q}$ is joint velocity, and $\ddot{q}$ is joint acceleration.

Various forces influence motion, including gravity $G(q)$, inertia $M(q)\ddot{q}$, Coriolis forces $C(q,\dot{q})\dot{q}$, and external torques τ_{ext}. Torques generated by gravity at the robot joints, which result in downward forces on the robot's end-effector, must be compensated in real-time for smooth, robust motion, particularly in HRI systems. Inertia is the tendency of a robotic link to resist changes in motion, determined by its mass and distribution. Coriolis forces arise from motion, and velocity induces an additional force that affects the robot's motion. External forces are those applied to the robot by other sources, such as humans in HRI.

As discussed, the primary forces affecting robotic manipulator motion include gravity, inertia, Coriolis forces, and external forces. Among these, the effects of inertia and Coriolis forces are particularly significant in high-speed robotic applications. However, due to the low-speed application of the HRI system in this study, and to reduce computational cost for real-time applications, only the gravity term is used. The external forces are treated as external torque applied by humans, while inertia and Coriolis torque are neglected.

The torque equation to model each joint is formulated as follows:

$$\tau_i = \tau_{gravity_i} + \tau_{human_i} \quad (2)$$

$$\tau_{gravity_i} = G(q) \quad (3)$$

where, τ_i is the total torque applied on the i_{th} joint, $\tau_{gravity_i}$ is the torque applied on the i_{th} joint by gravity, and τ_{human_i} is the torque applied on the i_{th} joint by a human.

The robot sensors provide applied torque and joint position values. τ_i is the torque that each sensor shows, and by subtracting $\tau_{gravity_i}$ from τ_i, τ_{human_i} is derived. So, computing the torques exerted by gravity at the robot's joints enables compensating for the gravitational force acting on the robotic manipulator, and deriving the torques applied to each joint by a human. To solve this problem, the torque applied to each joint of the robotic manipulator due to gravity should be calculated and then compensated by generating an opposing torque, thereby stabilizing the manipulator in each configuration. To achieve this goal, only the gravity term of the dynamic equation is derived. Consequently, seven gravity torque equations are derived for the 7 degree-of-freedom (DoF) robot:

$$\tau_{gravity_i} = G(q_1,\ q_2, \dots, q_7)\ \ for\ i = 1,2,\dots,7 \tag{4}$$

Each equation corresponds to a joint and computes the torque exerted by gravity on that joint, based on the robot's joint positions. Since the torque applied to a joint depends on the robot's overall configuration, for each joint, all seven angular positions are needed as inputs to compute $\tau_{gravity_i}$. These equations are used in the robot's control algorithm to enable smooth, robust HRI. The derived gravity terms are presented in the Appendix Equation section, where each joint's gravitational torque equation is explicitly formulated.

4.2 Control

In controlling the proposed system, several control objectives must be addressed, including managing the robot's stiffness in response to the human's applied force and controlling the end-effector's position relative to potential targets. The controller must ensure this movement is executed smoothly and safely, and that it responds to feedback in real-time. To achieve the mentioned architecture, two control techniques are used: proportional-derivative (PD) position control and direct joint torque control. The PD controller ensures the manipulator's end-effector moves toward its goal position.

$$u(t) = K_P\, e(t) + K_D \frac{d}{dt} e(t) \tag{5}$$

$$e(t) = r(t) - y(t) \tag{6}$$

where, $r(t)$ is the reference (desired) signal, $y(t)$ is the system (measured) output. $u(t)$ is control output, $e(t)$ is the error signal (difference between reference and actual state), K_P is proportional term gain, $K_D$is the derivative term gain, and t is time.

Direct joint torque control is a key technique for enabling physical HRI. Operating directly in the torque domain, this control approach allows the robot to respond in real-time to external forces by compensating, reducing, or amplifying them. Such an approach is essential in HRI scenarios as it allows users to safely influence the robot's motion or configuration in real-time while the controller compensates for gravity.

In the direct joint torque control, real-time torque and position feedback are obtained from each joint sensor. Using the previously derived dynamic equations, the gravity-generated torque ($\tau_{gravity_i}$) at the i_{th} joint is computed based on the measured joint positions. Under static condition, the sensor-measured joint torque at i_{th} joint (τ_{sensor_i}) represents the combined effect of gravity-induced torque and the torque applied by the human user. By subtracting the calculated gravity torque from the sensor-measured torque, the torque applied by the human at i_{th} joint (τ_{human_i}) can be estimated as:

$$\tau_{human_i} = \tau_{sensor_i} - \tau_{gravity_i} \tag{7}$$

where, τ_{sensor_i} is sensor-measured torque at i_{th} joint.

When a participant does not interact with the robot, the human-applied joint torques are assumed to be negligible, and the torque measurements obtained from the robot's sensors primarily reflect the gravitational torques acting at each joint. Under this condition, the torques generated by the robot closely match the analytically computed gravity compensation torques.

When a human moves together with the robot at sufficiently low speeds, dynamic effects are minimal, and the sensor-measured torques are expected to closely match the calculated gravitational torques at each joint. To evaluate this assumption, the robot was moved slowly while applying the gravity compensation algorithm. The corresponding end-effector trajectory and robot configurations are shown in Figure 4.1.

Figure 4.2 illustrates the comparison between the calculated gravity torque and the sensor-measured torque for Joint 2. For consistency across all seven joints, the absolute values of both the

calculated gravity compensation torques and the sensor-measured torques are presented in Figures 4.3 to 4.9, demonstrating the accuracy of the derived dynamic equations.

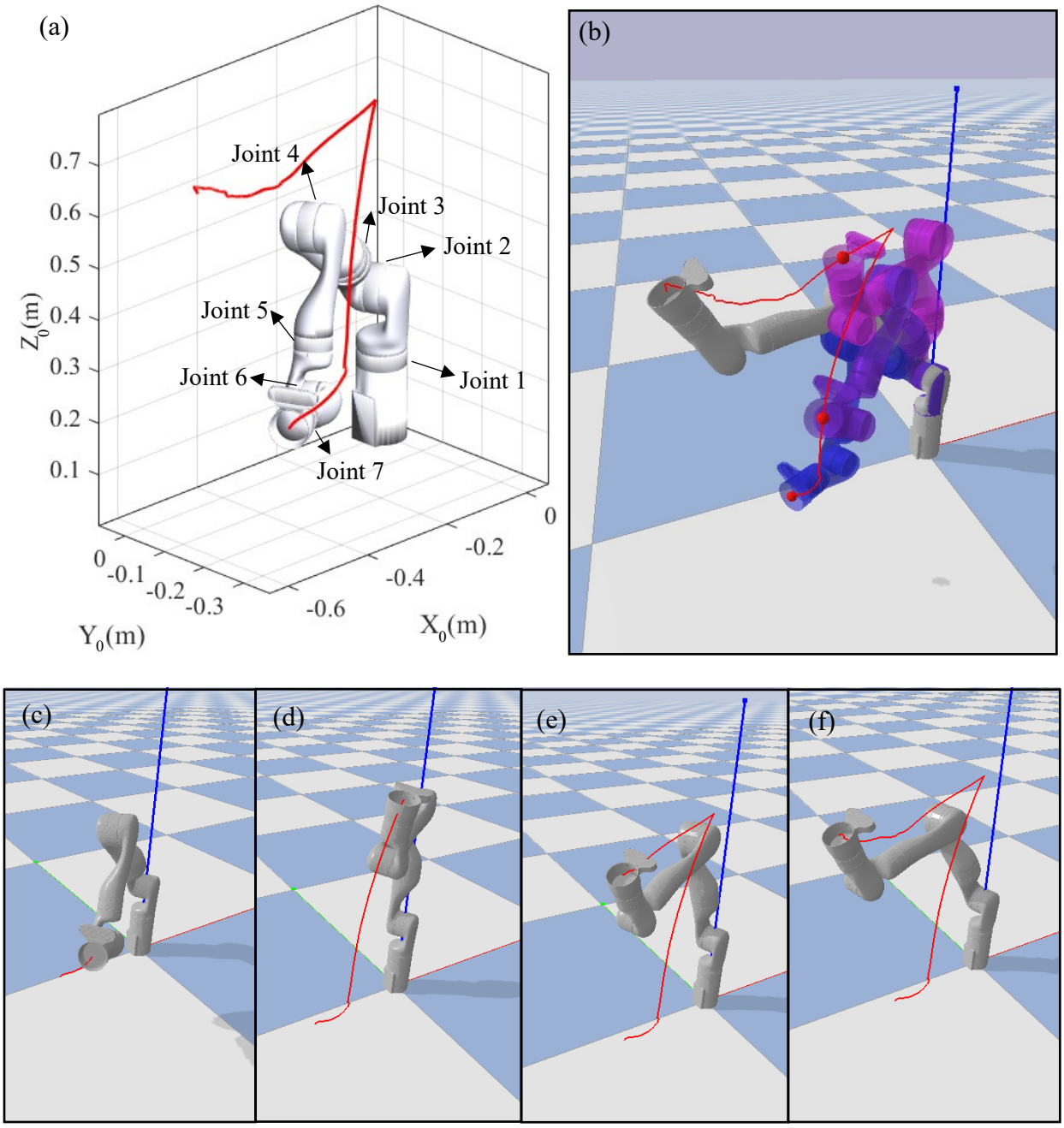

Figure 4.1: (a) End-effector trajectory represented in the robot base coordinate frame; (b) Robot configurations during task execution, including the corresponding end-effector path; (c) to (f) Selected time instances along the trajectory showing the progression of robot joint configurations during motion.

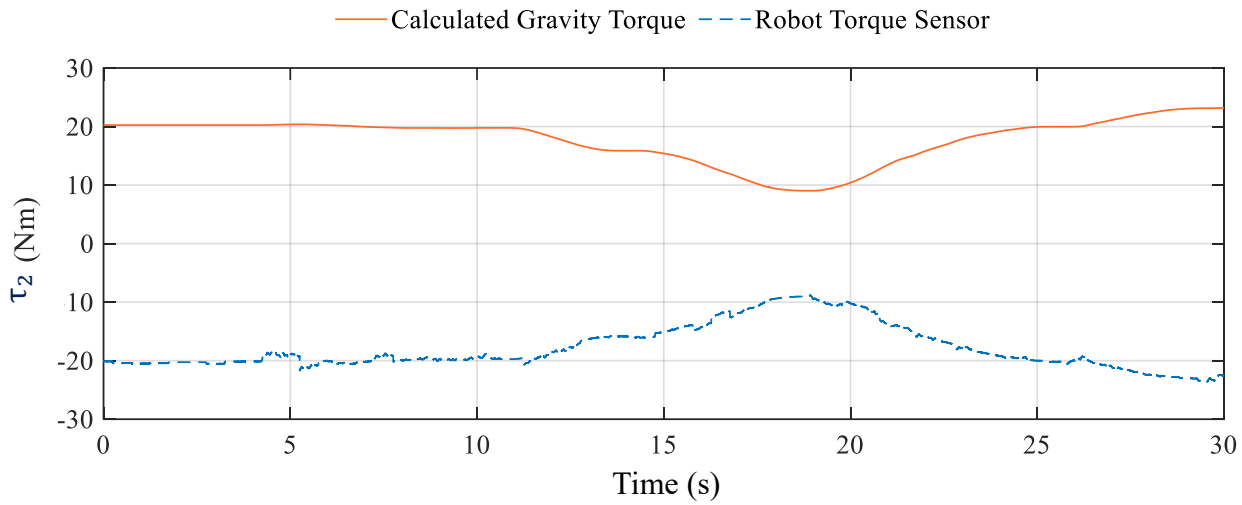

Figure 4.2: Comparison of the calculated gravity and sensor-measured torque at joint 2.

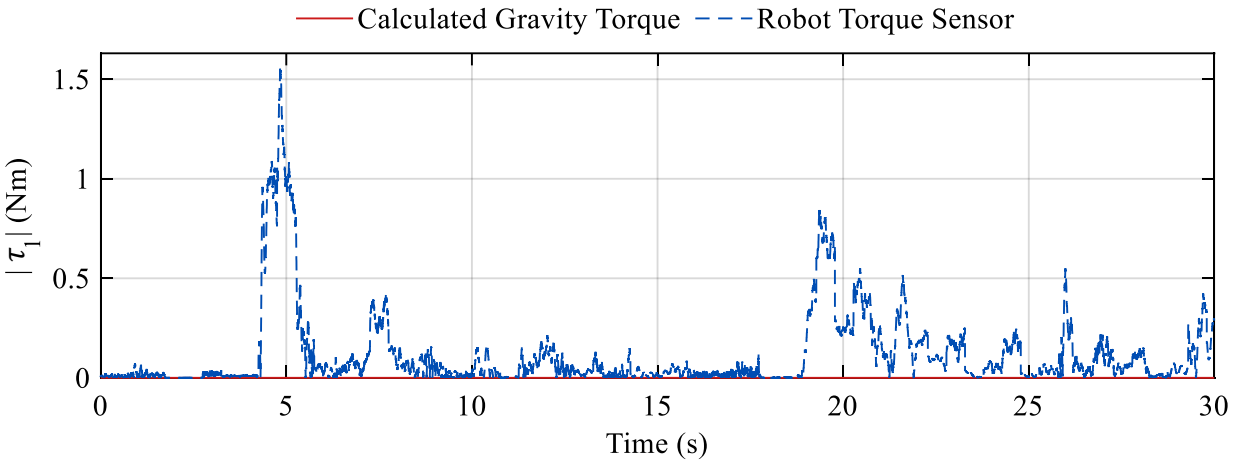

Figure 4.3: Comparison of the absolute calculated gravity and sensor-measured torque at joint 1.

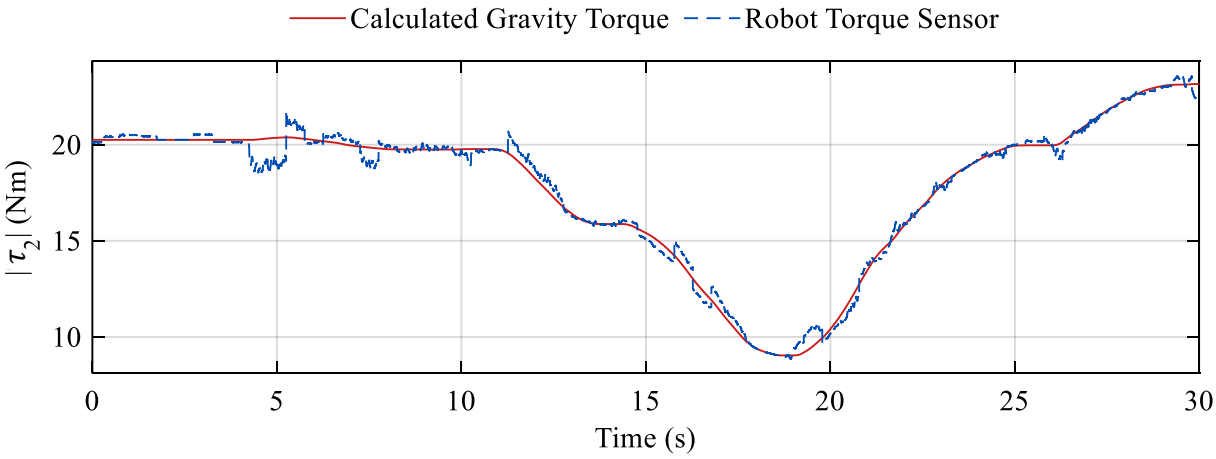

Figure 4.4: Comparison of the absolute calculated gravity and sensor-measured torque at joint 2.

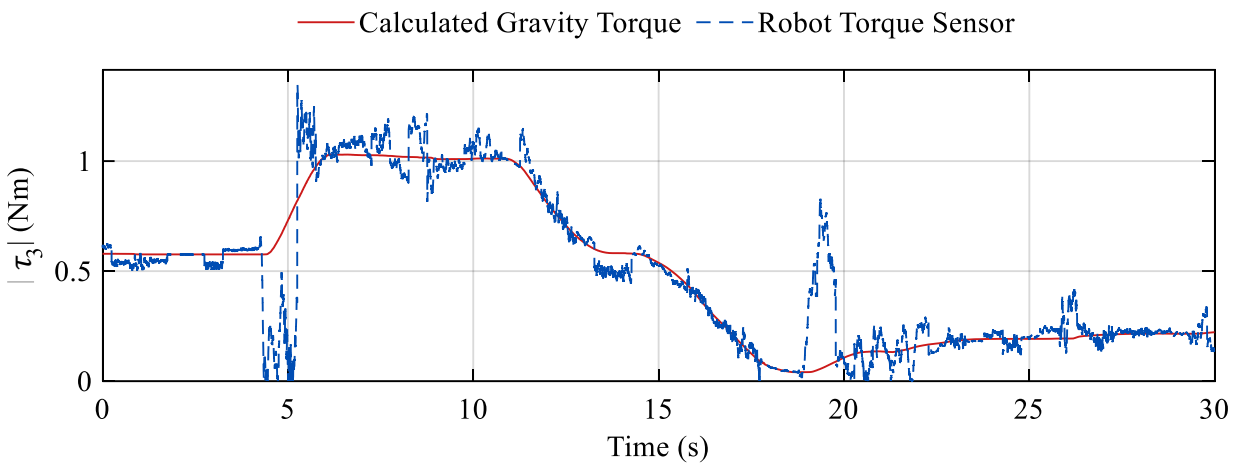

Figure 4.5: Comparison of the absolute calculated gravity and sensor-measured torque at joint 3.

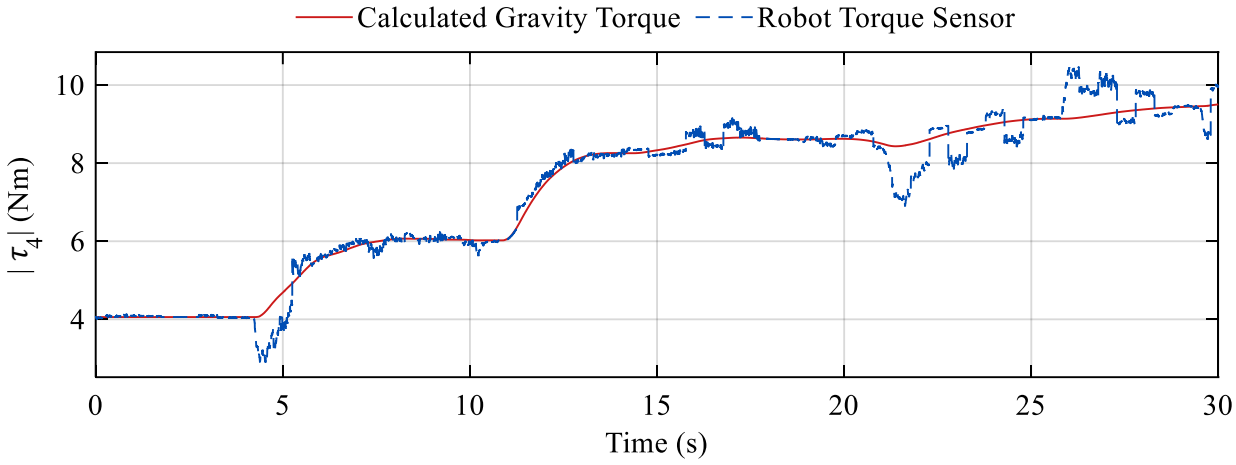

Figure 4.6: Comparison of the absolute calculated gravity and sensor-measured torque at joint 4.

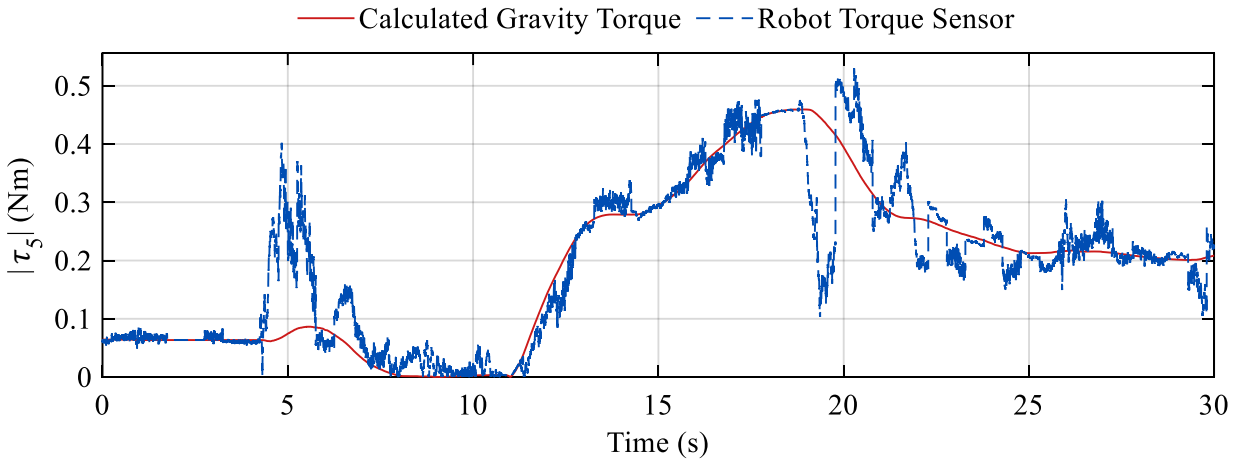

Figure 4.7: Comparison of the absolute calculated gravity and sensor-measured torque at joint 5.

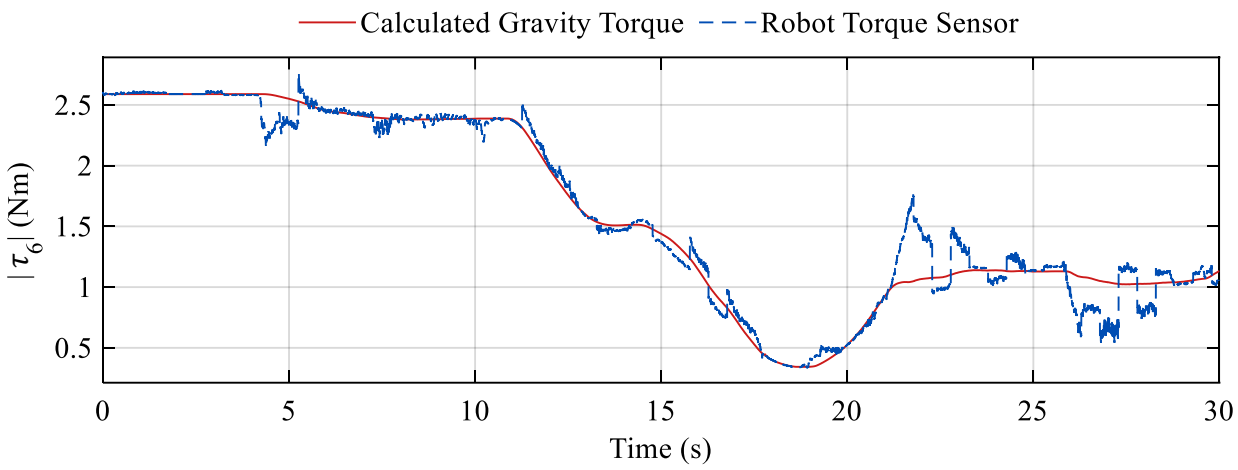

Figure 4.8: Comparison of the absolute calculated gravity and sensor-measured torque at joint 6.

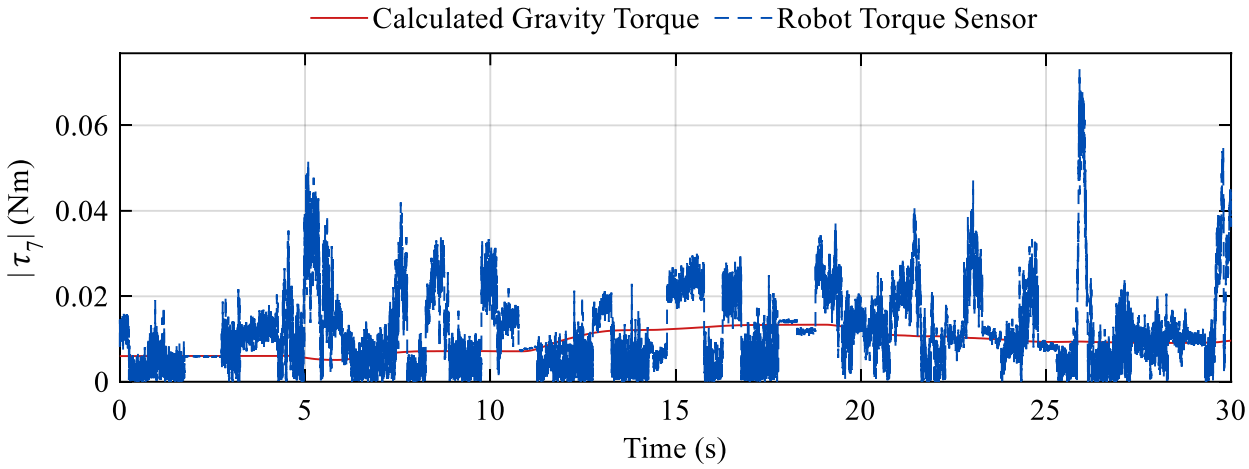

Figure 4.9: Comparison of the absolute calculated gravity and sensor-measured torque at joint 7.

To enable smooth, safe, and robust response of the robot to human forces in real-time and provide various practical rehabilitation sessions with different modes such as assistive, passive and resistive, the following general equation is used for each joint of the robot and, for each HRI mode, is slightly modified:

$$\tau_{input_i} = -\tau_{gravity_i} + k_{gain} \cdot \tau_{human_i} + \tau_{random} \quad (8)$$

where, τ_{input_i} is the torque that the robot should generate at i_{th} joint, k_{gain} is the interaction stiffness coefficient, and τ_{random} is a random torque that may be applied based on the application.

The term τ_{random} represents a randomly generated torque component intentionally introduced to make the interaction unpredictable in specific applications. It adds variability to the robot's response, preventing users from fully anticipating the motion. This unpredictability helps evaluate a user's adaptability and response during interaction. The magnitude of random torque is limited to ±20% of τ_{human_i} to ensure safe yet dynamic behavior.

4.3 Interaction Modes

Interaction modes are designed to tune the robot's behavior during HRI, particularly in rehabilitation applications. These modes determine how the robot reacts to the torques applied by the human. This ensures the interaction is adaptable, intuitive, and practical for the user's specific needs. Three primary interaction modes are implemented in this thesis: passive, resistive, and assistive. Each mode is tailored to provide different levels of assistance, resistance, or compliance based on the rehabilitation task's requirements.

4.3.1 Passive Mode

In passive mode, the control algorithm only compensates for gravity and does not resist or assist the human's motion during the interaction. When a human applies torque to the robot, it moves only due to the applied force, and the actuators do not generate any additional torque beyond gravity compensation torque. The input torque for i_{th} joint is then computed as follows:

$$\tau_{input_i} = -\tau_{gravity_i} \tag{9}$$

This mode is ideal for situations where the human is entirely in control of the motion, and the robot acts as a compliant system. For instance, during the rehabilitation exercises, the patient moves the robot end-effector freely, without the robot providing any resistance or assistance.

4.3.2 Resistive Mode

The purpose of the resistive mode is to challenge the human in playing the game. This challenge is designed to be both more exciting for humans and helpful in building strength or enhancing motor skills in rehabilitation tasks. In resistive mode, as in passive mode, gravity's torque is continuously compensated. But there is an extra term that adds stiffness based on the torque applied by the human. There are three sub-modes in the resistive mode: Gain resistive, Random resistive, and Gain-random resistive mode.

In Gain resistive mode, in addition to gravity torque compensation, the robot actively resists the human's motion by applying additional torques in opposition to the human's torque. This resistive torque is the product of the human-applied torque by k_{resist} coefficient ranging from 0 to -0.5 based on the human skill, requiring the user to exert more force to complete the task. The generated torque by the robot at the i_{th} joint in this mode is as follows:

$$\tau_{input_i} = -\tau_{gravity_i} + k_{resist} \cdot \tau_{human_i} \tag{10}$$

In Random resistive mode, the robot, along with gravity compensation, generates a random force at its end-effector. The random force is generated within a controlled range, limited to ±1.9N, making the interaction more challenging and unpredictable for the human. Due to the limited range of the generated random force at the end-effector, whose magnitude is also adjustable based on the condition of the human, the safety of the human is guaranteed. The generated torque by the robot at the i_{th} joint in this mode is as follows:

$$\tau_{input_i} = -\tau_{gravity_i} + k_{random} \cdot \tau_{gravity_i} \tag{11}$$

where, k_{random} is a random number limited to ±0.2.

Gain-random resistive mode is a combination of Gain and Random resistive modes. The generated torque by the robot at the i_{th} joint in this mode is as follows:

$$\tau_{input_i} = -\tau_{gravity_i} + k_{gain} \cdot \tau_{human_i} + k_{random} \cdot \tau_{gravity_i} \tag{12}$$

4.3.3 Assistive Mode

In assistive mode, the robot aids the human's motion by amplifying the torque applied by the human or performing the task partially autonomously. This mode ensures that the user can complete movements even with limited strength or mobility. There are three sub-modes in the resistive mode: Gain assistive, PD-based assistive, and Gain-PD-based assistive mode.

The first mode is the Gain assistive mode. The robot amplifies the torque applied by the human by a coefficient (k_{assist}) ranging from 0 to 3 based on the joint, helping the user achieve the desired motion. The generated torque by the robot at the i_{th} joint in this mode is as follows:

$$\tau_{input_i} = -\tau_{gravity_i} + k_{assist} \cdot \tau_{human_i} \tag{13}$$

The second assistive mode, PD-based assistive mode, enables the robot to assist the human in reaching the desired end-effector position using PD control of the end-effector position. In this mode, the robot guides the human's hand toward the target position, as a physiotherapist would during rehabilitation. The robot remains adaptive to the torques applied by the human, allowing the robot and the human to collaborate in playing the game. The generated torque by the robot at the i_{th} joint in this mode is as follows:

$$\tau_{input_i} = -\tau_{gravity_i} + \tau_{PD_i} \tag{14}$$

$$\tau_{PD_i} = \alpha_{PD} \, . \, \tau_{gravity_i} \tag{15}$$

$$\alpha_{PD} = sat\left(\frac{(k_P \, . e + k_D \, . \dot{e})}{e_{max}}\right), \qquad -1 \leq \alpha_{PD} \leq +1 \tag{16}$$

where, τ_{PD} is the torque generated by the robot to minimize the paddle and the ball's y position error, α_{PD} is a dimensionless PD-based scaling factor, e is the paddle and the ball's y position normalized error, $\dot{e}$ is a derivative of the paddle and ball's y position normalized error, and e_{max} is the maximum possible error.

The final assistive mode is the Gain-PD-based assistive mode. In this mode, the robot integrates both Gain assistive mode and PD assistive modes to provide a higher level of support during gameplay. In this mode, the robot simultaneously guides the user's hand to the desired end-effector position using PD control while amplifying the force the user applies to the end-effector. This results in even higher assistance for the human while playing the game. The generated torque by the robot at the i_{th} joint in this mode is as follows:

$$\tau_{input_i} = -\tau_{gravity_i} + k_{assist} \, . \tau_{human_i} + \tau_{PD_i} \tag{17}$$

4.4 Summary

This chapter presented the control architecture for enabling safe, precise, and adaptive HRI using a 7 DoF robotic manipulator. Torque and position sensors provide continuous feedback at 1000 Hz, allowing the system to distinguish between gravity-applied and human-applied torques, ensuring compliant and responsive interaction. The architecture integrates PD-based end-effector position control for assistive movement and direct joint position control of all seven joints for real-

time responsiveness and adaptability to externally applied human forces. Additionally, a gain factor for each joint is added to the control algorithm to enable the design of each interaction mode.

To support different rehabilitation and interaction objectives, the proposed control framework implements three interaction modes: passive, resistive, and assistive. In passive mode, gravity-induced joint torques are compensated, allowing free and user-controlled motion. In resistive mode, gravity compensation is combined with torque terms that oppose the user's applied torque through gain-based, random, or gain-random resistance, increasing task difficulty and promoting strength and motor control. In assistive mode, the robot supports motion by amplifying human-applied joint torque or by modulating gravity-induced torques using a PD-based scaling factor derived from end-effector position error. Together, these modes enable smooth transitions between free motion, resistance, and adaptive assistance in rehabilitation and gameplay scenarios.

5 Computer Vision

5.1 Overview

This chapter introduces computer vision, which enables machines to interpret and analyze visual data such as images and videos. Computer vision mimics the human visual system using sensors, algorithms, and AI models. It supports many applications, including image recognition, object detection, and video tracking. In rehabilitation, it enables the analysis of human posture and movement.

The primary objective of using vision technology in this study is to accurately capture the Human-robot Interaction (HRI) scene. This enables the extraction of essential data for real-time visualization of human-robot configurations in a shared environment. The collected data can also be archived for systematic and comprehensive analysis.

The system designed for this project employs an RGB-D camera, specifically the RealSense 455f, the details are explained in [57]. The camera is mounted 2 meters from the robot, facing it, shown in Figure 5.1. It can capture a scene of interaction between a human and a robot. Throughout this interaction, the camera and the robot's base remain stationary.

The camera coordinate frame $\{C\}$ is defined such that the depth direction corresponds to the Y_c-axis, while the remaining axes follow the right-hand rule. This camera frame convention is used consistently throughout this chapter.

The RGB-D camera data is similar to human visual input but is processed differently. While the human brain has evolved over millions of years to interpret visual information effectively, this system aims to develop comparable vision capabilities. Processing vision data in real-time, as the human brain does, is highly computationally intensive and impractical, especially when the goal is to extract only the data needed for real-time visualization. Therefore, the developed system focuses on essential information to enable real-time 3D visualization of the robot and the human. To achieve this, the system relies on pre-trained AI models, custom-designed filters, and a high-end GPU to extract only the most critical insights from the vision data.

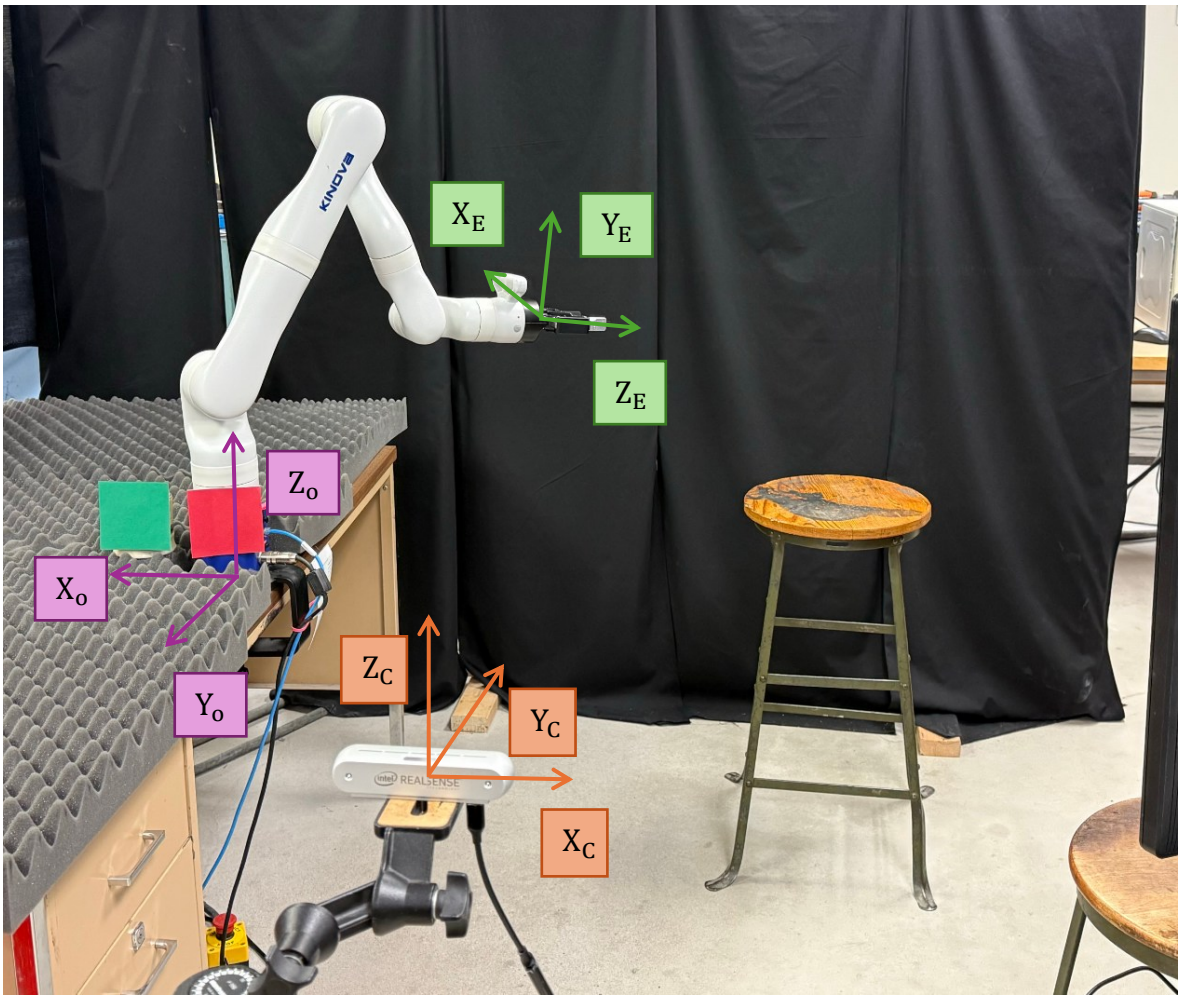

Figure 5.1: Camera and robot configuration relative to each other.

5.2 Robot Simulation

To simulate the stationary manipulator robot in real time, a fixed base position is first established. This position defines the reference frame for the robot used in this study. By detecting the robot's base once via vision, it eliminates the need for continuous vision-based tracking. Once the 3D position of the robot's base relative to the camera is determined, a coordinate system is defined at that location within the 3D environment. The robot's model, described in URDF (Unified Robot Description Format), is then positioned within this coordinate system. During simulation, the robot model is continuously updated using the angular position sensors at each joint, eliminating the need for vision.

To establish the robot base coordinate frame, the X_0 axis is determined first. The Z_0 axis is then defined. The Y_0 axis is obtained as the vector orthogonal to both axes. To extract these axes with minimal computation, two colored 10 cm square papers are placed near the robot base, as shown in Figure 5.2. The red paper is attached directly to the robot base, while the green paper is placed immediately beside it, with both squares facing the camera.

From the camera's viewpoint, the centers of the two papers lie on a horizontal line. This line is parallel to the true X_0 axis of the robot base, as indicated by the orange axis in Figure 5.2(b). Because both markers lie on the same planar surface facing the camera, the projected line connecting their centers accurately represents the horizontal direction of the robot base. The horizontal line connecting the paper centers (dark-blue line in Figure 5.2(b)) provides the direction of the robot's X_o axis, but is vertically offset from the true base axis by 10 cm. This offset corresponds to the vertical distance between the paper's connection line and the robot base origin, illustrated by the light-blue arrow. The robot's X_0 axis is obtained by connecting the centers of the red and green papers. This line is then translated downward by 10 cm along the camera Y-axis, as shown in Figure 5.2(a).

When building the Z_o axis of the robot, it is assumed that the camera is mounted such that its Y axis is aligned with the gravity direction. Under this assumption, the robot's base Z_o axis is defined as vertical and passes through the center of the red paper. The origin of the robot base coordinate system is determined at a point located 10 cm below the center of the red paper along the vertical direction. The resulting vertical axis is shown as the orange arrow representing the Z_o axis in Figure 5.2(b). Finally, the Y_o axis is defined as the vector perpendicular to both the X_o and Z_o axes, pointing inward in the image. This resulting coordinate frame represents the robot's base coordinate system.

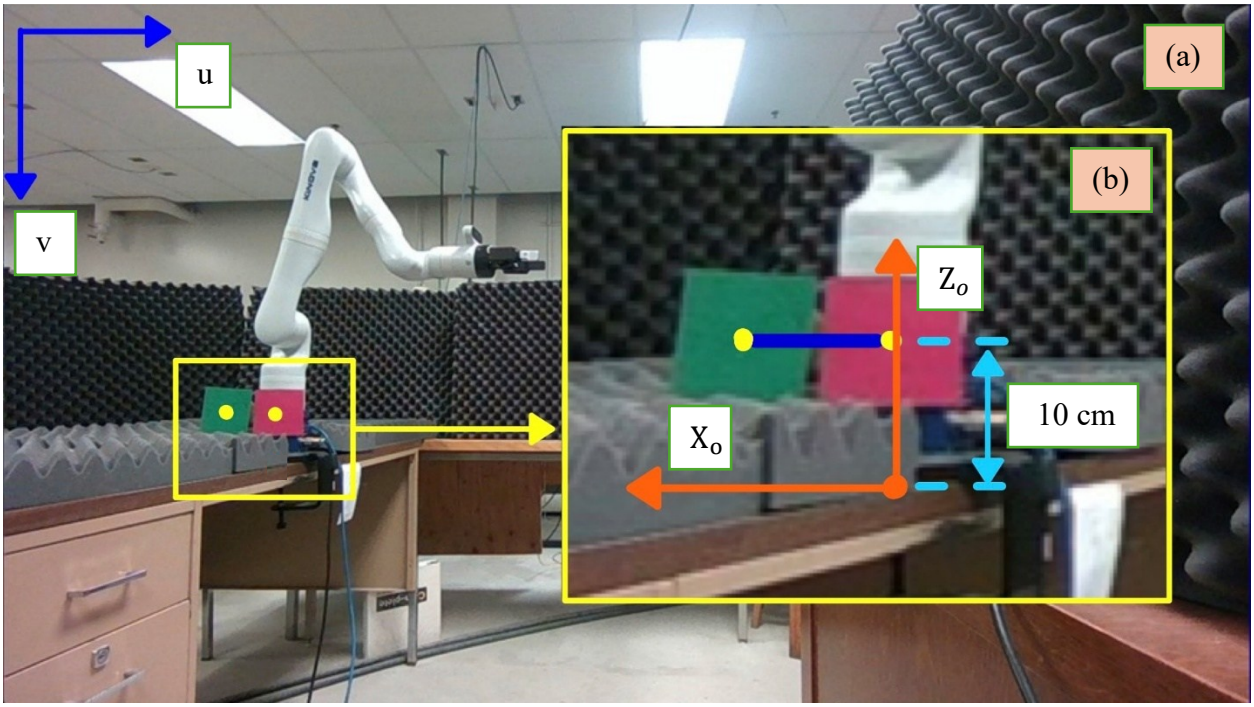

Figure 5.2: Camera view capturing green and red sheets to establish the 3D base coordinate system.

After defining the robot's base coordinate system, the next step is to create the same coordinate framework in the visualization environment. Once established, the robot's Unified Robot Description Format (URDF) model is loaded into this coordinate system to ensure proper alignment and reference. It is important to note that, in the simulation environment, the global coordinate system is inherently aligned with the camera's coordinate system, serving as a universal reference frame for all objects and interactions. The final visualization of the robot model within this environment is illustrated in Figure 5.3.

During interaction, the robot is simulated using angular position sensors embedded in each joint. These sensors transmit the robot's joint positions at a frequency of 1000 Hz. This data is employed to update the robot model, thereby facilitating a real-time 3D simulation of the robot's configuration.

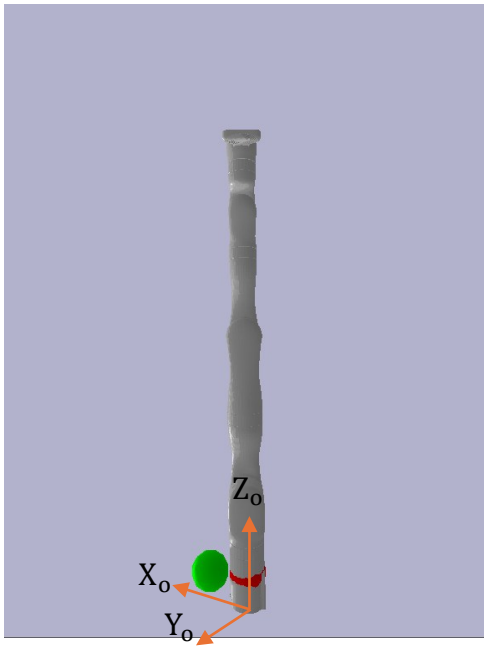

Figure 5.3: The robot URDF in PyBullet.

5.3 Human Posture Visualization

At the beginning of each interaction, a human body kinematic model is generated. The model is then updated in real time using detected body key points. This approach is essential to create a personalized human model, since everyone has unique body measurements, and using a single universal model would significantly decrease accuracy. Once established, the model continues to update with visual data.

5.3.1 Pose Detection

In HRI, pre-trained models for pose detection enable real-time, accurate tracking of human movements. These models identify keypoints throughout the body from RGB input, and when combined with aligned depth data, the 3D position of each key point can be derived. The selected models must support rapid and precise joint detection during dynamic interactions. Efficiency, licensing, and compatibility are also considered in the model selection. Several models, including detectron2, OpenPose, and PoseNet, were tested, with YOLOv8n-Pose and MediaPipe Hands emerging as top options. YOLOv8n-Pose offers fast, efficient full-body tracking on GPUs, though it cannot detect hand keypoints. MediaPipe Hands complements this by providing detailed hand gesture data. Together, these models offer comprehensive coverage from general movement to fine hand gestures, supporting advanced HRI capabilities.

As shown in Figure 3.3, YOLOv8n-Pose detects 17 body joints, including ankles, knees, hips, spine, neck, elbows, wrists, nose, and head tips. On high-performance GPUs such as the RTX 4090, the model operates at approximately 90 Hz. The detected joints are marked by green dots. MediaPipe Hands detects hand keypoints, including the tips of the fingers, the bases, and the palm center, at around 25 Hz on CPUs, with detected joints marked by red dots in Figure 5.4, enabling analysis of gestures and movements.

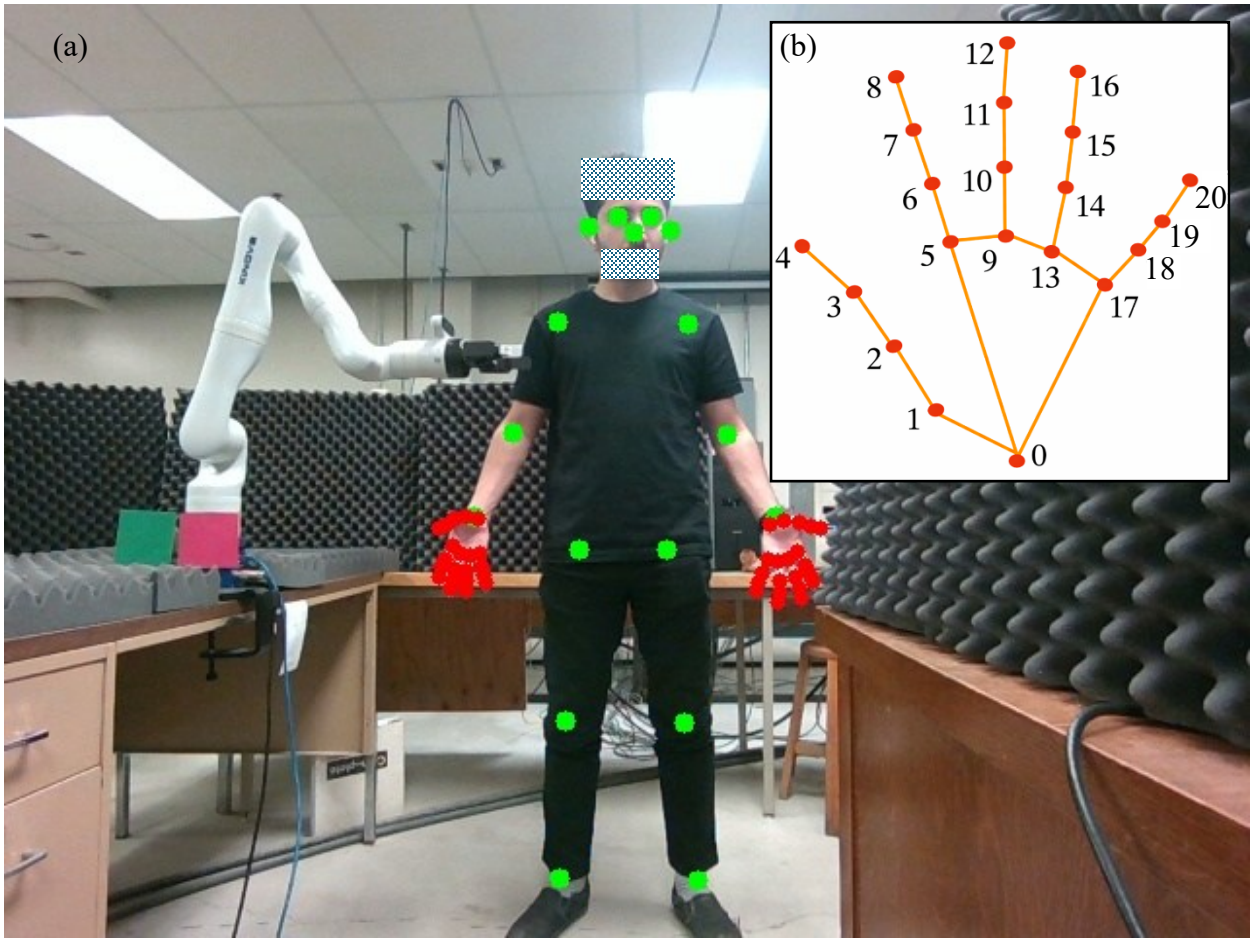

Figure 5.4: Human body keypoints detection by YOLOv8n-Pose and MediaPipe Hands models.

Table 1: Hand keypoint indices and corresponding anatomical joint names in Figure 5.4. MCP: Metacarpophalangeal joint; PIP: Proximal interphalangeal joint; DIP: Distal interphalangeal joint; IP: Interphalangeal joint.

Index	Keypoint Name	Index	Keypoint Name
0	Wrist	11	Middle Finger DIP
1	Thumb CMC	12	Middle Finger Tip
2	Thumb MCP	13	Ring Finger MCP
3	Thumb IP	14	Ring Finger PIP
4	Thumb Tip	15	Ring Finger DIP
5	Index Finger MCP	16	Ring Finger Tip
6	Index Finger PIP	17	Pinky MCP
7	Index Finger DIP	18	Pinky PIP
8	Index Finger Tip	19	Pinky DIP
9	Middle Finger MCP	20	Pinky Tip
10	Middle Finger PIP		

5.3.2 Pose visualization

The camera can provide RGB frames at up to 90 fps with a resolution of 1280 × 720 pixels, and depth frames at 1280 × 800 pixels at 30 fps. The depth frame range is from 0.6 m to 6 m with an accuracy of <2% at 4 m. To achieve a good balance between computational cost and accuracy, the camera is located approximately 1.5 m from the robot, and both frames are set to 640 × 480 pixels at 30 fps.

To construct the human body kinematic model, the subject is first positioned facing the camera. This placement minimizes joint occlusion and depth inaccuracies before the interaction session begins (see Figure 5.5 (a)). Then, the 2D positions of the body keypoints are detected in the RGB frame. Next, the corresponding depth values are extracted from the depth frame to recover the 3D joint positions. This initial data is used to construct the human kinematic model. A 3D human model tailored to the person interacting with the robot is then developed (see Figure 5.6). This personalized model helps reduce simulation errors when the human is close to the robot. After creating the 3D kinematic model of the human body, the next step is to continuously update it

within the simulation environment. This kinematic model dynamically updates in real-time using camera data and pose detection models to accurately reflect the human's live posture and position relative to the robot. In the following figures, the camera's point of view of the HRI scene is depicted as RGB frame data, and the human model is updated accordingly.

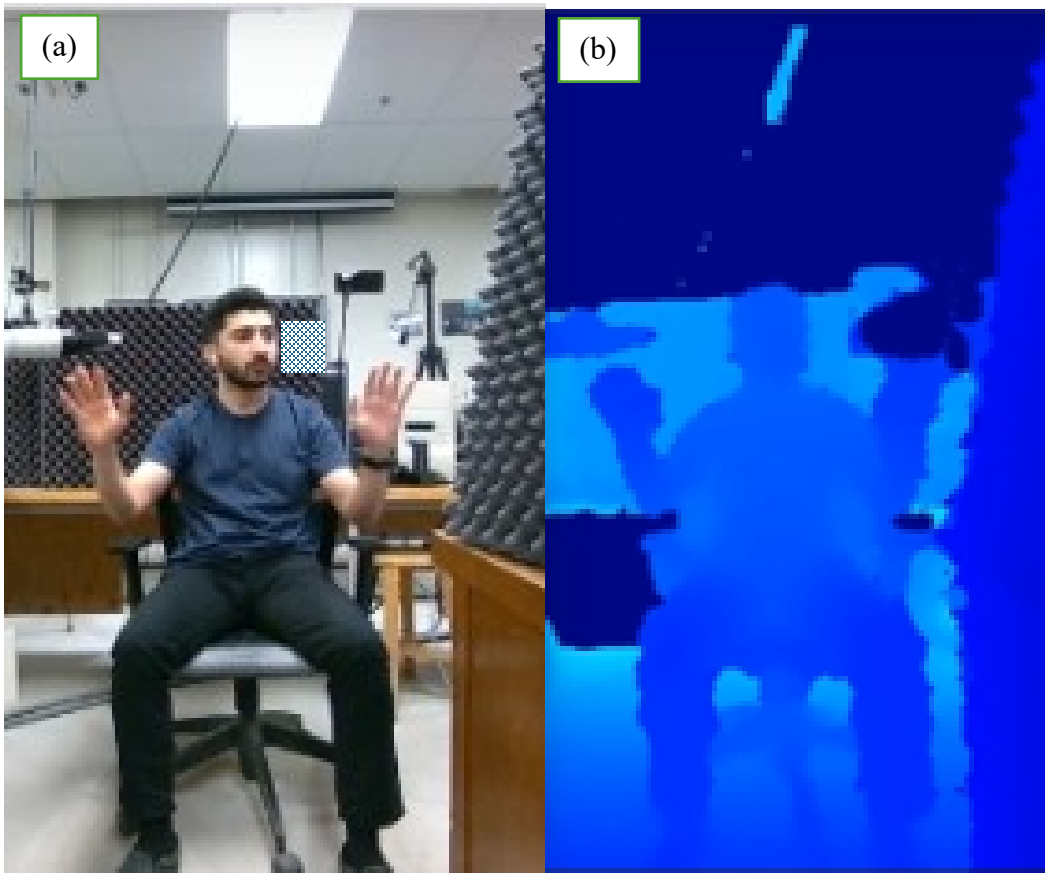

Figure 5.5: (a) Cropped RGB image of the camera capturing a human; (b) Corresponding depth image aligned with RGB frame.

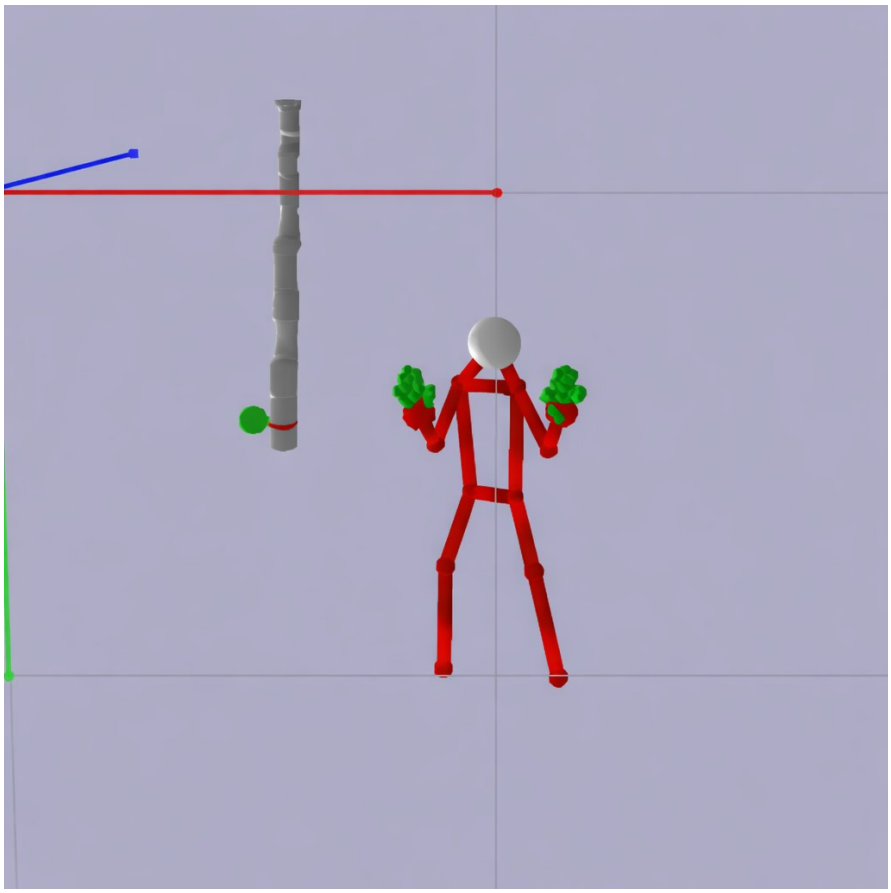

Figure 5.6: Human model updates based on the camera frame in the simulation (enhanced).

5.4 Data Processing

Updating the human body's keypoints using an RGB-D camera comes with challenges. As explained earlier, the human keypoints are detected as pixel positions in the RGB frame, these positions are then used to derive the depth values for the corresponding keypoints. During this stage, where the depth values should be derived, significant challenges arise.

Firstly, the depth frame often contains noise. This noise can lead to incorrect depth estimates, thereby affecting overall 3D positioning accuracy. Secondly, the RGB and depth frames are not always perfectly aligned. These result in a false depth value for keypoints near the body's boundary, such as shoulders, elbows, wrists, and, specifically, the hand keypoints. These challenges require robust processing techniques to improve 3D keypoints localization accuracy, such as smoothing noise in depth data and detecting false depth values.

5.4.1 Denoising and False Data Reduction

RGB-D sensors often suffer from noise and misalignment. As a result, the depth value corresponding to each detected keypoint can sometimes be unreliable. This issue is more significant for hand keypoints because each finger occupies only a small area in the depth frame. To mitigate this, a multi-step depth filtering pipeline was designed, as shown in Figure 5.7.

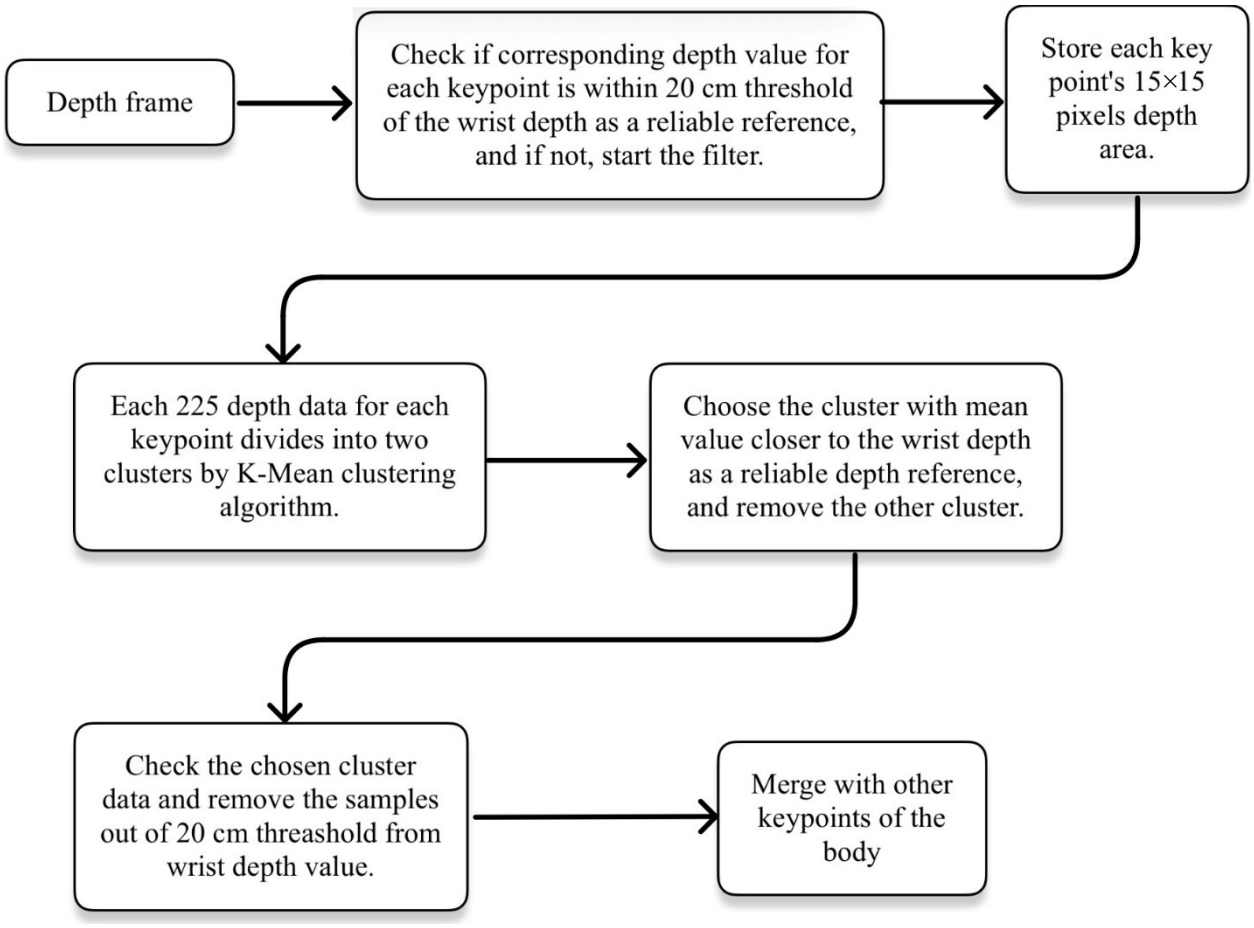

Figure 5.7: Flowchart diagram of the hand depth filtering pipeline.

The process starts with detecting keypoints in each frame. Only the right-hand index finger tip keypoint is processed, shown in Figure 5.8. The red point represents the keypoint detected in the RGB frame in Figure 5.8 (a) and its corresponding depth value in Figure 5.8 (b), which is 220 cm. In contrast, the wrist (yellow point in the RGB frame) has a depth value of 166 cm. This indicates that the depth value for the finger keypoint is invalid and should be replaced with a more reliable value. After passing the initial detected value through the filter, the desired reliable value

is obtained, shown as the green point. Orange X and Y coordinate systems in Figure 5.8 (a), andFigure 5.8 (b) show the indexing coordinate of each frame.

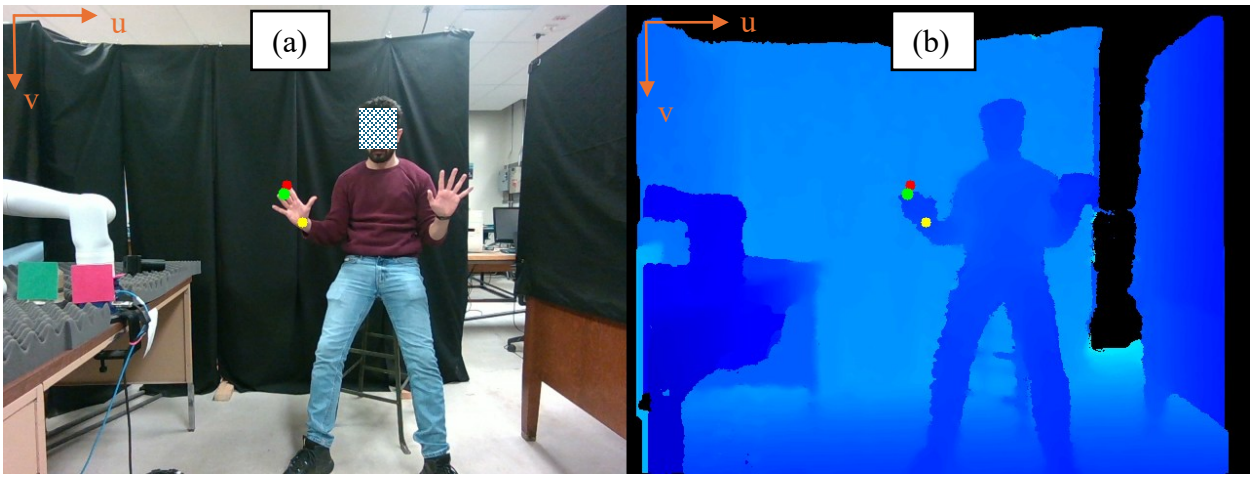

Figure 5.8: (a) RGB frame, and (b) Corresponding depth frame used as inputs to the system.

To obtain the correct depth value for the detected keypoint, initially, a local 15×15 pixel region is extracted from the aligned depth frame (see Figure 5.9). This region provides richer depth context for the estimated 2D keypoints. The red point in the figure indicates the initial incorrect depth. Upon closer inspection of the frame, the correct value should be in the darker blue region, which is the actual depth values representing the finder tip. To facilitate this, K-means clustering is used.

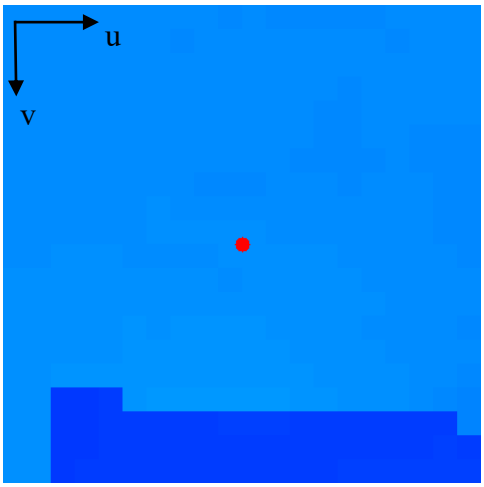

Figure 5.9: The 15×15 pixel neighborhood of the detected keypoint in the depth frame.

K-means clustering is an unsupervised learning method that partitions data into K distinct groups based on feature similarities. The algorithm iteratively assigns each data point to the cluster with the closest centroid. This process minimizes cluster variance and improves separation between clusters. This algorithm operates in two main steps: “Assignment” and “Update”. In the Assignment step, each data point is assigned to the nearest cluster center based on a distance metric, typically the Euclidean distance. Mathematically, this can be expressed as:

$$C_i = \arg\min_k \| x_i - \mu_k \|^2 \tag{18}$$

where C_i is the index of the cluster assigned to the data point x_i, K is the total number of clusters, x_i is the feature vector of the i_{th} data point, and μ_k is the centroid (mean position) of the k_{th} cluster.

In the Update step, after all data points have been assigned, the centroids are recalculated as the mean of the data points in each cluster. This is formulated as:

$$\mu_k = \frac{1}{N_k} \sum_{i \in C_k} x_i \tag{19}$$

where N_k is the number of data points assigned to cluster k, and x_i is the data point belonging to the cluster k,

These two steps are repeated iteratively until the centroids converge, meaning the change in their positions between iterations becomes negligible or the algorithm reaches a predefined number of iterations. The objective function minimized by the K-means algorithm is given as:

$$J = \sum_{k=i}^{k} \sum_{i \in C_k} \| x_i - \mu_k \|^2 \tag{20}$$

Where J is the total within-cluster sum of squared distances (clustering cost), and x_i is a data point in cluster k,

Through iterative optimization, K-means groups the dataset into compact and well-separated clusters. The result is illustrated in Figure 5.10. The red crosses represent the background, and the hollow green circles represent the hand depth values. This step helps distinguish background depth or reflections from the actual depth values associated with the

detected keypoints. The cluster closer in depth to the wrist keypoint, assumed to be more reliable, is chosen as the reference cluster for that keypoint. To enhance robustness, the selected cluster is compared to the wrist depth. If any data point differs from the wrist depth by more than 20 cm, that point is removed from the cluster. The remaining values are each averaged and used as the chosen depth value. The new value is 157 cm, shown as a solid green circle in Figure 5.10. Compared with the wrist depth of 166 cm, this value is considered acceptable.

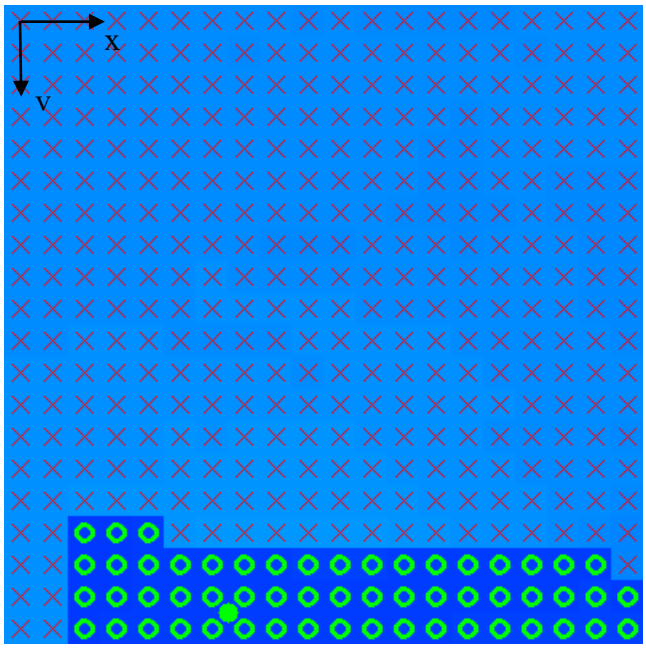

Figure 5.10: Clustered pixels, shown with green circles and red crosses.

In Figure 5.11, the initial detected depth for the detected keypoint is represented by a red point. Subsequently, this red point is replaced by a green point indicating the computed, more accurate depth value. The green point provides a refined and reliable estimate of the depth at that specific keypoint, improving the overall accuracy of the system.

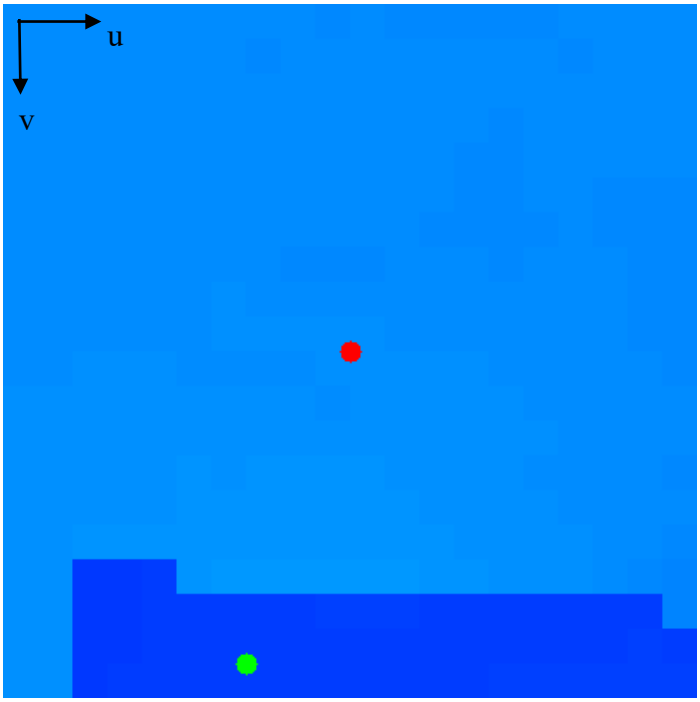

Figure 5.11: The red point indicates the initially chosen depth, and the green point indicates the derived depth via the custom filter.

Plots of this process are shown Figures 5.12 to 5.14. The wrist is shown as a yellow marker, while the red cross indicates the initial depth measurement at that pixel. A square region of depth pixels surrounding this location is extracted and visualized. The hollow red and green circles combined create the neighborhood values around the keypoint.

As shown, the neighborhood contains two distinct depth clusters: a dense cluster of outlier background pixels (hollow red circles) and a compact cluster corresponding to the actual finger keypoint (hollow green circles). These two groups are produced by K-means clustering, which separates the regions automatically. This allows the algorithm to discard the background cluster and retain only the foreground finger surface. The final computed depth, representing the mean of the correct filtered cluster, is marked with a green cross. This approach removes noisy depth samples caused by pixel misalignment between the RGB and depth streams and by background leakage, producing a stable and accurate depth estimate for the wrist joint.

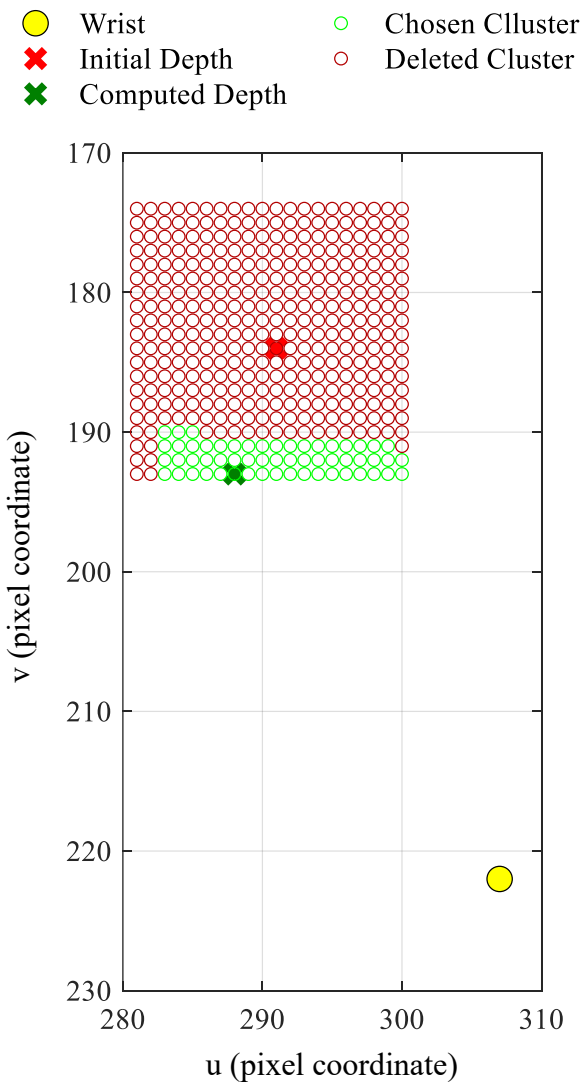

Figure 5.12: u–v image-plane of the Region of Interest (ROI) depth values, initial depth, computed depth, and the wrist.

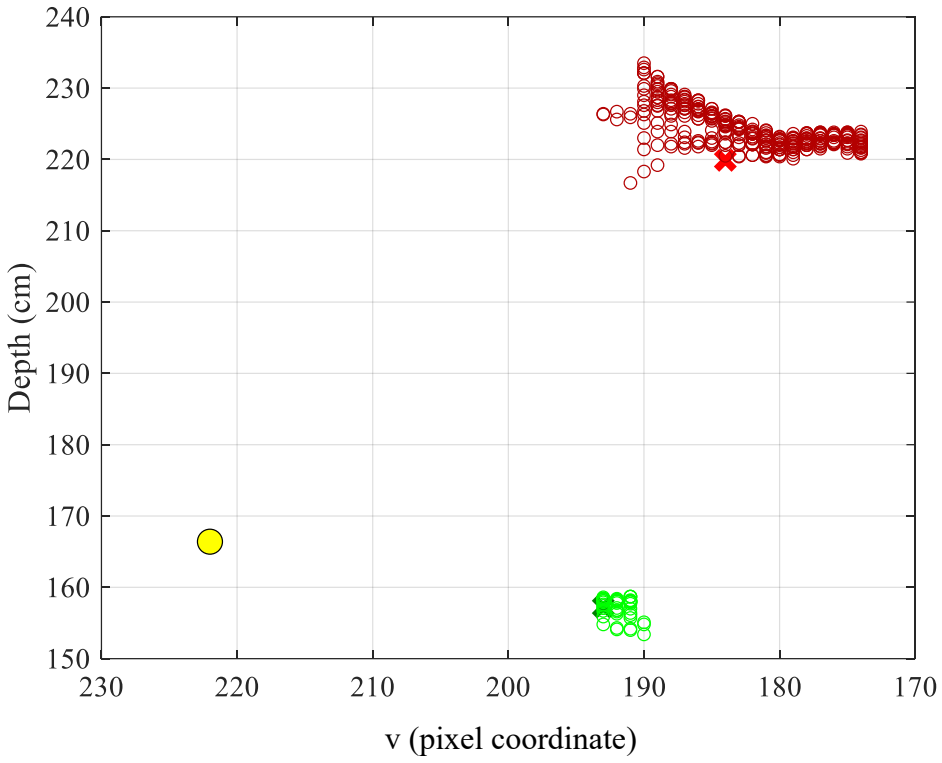

Figure 5.13: v-Depth view of the ROI depth values, initial depth, computed depth, and the wrist.

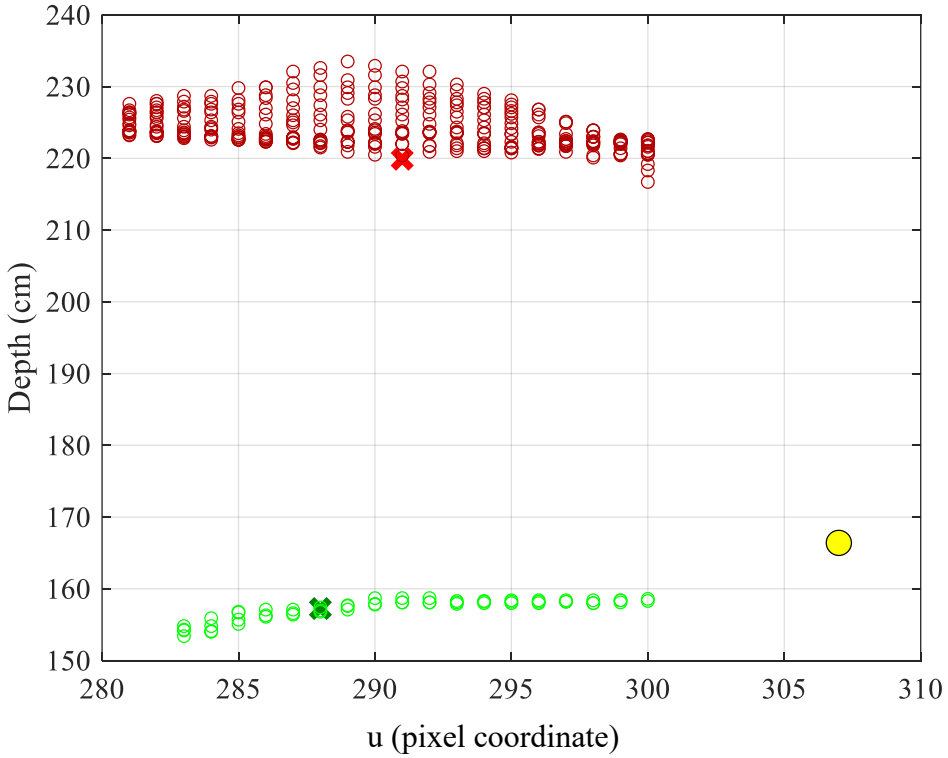

Figure 5.14: u-Depth view of the ROI depth values, initial depth, computed depth, and the wrist.

5.4.2 Body Keypoints Denoising

A Kalman filter is applied to further smooth fluctuations and improve stability across the body keypoints. These refined depth values are used to update the human body kinematic model, forming a consistent, noise-resilient 3D model ready for real-time simulation. Depth sensors, such as those used in 3D cameras, often produce distorted measurements. These distortions may be caused by sensor noise, object distance, or environmental conditions. The Kalman filter performs well in these situations by continually updating estimates based on the latest measurements, thereby improving the system's outputs step by step. The Kalman filter works through two main phases: prediction and update.

Prediction has two steps: “State” prediction and “Covariance” prediction. In State prediction, the filter predicts the next state of the depth values based on the previous estimates. This is formulated as:

$$x_{k|k-1} = Ax_{k-1|k-1} \tag{21}$$

Where $x_{k|k-1}$ is the predicted depth estimate, $x_{k-1|k-1}$is previous depth, and A is the state transition matrix.

In the covariance prediction step, the uncertainty of the prediction is updated. The uncertainty of the previous estimate is propagated through the system model, and process noise is added to account for model imperfections, as shown below:

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q \tag{22}$$

where $P_{k|k-1}$ is the predicted error covariance matrix, $P_{k-1|k-1}$ is covariance from the previous step, A^T is the transpose of the State transition matrix, and Q is denotes the process noise covariance matrix.

In addition to the prediction phase, the update phase has two steps: “Measurement” update and “Covariance” update. In the Measurement update, the filter adjusts the prediction in Equation 21. The Kalman Gain K_k determines the optimal weighting between the model prediction and the measurement. The denominator term $(HP_{k|k-1}H^T + R)^{-1}$ represents total expected uncertainty in the measurement, and its inverse ensures that the gain decreases when the measurement noise R is high.

$$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1} \quad (23)$$

where K_k is the Kalman gain matrix, H is an observation matrix, and R is the measurement noise covariance matrix.

Equation 24 corrects the predicted state by adding a weighted innovation term, the difference between the actual and predicted measurements. This correction ensures that the updated estimate optimally combines both the model prediction and the new measurement.

$$x_{k|k} = x_{k|k-1} + K_k(z_k - Hx_{k|k-1}) \quad (24)$$

where $x_{k|k}$ is the updated state estimate, $x_{k|k-1}$ is the predicted state estimate, and z_k is a new measurement.

During the "Covariance" update, the estimated state error is reduced, improving confidence in the measurement and, in turn, the updated estimate. The formulation is as follows:

$$P_{k|k} = (I - K_kH)P_{k|k-1} \quad (25)$$

where $P_{k|k}$ is updated error covariance, I is the identity matrix, and $P_{k|k-1}$ is predicted error covariance matrix.

Figures 5.15 to 5.20 show the wrist joint's X-index, Y-index, and depth value before and after filtering across a sequence of frames. The wrist location is indicated by a yellow point in Figure 5.8 (a). Raw measurements (red curves) show fluctuations and spikes due to noise and sensor inconsistencies. Filtering (blue curves) smooths these measurements over about 430 frames. The Kalman filter reduces high-frequency noise while preserving genuine movements. As a result, stable trajectories are produced that reflect the hand's drift and oscillations. In Figures 5.15, 5.17, and 5.19 no values between frame 0 and 100 are due to the human not being in front of the camera in the recorded data.

Figure 5.17 and Figure 5.18 illustrate these oscillations, showing the wrist moving up and down in a manner that mimics participant hand movements during the HRI session. Additionally, the noise characteristics differ between the X and Y indices and the depth data. The X and Y indices are affected by inconsistencies in pixel coordinate detection, while the depth data is influenced by depth frame inaccuracies. Consequently, the coefficients used for the X and Y indices differ from those used for depth, as detailed in Zoomed in view of Figure 5.19, showing the effect of the Kalman filter on the wrist Z (depth value). Table 2 shows the Kalman Filter parameter values.

Overall, the implementation of Kalman filters has significantly improved the stability and reliability of depth data.

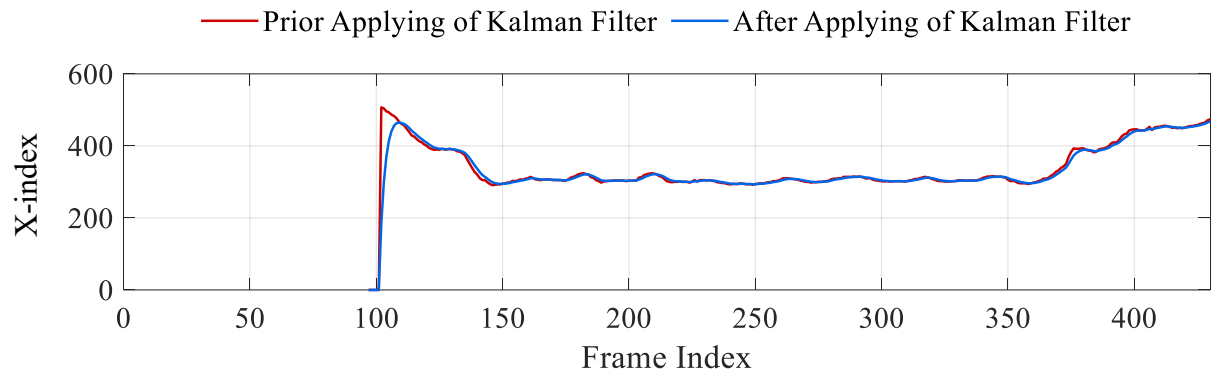

Figure 5.15: Effect of the Kalman filter on the wrist X-index over continuous frames.

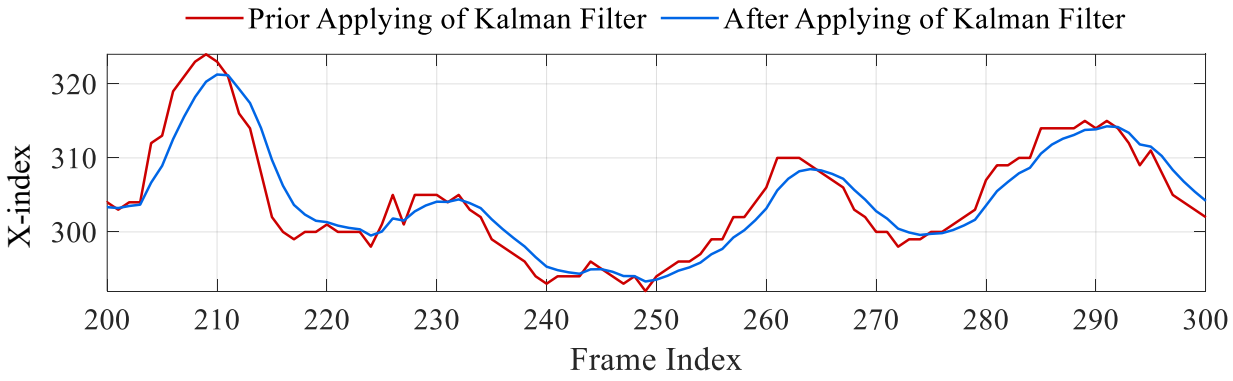

Figure 5.16: Zoomed in view of Figure 5.15, showing the effect of the Kalman filter on the wrist X-index.

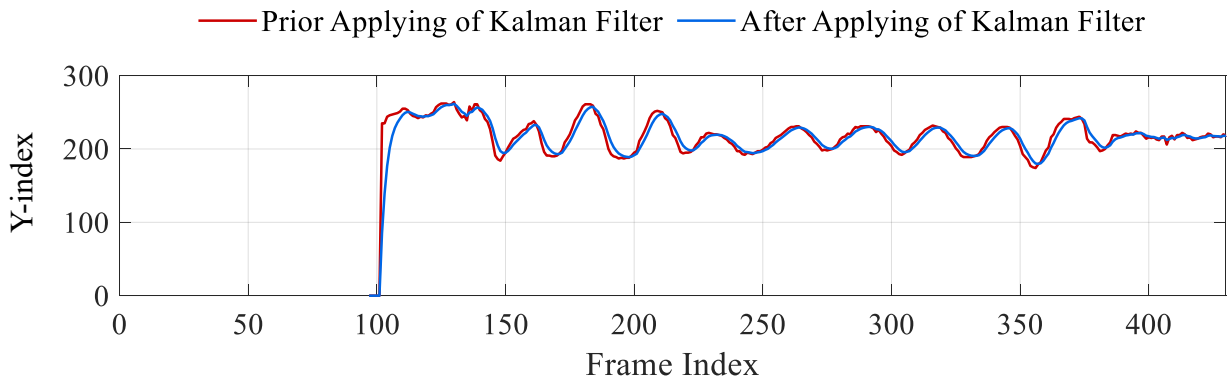

Figure 5.17: Effect of the Kalman filter on the wrist Y-index over continuous frames.

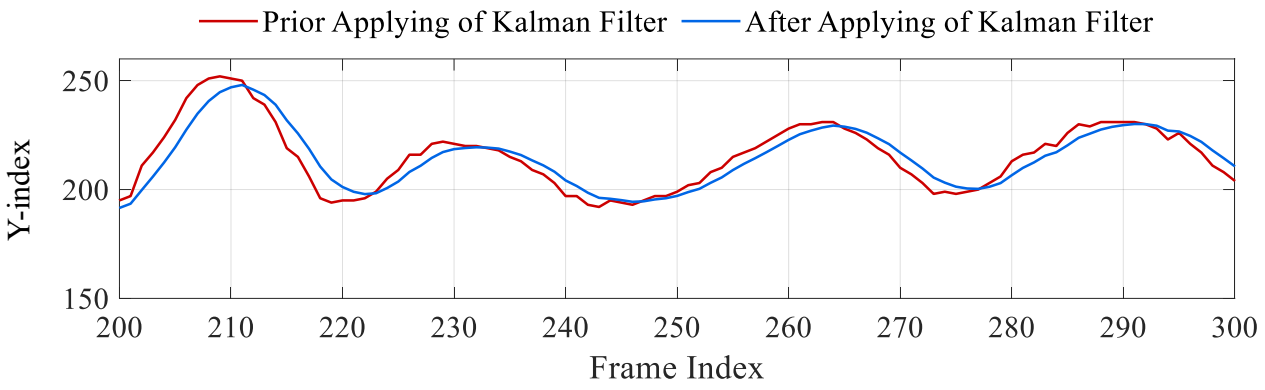

Figure 5.18: Zoomed in view of Figure 5.17, showing the effect of the Kalman filter on the wrist Y-index.

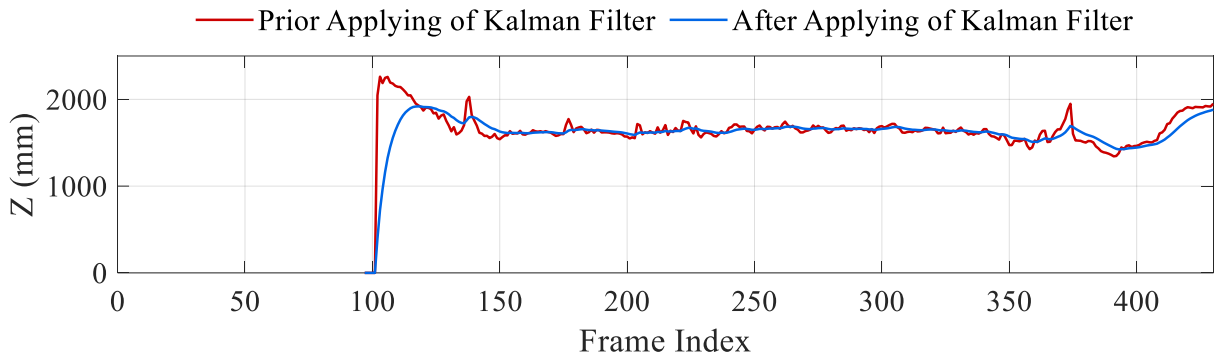

Figure 5.19: Effect of Kalman filter on the wrist depth.

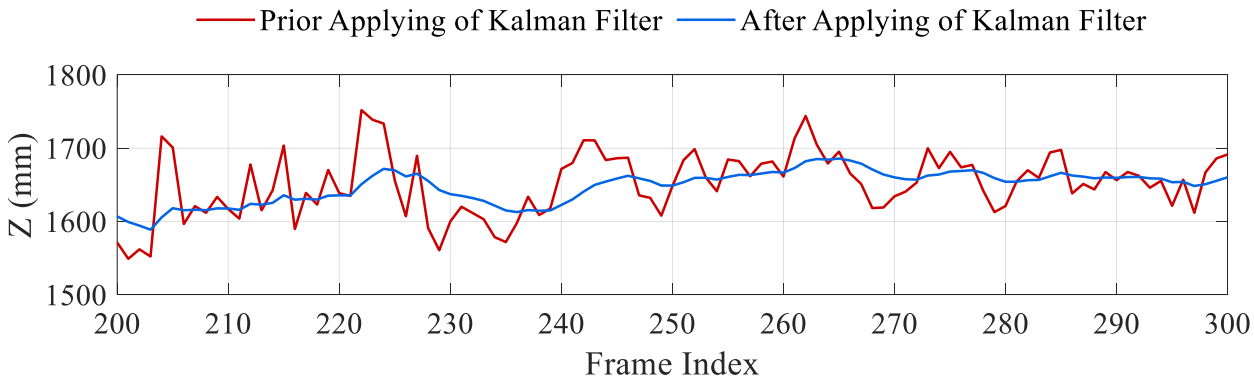

Figure 5.20: Zoomed in view of Figure 5.19, showing the effect of the Kalman filter on the wrist Z (depth value).

Table 2: Kalman Filter Parameters.

	X	Y	Depth
Q (Process Noise)	0.01	0.01	0.002
R (Measurement Noise)	0.05	0.05	0.1

5.5 Reconstruction of Body Keypoints

To obtain the 3D spatial position of each detected body joint, the 2D pixel coordinates (X-index, Y-index) provided by the vision system are first used. These coordinates are transformed into 3D coordinates in the camera reference frame using the camera intrinsics, including the focal lengths $f_x, f_y$and the principal point offsets c_x, c_y. Each 2D keypoint is then converted into metric coordinates using standard pinhole camera back-projection:

$$X_C = \left(X_{frame-index} - c_x \right) \frac{Z}{f_x} \tag{26}$$

$$Z_C = (Y_{frame-index} - c_y) \frac{Z}{f_y} \tag{27}$$

$$Y_C = Z \tag{28}$$

Where $X\ frame\ index$ is the RGB frame X-axis of the keypoint pixel in the RGB frame. $Y\ frame\ index$ is the RGB frame Y-axis of the keypoint pixel in the RGB frame. f_x is the focal length for X-index,f_y is the focal length for Y-index, c_x is the principal point offset of X-index, c_y is the principal point offset of the Y-index, Z is the depth value at that pixel in meters, (X, Y, Z) are reconstructed 3D coordinates in the camera reference frame in meters.

This transformation ensures that the body keypoints, detected initially in image space, are mapped onto physically meaningful 3D locations. The resulting metric coordinates are used throughout the system for joint tracking, skeleton modeling, and robot interaction. The values required for the 3D construction of the keypoints are shown in Table 3.

Table 3: Camera Intrinsic Parameters (640×480 Resolution)

	X-direction	Y-direction
Focal Lengths (pixels)	f_x = 320	f_y = 426
Principal Point Offsets (pixels)	c_x = 320	c_y = 240

5.6 Real-time Pose Detection Performance

Two case studies are presented. In the first study, only body key points are detected, at a 60 Hz processing speed. In the second study, both body and hand keypoints are detected simultaneously at 20 Hz due to increased computational demands. The entire pose detection pipeline includes frame capture, processing, and filtering. When detecting only body keypoints, it takes less than 15 ms per frame. When both body and hand keypoints are detected, the processing time increases to about 40–50 ms. Based on the minimum 20 Hz detection rate and the relatively slow nature of human movements during rehabilitation tasks, the system provides sufficient temporal resolution to capture relevant human motion throughout the session.

The entire processing loop was measured to evaluate system latency. This loop includes camera frame acquisition and updating the human kinematic model in PyBullet. Optimized multithreaded execution was used in the system. Pose detection and filtering were performed in parallel with robot control logic and game updates. The delay between a physical human movement and its visual representation in the simulation environment was consistently below 70 ms. This latency level is sufficiently low to support real-time motion capture and visualization in HRI applications.

Based on the measured processing rates and end-to-end latency reported in the two case studies, the proposed vision system operates in soft real-time. Although the update frequency decreases when both body and hand keypoints are detected, the system maintains bounded latency and consistent temporal behavior, enabling responsive motion capture and visualization throughout the interaction. These characteristics satisfy the timing requirements of real-time vision-based HRI in rehabilitation applications.

5.7 Summary

This chapter described the developed vision system for the proposed setup. The vision system utilizes pre-trained AI models to detect human keypoints. YOLOv8n-Pose runs on the GPU for body joint detection, while MediaPipe runs on the CPU for hand keypoint detection. The system initializes the robot's 3D base using visual landmarks and continuously estimates 3D human pose by integrating 2D keypoints from RGB frames along with depth frames. To mitigate noise and misalignment inherent in RGB-D sensors, filtering techniques were applied to refine depth measurements. These include K-means clustering and Kalman filtering, which improve the reliability of the reconstructed model. The system’s high-frequency performance was achieved through multithreading, separating vision, control, game logic, and simulation processes. The vision system can operate in two modes: body keypoints detection at 60 Hz, without hands, or full-body tracking at 20 Hz, including hands, providing flexibility. Latency evaluations confirmed an end-to-end delay below 70 ms, supporting the system’s ability to provide responsive interaction in HRI applications.

6 Game-based Rehabilitation

6.1 Game Design

The system is designed to serve as an effective Human-robot Interaction (HRI) platform for rehabilitation. Focusing only on interaction with a robot is insufficient for therapeutic benefit. Therefore, the system emphasizes the importance of establishing shared goals between the human and the robot. Incorporating a game element enhances the practicality and engagement of the system. This approach provides patients with an immersive therapeutic experience. Additionally, the enhanced system allows the robot to receive feedback on human performance and adapt its responses accordingly.

By combining entertaining gameplay with robotic assistance, the enhanced system encourages humans to participate in their rehabilitation process. The game is structured to be adaptable and motivating, allowing individuals with varying levels of mobility to benefit from the interaction while keeping its simplicity. The primary considerations in the design include clear objectives, well-defined game mechanics, and an accessible user interface. The system ensures that human can control the robotic arm naturally, thereby improving their motor function. It is worth noting that the vision system and the game are independent systems that operate separately and can function without relying on each other.

6.1.1 Joint Movement

The game should be used for rehabilitation applications targeting various joints and movement patterns. However, the main focus of this implementation is the shoulder joint, which is essential for upper-limb mobility and functional recovery. The shoulder joint enables different groups of motions, including Flexion, Extension, Abduction, Adduction, Internal Rotation, and External Rotation [58], these movements are illustrated in Figure 6.1. In this system, the focus is placed on flexion and extension movements. These movements are essential for daily functional tasks such as reaching forward, lifting objects, and restoring general shoulder mobility. By targeting flexion and extension movements, the system aims to improve range of motion, muscle coordination, and motor control. This supports effective rehabilitation for patients recovering from shoulder injuries or mobility impairments.

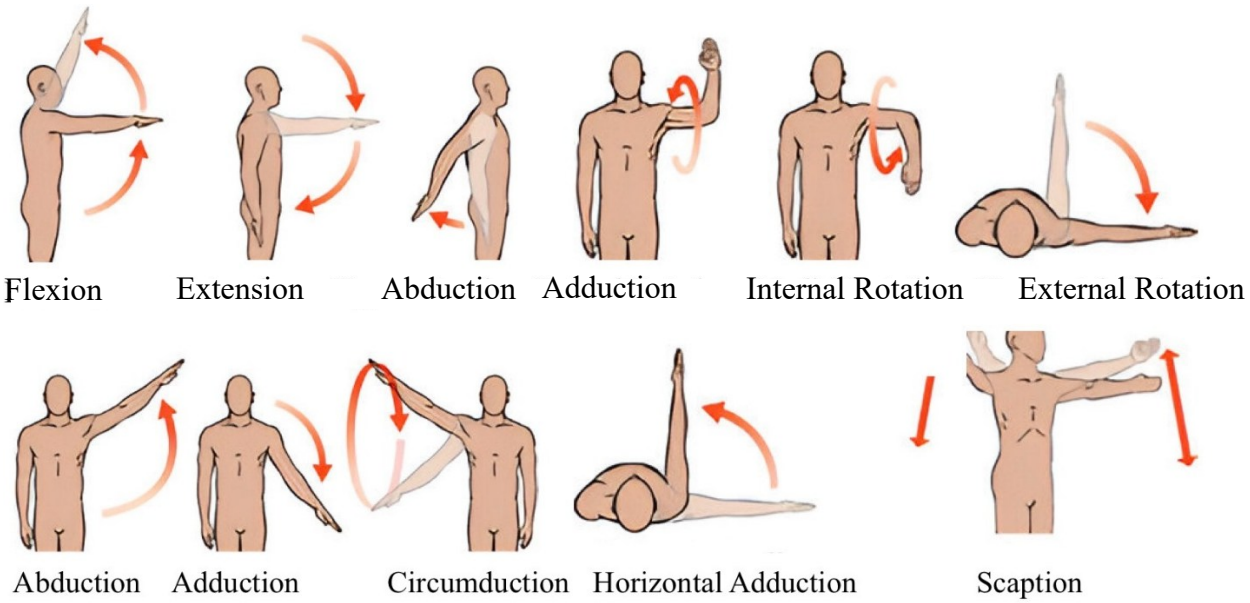

Figure 6.1: Movement range of the upper extremity.

6.1.2 Objectives

The main goal of the game-based rehabilitation using HRI is to enhance motor recovery of the shoulder joint, increase engagement, and support tracking progress during sessions. To fulfill this goal, the following objectives are considered in designing the game:

I. Repetitive Movements: The game encourages repeated execution of targeted movements. This repetition is critical for motor recovery because it reinforces muscle memory and gradually improves motor function.
II. Hand-Eye Coordination: The game requires coordinated control of the robotic end-effector, enabling users to interact with virtual objects and improve hand–eye coordination through gameplay.
III. Feedback: The game provides meaningful performance feedback, generating quantitative and qualitative information that can be used to assess human motor ability during task execution.
IV. Patient Adaptation: The game includes adjustable difficulty levels that adapt to the user's performance, ensuring that rehabilitation tasks remain challenging while still achievable.
V. Tracking Progress: Quantitative performance metrics, such as movement accuracy, reaction time, and task success rate, are recorded and analyzed to evaluate patient improvement across multiple sessions.

VI. Data recording: Human body posture keypoints and robot configuration data are recorded throughout gameplay to enable post-session analysis and long-term assessment.

VII. Simplicity: The game interface and interaction mechanics are intentionally kept simple to minimize cognitive load. This ensures that performance reflects the user's motor capabilities rather than task complexity.

6.1.3 Feedback and Target Tracking

The shared target serves as a central element connecting the human, the robot, and the game environment. It establishes a common goal for collaboration. It creates a mutual reference point that helps the robot interpret the human's actions within the game context and adapt its behavior accordingly. The system enables real-time visibility of task goals, making it easier for the patient to align movements with the objectives. Simultaneously, the robot continuously monitors the difference between the person's current position and the target to determine its own movement strategy. This interaction promotes adaptive responses from the robot, encouraging coordinated, goal-oriented motion and enhancing rehabilitation outcomes.

6.1.4 Game Characteristics

The game is designed with a simple yet effective mechanism that aligns with rehabilitation objectives. The game includes a paddle controlled by the human via the robot's end-effector, shown in Figure 6.2. A moving ball appears on the screen, requiring the participant to move the paddle to intercept and catch it. The difficulty level is adaptable to the user's ability, increasing speed or complexity to provide a challenging yet supportive experience. In each game session, multiple balls are generated sequentially from the left side of the screen and move horizontally to the right. The patient must position the paddle correctly to catch each ball before it reaches the end of the screen.

To illustrate the game mechanics, six sequential screenshots captured during gameplay are presented in Figure 6.2. In frame (a), a new target is generated, establishing a shared objective for both the human participant and the robot. The initial positions of the target and the paddle are defined, along with their intended directions of motion. In frame (b), the target begins moving toward the interaction zone, indicated by a red cross. This allows both the human and the robot to

predict the target trajectory and align their movements accordingly. Frame (c) depicts the moment of interception, where the target reaches the paddle, and a successful interception indicates accurate tracking and coordination. In frame (d), the game updates the score and prepares for the next round. Frames (e) and (f) illustrate the continuation of the gameplay. During this stage, the human and the robot adapt to each other's movements. This sequence demonstrates cooperative motion, effective real-time target tracking, and shared task execution between the human participant and the robotic system.

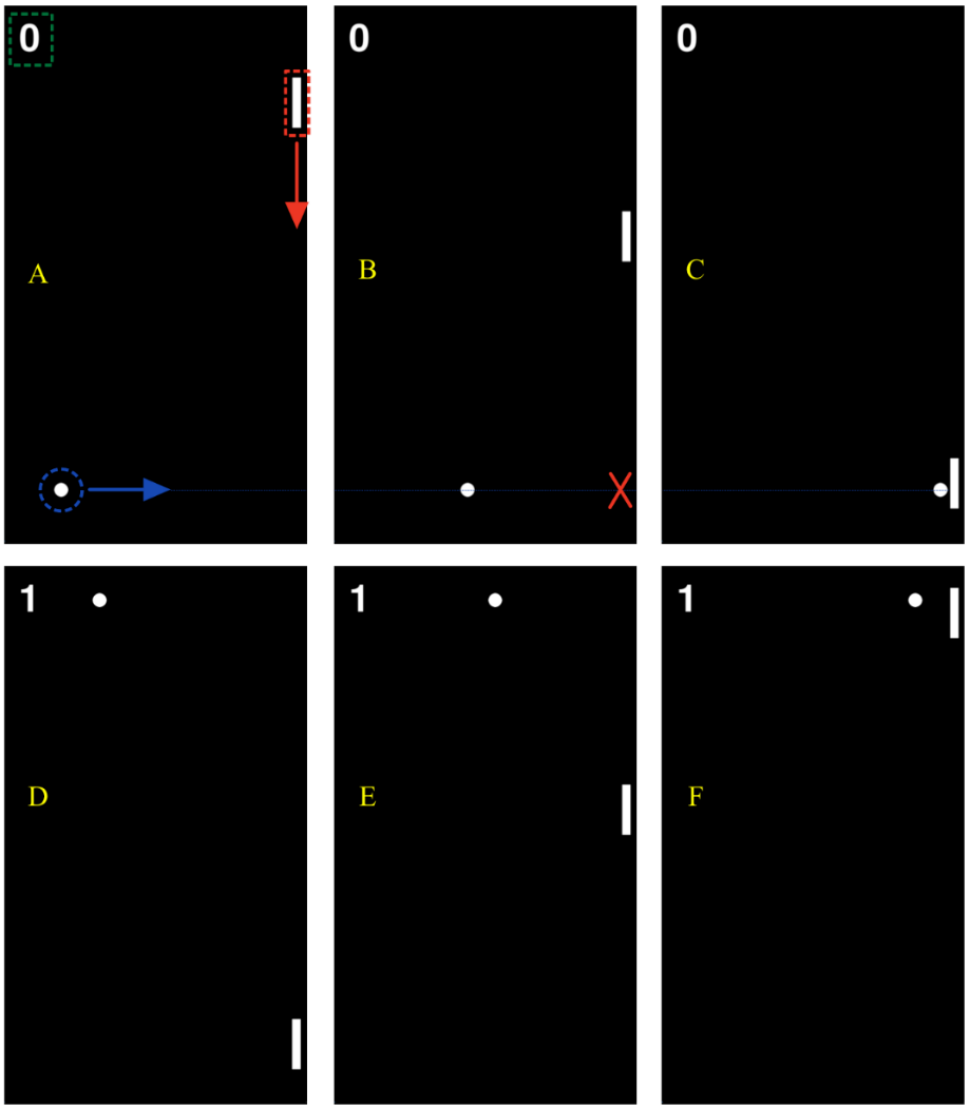

Figure 6.2: Sequential screenshots of the game; (a) Initialization; (b) Target motion; (c) Interception; (d) Score updates; (e) Next target generation; and (f) Intercepting the ball.

6.2 Robot-game Integration

The integration between the robot and the game is fundamental, ensuring a seamless and practical rehabilitation experience. The robot acts as an intermediate element between the human and the game. It translates the patient's movements into meaningful actions within the game environment. The game is designed to respond to patient-driven inputs, allowing real-time control of the paddle through the robot while providing target tracking to guide their actions.

6.2.1 End-position and Paddle-position Mapping

One of the few characteristics of the robot understandable to both the human and the game system is its end-effector position. The human interacting with the robot naturally perceives the end-effector's location. When this position is synchronized with the paddle in the game, the interaction becomes intuitive. The user can move the paddle up and down by moving the robot's end-effector along the vertical axis.

The paddle in the game moves in only one dimension, aligned with the Z-axis (Z_o) of the robot's base coordinate system in Figure 5.3. From the human's perspective, this interaction is mentally simple to understand. Moving the robot's end-effector upward moves the paddle up, and moving it downward moves the paddle down. This eliminates the need for complex mental mapping. Based on this mapping strategy, the vertical Z-position of the robot's end-effector is directly mapped to the paddle position in the game. Figure 6.3 illustrates the definition of the robot's playable vertical range relative to its base coordinate system. In Figure 6.3(a), the robot's end-effector is positioned at the lowest meaningful vertical location, corresponding to a height of 27 cm along the robot's base Z_0-axis. This position defines the lower boundary of the playable zone and corresponds to the green arrow distance indicated in the figure.

Figure 6.3(b) shows the highest meaningful vertical position of the robot's end-effector, which is located 40 cm above the lower boundary along the Z-axis. As a result, the maximum defined vertical position in the game corresponds to a height of 67 cm relative to the robot base. The vertical distance between these two limits defines the robot's playable zone. Within this region, changes in the robot's end-effector Z-position directly affect the paddle position

End-effector positions below 27 cm or above 67 cm are saturated at the corresponding lower or upper paddle limits, as positions outside this range are not meaningful for gameplay.

Movements along the *X* and *Y* directions are not mapped to any game parameters and therefore do not influence the game behavior.

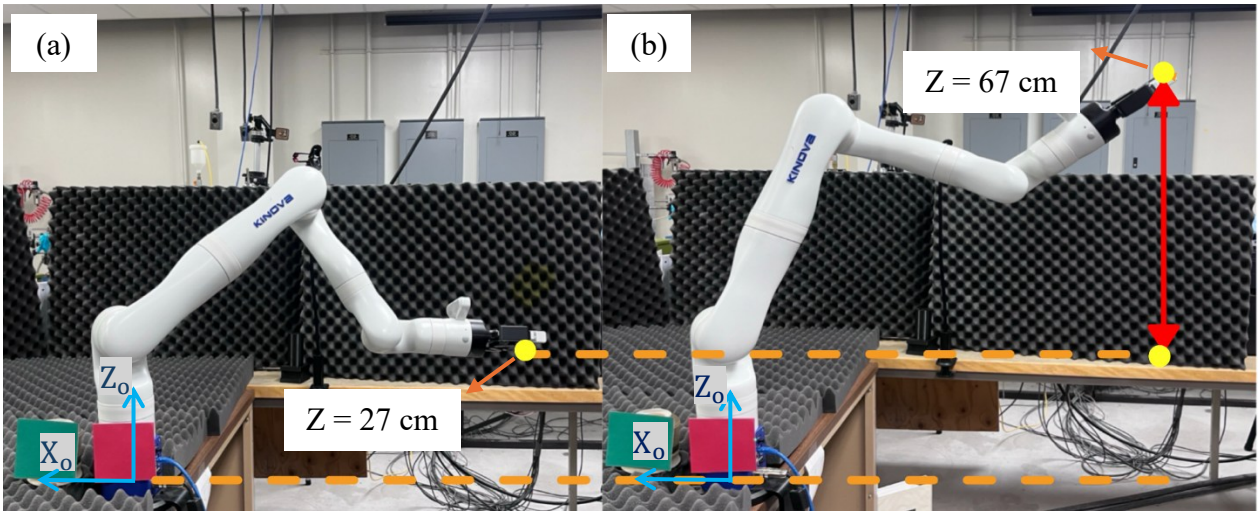

Figure 6.3: Robot end-effector vertical range relative to the base coordinate system in the game: (a) Lowest meaningful end-effector z position, (b) Highest meaningful end-effector z position.

Figure 6.4 illustrates the movement of the robot end-effector and the corresponding paddle. When the robot is at the lowest position within its movement range, the paddle is also at the bottom of the display. Meanwhile, a ball starts moving to the right end of the screen and should be caught by the paddle. As the end-effector moves toward the upper limit of its playable range, the paddle also reaches its upper boundary of movement, resulting in catching the ball at its intersection.

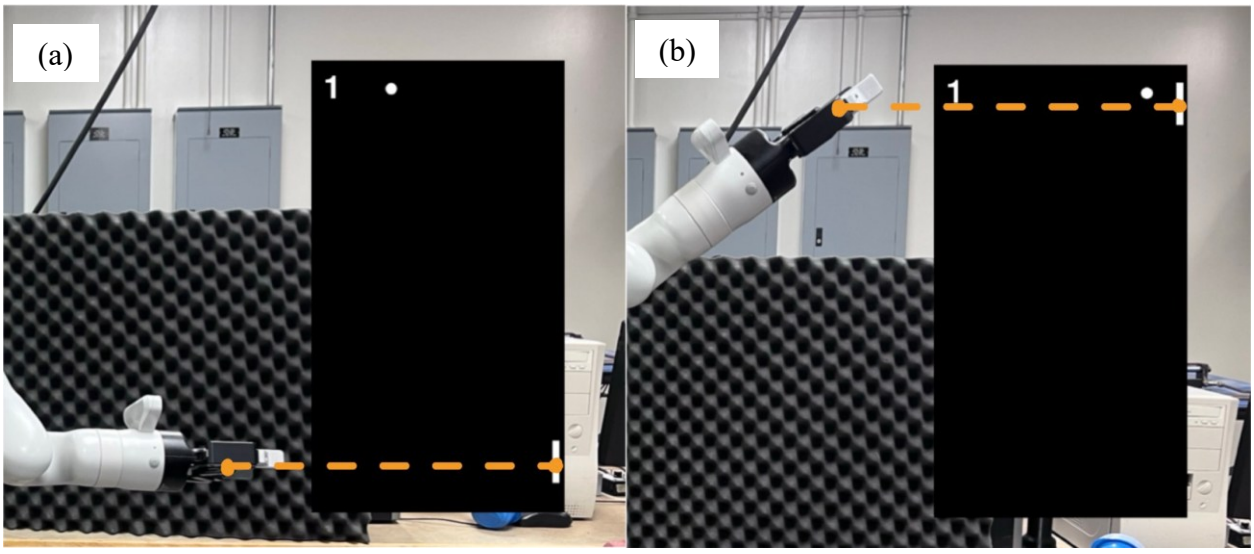

Figure 6.4: Mapping of the robot end-effector to paddle position; (a) lowest z-position is paddle lower limit, (b) highest z-position is paddle upper limit.

The ball speed in the game is adjustable to tailor the difficulty level, suitable for each participant's needs and abilities. Game elements are simple enough to prevent cognitive overload for humans. The game control is straightforward. The participant interacts naturally using hand movements, eliminating the need for complex button inputs.

Regarding the Paddle Movement and speed, the playable range of the robot's end-effector position is 40 cm along the Z axis of the robot base coordination system. The display has a resolution of 1080×1920 pixels and a 24-inch size. The vertical side of the screen with 1920 pixels is mapped to the playable zone, and the paddle position is synchronized with the robot's end-effector and updated accordingly. Therefore, the paddle position and speed depend on the robot's end-effector position and velocity. They are also influenced by the display dimensions. The sizes of the game elements, such as the paddle and the ball, depend on the display's pixel size as well. To derive the ratio between paddle speed and end-effector speed, the display dimensions and pixel size must first be determined. The vertical paddle movement in the game can be transformed into horizontal movement if the target movement changes. This process is illustrated in Figure 6.5.

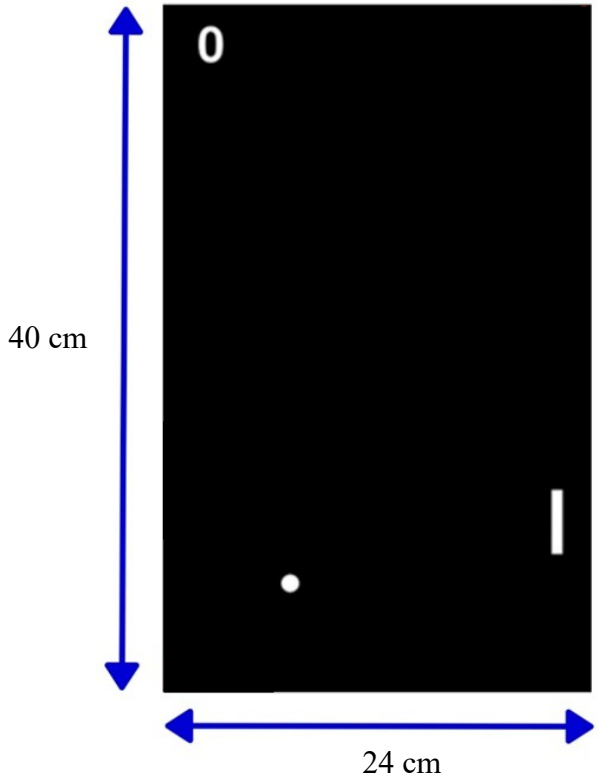

Figure 6.5: Nominal dimensions of the display used in the system.

Table 4: Display and Game Elements (Pixels).

	Height (pixels)	Width (pixels)	Diameter (pixels)
Display	1920	1080	
Pixel	1	1	
Paddle	180	30	
Ball			50

To derive the display dimensions and each pixel size, the following equations are used:

$$Display\ ratio = \frac{Width\ \ pixels}{Height\ pixels} \tag{29}$$

$$Width\ size = Display\ diagonal\ size \times \frac{Width\ \ pixels}{\sqrt{Width\ \ pixels^2 + Height\ pixels^2}} \tag{30}$$

$$Pixel\ width\ = \frac{Width\ size}{Width\ pixels} \tag{31}$$

$$Height\ size = Display\ diagonal\ size \times \frac{Height\ \ pixels}{\sqrt{Width\ \ pixels^2 + Height\ pixels^2}} \tag{32}$$

$$Pixel\ height\ = \frac{Height\ size}{Height\ pixels} \tag{33}$$

where display ratio is the aspect ratio of the display, width pixels is the number of pixels the display width has, height pixel is the number of pixels the display height has, display diagonal size is the size of the display diagonal in cm, width size is the size of the display width in cm, pixel width size is the size of the pixel width in cm, height size is the size of the display height in cm, pixel height size is the of the display height in cm,

Using the above equations, the pixel and display dimensions are derived and shown in Table 5 for the display used in this study.

Table 5: Display and Game Elements (cm).

	Height (cm)	Width (cm)	Diameter (cm)
Display	53.76	30.24	60.96
Pixel	0.028	0.028	
Paddle	4.98	0.83	
Ball			1.4

To find the ratio of the robot's paddle movement to the end-effector speed, first note that a 40 cm movement by the robot corresponds to 53.16 cm on the display. Based on this, the ratio of the paddle's displacement to the end-effector's movement is calculated as follows.

$$Paddle\ to\ End\ effector\ Ratio\ of\ Displacement = \frac{\Delta\ Paddle}{\Delta\ End\ effector} \tag{34}$$

where:

$\Delta\ Paddle$ = Displacement of the paddle,

$\Delta\ End\ effector$ = Displacement of the end effector,

$$\frac{v_{Paddle}}{v_{End\ effector}} = \frac{\frac{d\Delta\ Paddle}{dt}}{\frac{d\Delta\ End\ effector}{dt}} = \frac{\Delta\ Paddle}{\Delta\ End\ effector} \tag{35}$$

where:

v_{Paddle} = Paddle velocity,

$v_{End\ effector}$ = End-effector velocity,

Based on these equations, the ratio of the v_{Paddle} to $v_{End\ effector}$ is 1.34, which means that if the end-effector position moves at 1 m/s, the paddle moves at 1.34 m/s.

6.2.2 Game Loop and Updating Frequency

The game runs at 60 frames per second (FPS) to ensure smooth, responsive interactions. This high update frequency enables precise tracking of the patient's movements while minimizing latency. It also ensures that the paddle responds accurately to real-time robotic end-effector adjustments. A 60 FPS update rate also enhances visual fluidity, reducing motion blur and making the gameplay experience feel more natural and engaging.

6.2.3 Robot-game communication

Communication between the game and the robot is vital. It enables a practical interaction session and provides valuable data for analyzing human performance during the interaction. Each ball's travel duration from left to right on the screen can be as low as 1.2 seconds. This requires minimal delay in data transfer between the robot and the game.

The robot and the computer are connected via a LAN, operating at approximately 1000 Hz. The PC concurrently controls the robot. It receives sensor feedback, sends control commands according to the designed algorithm, and calculates the end-effector position in real time using angular position sensors. Simultaneously, it runs a game at 60 Hz, dynamically adjusting the paddle position based on the robot's end-effector location.

Given the communication update frequency between the robot and the PC, the delay in receiving the robot's status is approximately 1 ms. In contrast, the game operates at a 60 Hz update rate. Therefore, the delay between the PC updating the robot's end-effector configuration and the changes appearing in the game is about 16 ms. As a result, the total delay from when the human moves the robot's end-effector to when the paddle position updates in the game is ≈ 17 ms.

To analyze this delay, the calculated robot's end-effector position and the paddle position are captured and normalized to enable direct comparison. Both signals were scaled to a range between 0 and 1. This allowed consistent evaluation over a 10-second game session, as shown in Figure 6.6. As shown, the paddle and the robot's end-effector positions overlap closely, demonstrating insignificant delay in the system. Figure 6.7 shows a zoomed in version of the plot in Figure 6.6 from 3.59 seconds to 3.65 seconds. The plot confirms that the delay between the paddle position and the robot's end-effector position is approximately 17 milliseconds.

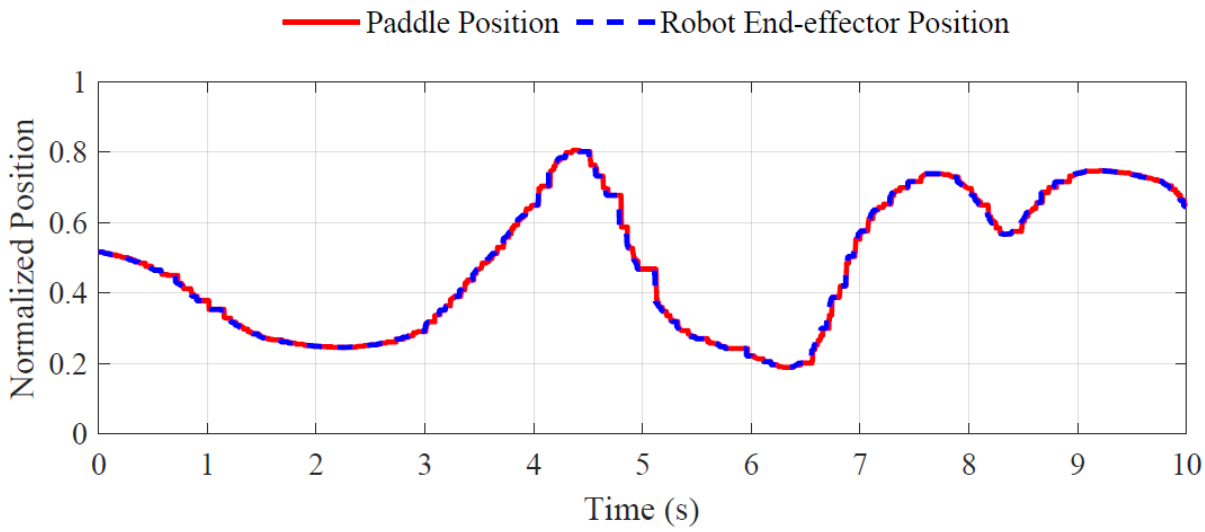

Figure 6.6: End-effector and paddle normalized positions.

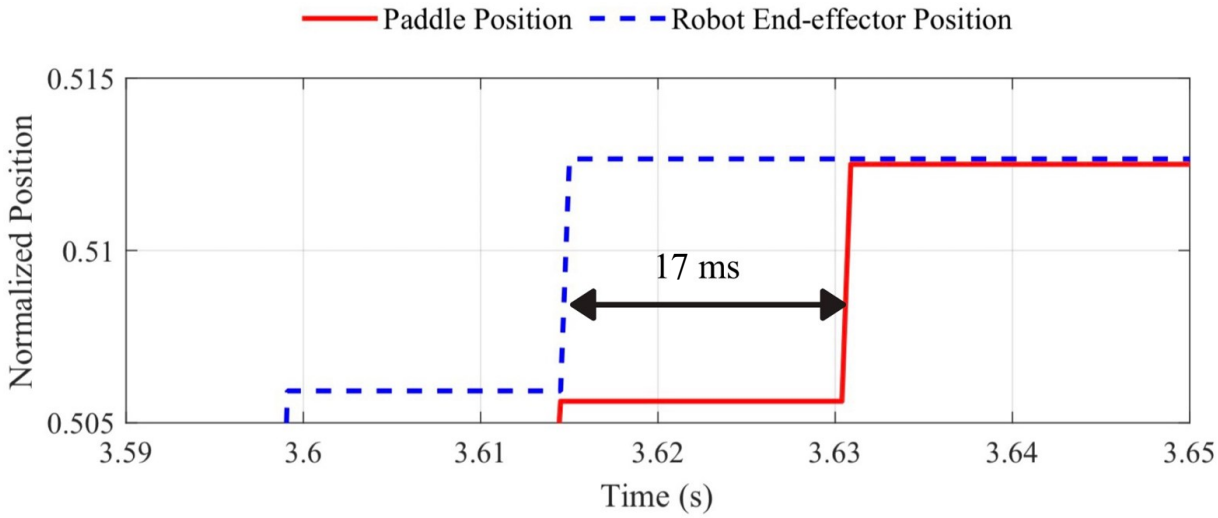

Figure 6.7: End-effector and paddle normalized positions (zoomed out).

6.3 Game Modes

The rehabilitation game features three main modes. Each mode is tailored to the user's ability level and selected according to the recovery stage. These modes ensure that participants receive appropriate levels of assistance or resistance depending on their motor capabilities. The game modes are Passive Mode, Resistive Mode, and Assistive Mode. Each mode plays a vital role in rehabilitation, from early-stage assistance to the development of strength and coordination.

6.3.1 Passive Mode

In Passive Mode, the robot does not exert any force at the end-effector. It moves freely in response to the force applied by the participant. The system gives the participant complete control over the movement, ensuring that the robot follows their natural motions without interference. Passive Mode allows the participant to explore their movement capabilities. This helps assess mobility limitations, reduce stiffness, and improve coordination before progressing to more advanced rehabilitation modes Equation (9).

6.3.2 Resistive Mode

In Resistive Mode, the robotic system applies adjustable resistance to challenge the participant's strength and coordination. This requires the participant to exert more effort and encourages active engagement. The resistance level can be gradually increased as the participant progresses. As a result, three sub-modes are designed in this mode, each of which can be used for different needs, such as improving muscle strength, motor control, and endurance. Resistance mode is suitable for participants with some voluntary movement who need to rebuild strength and coordination, making it ideal for developing muscle strength and endurance during rehabilitation Equations (10) to (12).

6.3.3 Assistive Mode

In Assistive Mode, the robotic system helps the participant complete movements. It does this by amplifying their effort or by guiding the participant's hands during gameplay. It also has three sub-modes designed to help participants reach their targets even when they lack full strength or control. If a participant struggles to reach a target, the robot intervenes to ensure the task is completed successfully. The level of assistance can be customized to meet each participant's individual needs, preventing frustration and discouragement. This keeps participants engaged and makes the mode ideal for those who need extra support to accomplish their tasks Equations (13) to (17).

6.4 Human Performance Analysis

Practical sessions require continuous tracking of the participant's progress to assess improvements and adjust the game level accordingly. To ensure that the participant's movements

align with defined objectives, the system analyzes the participant's movement accuracy after each session.

Several key metrics are analyzed to evaluate performance. One metric is target-reach time, defined as the time required for the paddle to reach a specific neighborhood of the ball. The next one is the game score, which shows the number of balls the participant captures in each session. Reaction time is another critical metric. It measures how quickly the participant responds to a newly generated ball, providing insight into reflex improvement and cognitive-motor coordination. Movement consistency is also monitored to determine whether the participant performs movements in a stable, repeatable manner or shows inconsistencies across multiple attempts.

6.5 Benefits of the Game Integration

Integrating game-based interaction into rehabilitation can offer several advantages over traditional therapeutic methods, providing a more engaging and dynamic experience for those who require rehabilitation. By incorporating interactive gameplay, feedback, and robotic assistance, the rehabilitation process becomes more motivating and enjoyable.

Traditional rehabilitation exercises can lead to a lack of motivation, but incorporating gaming elements, such as scoring systems, challenges, and rewards, encourages active participation and long-term adherence to therapy. Additionally, the game environment fosters enhanced neuroplasticity and motor learning, as repetitive, goal-oriented movement practice reinforces correct movement patterns. The system’s real-time tracking ensures that each movement is monitored and stored.

The game environment enhances hand-eye coordination and reflex development by requiring patients to visually track game elements and coordinate motor responses. Its adaptive system supports various recovery stages with Passive, Resistive, and Assistive modes. The system tracks performance metrics, such as accuracy and response time, reducing fatigue through interactive elements like animations and difficulty adjustments. Overall, incorporating the game element into the system boosts motivation, facilitates faster recovery, and improves outcomes.

6.6 Summary

Chapter 6 presented the design, implementation, and evaluation of a game-based rehabilitation system that integrates a robotic arm with an interactive game environment. The system allows participants to control a paddle in a virtual game using the end-effector of a robot, transforming repetitive motor exercises into an engaging and meaningful activity. The chapter described the game objectives, mechanics, and user interface, emphasizing intuitive interaction and adaptive difficulty. Three different game modes, including passive, resistive, and assistive, were introduced to accommodate various levels of participants' movement motor ability, each presenting a unique challenge. The robot-game integration was thoroughly explained, highlighting the mapping between the robot's end-effector and the paddle position, along with real-time visual feedback that guides the user throughout the session.

7 Visualization Results

7.1 Testing Procedure

Typical human-robot gaming interaction is illustrated in Figure 7.1. The focus is on the application for rehabilitation. Before using the system with individuals with limited physical abilities, its usability must be thoroughly evaluated with individuals who are not undergoing rehabilitative therapy. Their performances were recorded and analyzed, while observations and user feedback were documented.

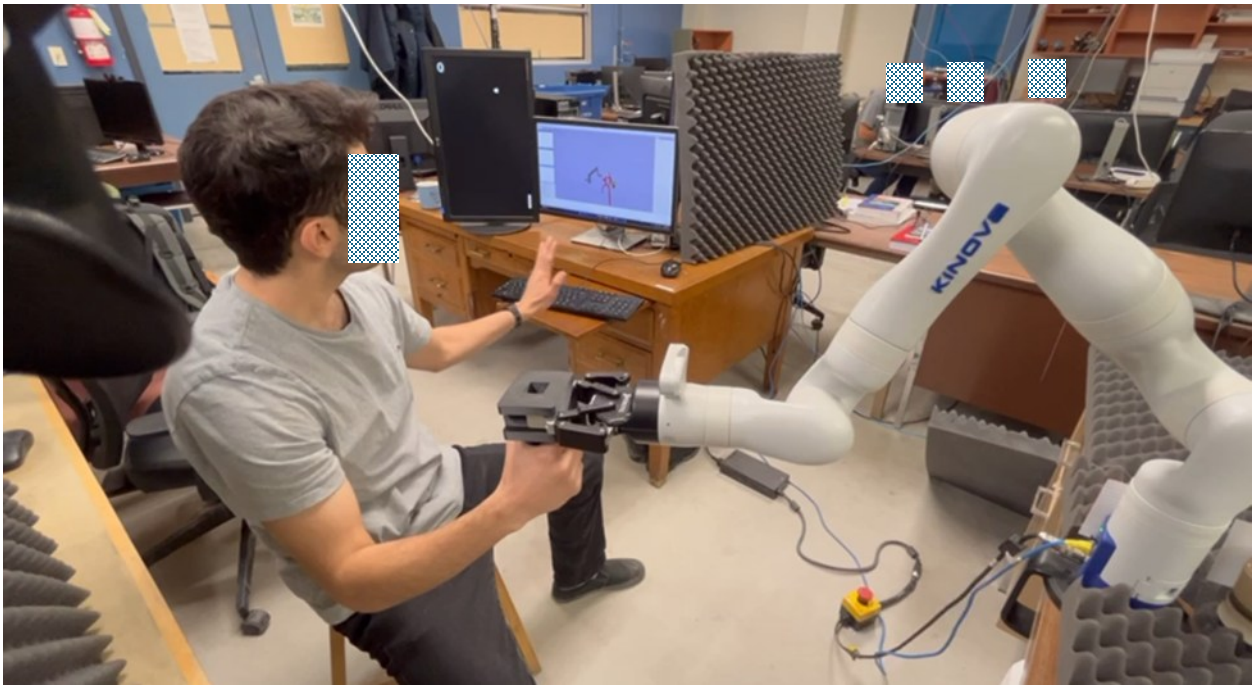

Figure 7.1: Participant playing the game.

Data logs collected during trial runs involving seven healthy volunteers (participants) were analyzed and visualized using a series of graphs. Each participant played the game once and experienced all available control modes. The resulting data provided insights into system performance, user adaptation, and potential areas for improvement.

The designed system incorporates adjustable parameters that are tailored to the participant's conditions to provide a suitable gaming interaction session, as shown in Table 6. These parameters can be divided into two groups: game parameters and Human-robot Interaction (HRI) parameters.

The game parameters include paddle size, session duration, ball speed and diameter, and paddle movement trajectory. The HRI parameters consist of the assistance gain, proportional–derivative (PD) control coefficients, and the range of applied random forces, which together define the system's control configuration.

Table 6: Fixed game parameters used during experimental trials.

	Height (cm)	Width (cm)	Diameter (cm)	Speed (cm/s)	Time (s)
Screen	53.76	30.24			
Paddle	4.98	0.83			
Ball			1.4		
Ball speed				22	
Session duration					30

Table 7: HRI control parameters used across interaction modes. shows the parameter values used in Equations (9) to (17) across different interaction modes. The primary configurable parameters are the interaction gain and PD-based position control term, which were described in detail in Chapter 4.5. A positive gain amplifies the interaction torque applied by the participant to the robot joints, resulting in increased resistive or assistive force feedback to the participant, whereas a negative gain produces the opposite effect.

The PD-based position control strategy monitors the position error between the paddle and the ball along the robot's Z-axis and adjusts the end-effector motion accordingly to facilitate ball interception during gameplay.

An additional factor, referred to as the random gain, is employed in two of the resistive interaction modes. This gain is updated every 0.5 s and is randomly sampled within the range of −0.1 to +0.2, resulting in variable interaction intensity primarily along the robot's Z-axis. This mechanism introduces controlled unpredictability into the interaction, enhancing user engagement and emulating realistic dynamic disturbances. Overall, the systematically tuned gain parameters across all seven robot joints enable adaptive control behavior suitable for a range of interaction objectives, from assistive to resistive modes, within the experimental framework.

It is important to note that different coefficients were used for various joints of the robot due to differences in actuator sizes. Joints 2 to 4 are equipped with larger actuators, whereas joints 5 to 6 have smaller actuators, resulting in a varied distribution. The additional amplification applied to joints 5 and 6 was based on participant feedback, resulting in an improved user experience during the game. Joints 1 and 7 contribute minimally across the interaction modes, as joint 1 corresponds to base rotation about the Z-axis, which is unnecessary for the task, while joint 7 induces end-effector roll motion that does not affect paddle–ball interaction.

Table 7: HRI control parameters used across interaction modes.

Game mode	Game sub-mode	Parameters	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7
Assistant Mode	Gain	Gain coefficient	0	1	1	1	3	3	0
		P coefficient	0						
		D coefficient	0						
	PD-based	Gain coefficient	0	0	0	0	0	0	0
		P coefficient	+1						
		D coefficient	+0.01						
	Gain-PD-based	Gain coefficient	0	+1	+1	+1	+3	+3	0
		P coefficient	+1						
		D coefficient	+0.01						
Passive Mode:	Gain	Gain Coefficient	0	0	0	0	0	0	0
Resistive Mode:	Gain	Gain coefficient	0	-0.5	-0.5	-0.5	-0.5	-0.5	0
		Random Gain Range	0						
	Random	Gain coefficient	0	0	0	0	0	0	0
		Random gain range	Min = -0.1, Max = +0.2						
	Gain-random	Gain Coefficient	0	-0.35	-0.35	-0.35	-0.35	-0.35	0
		Random gain range	Min = -0.05, Max = +0.05						

During the sessions, participants played all of the mentioned game modes with the specified parameters.

7.2 Analyses of Results

7.2.1 Qualitative Analysis

In this section, the performance of a single participant is selected for qualitative analysis from seven participants. This assessment is based on the positions of the paddle and ball, as well as the torques applied by both the participant and the robot. These elements are crucial for understanding the process's effectiveness and the participant's intuitive engagement with the system.

Figures 7.2 to 7.8 illustrate the selected participant's performance over time based on ball and paddle positions. To ensure consistency and allow direct comparison between the end-effector, paddle, and ball trajectories, all position signals were normalized. Normalization maps each signal to a standard 0 to 1 scale while preserving its overall shape, enabling variables with different physical ranges to be interpreted on an equivalent basis.

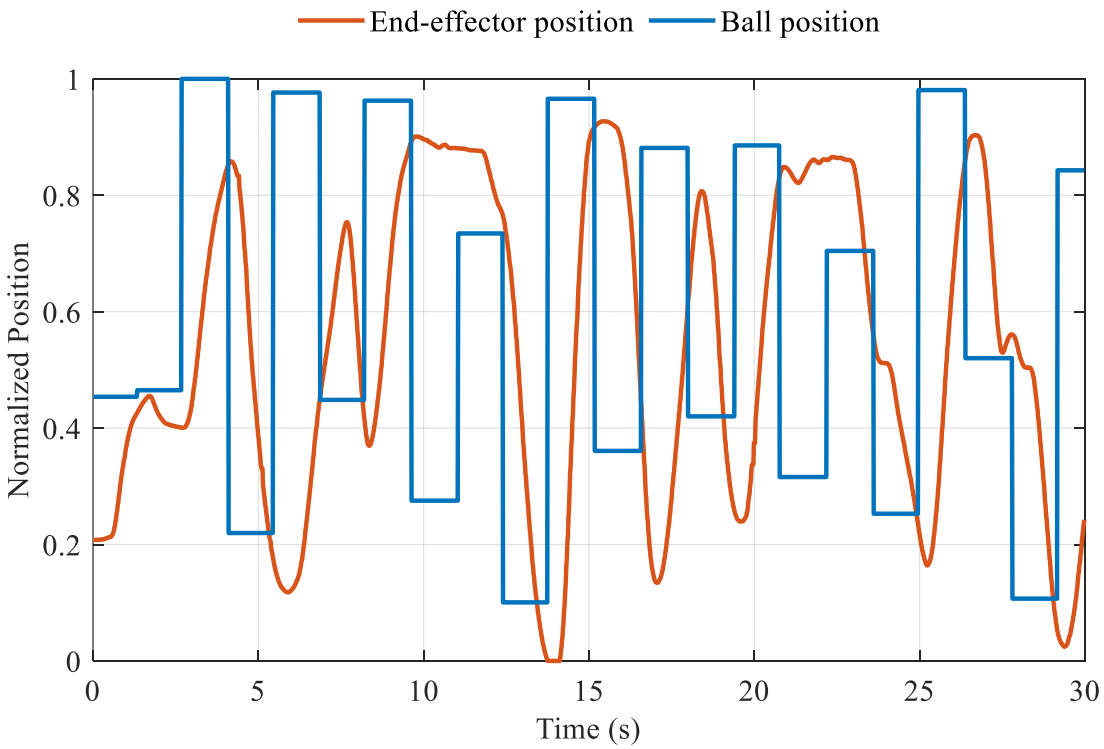

Figure 7.2: Normalized paddle and ball position - Resistive Gain-random mode.

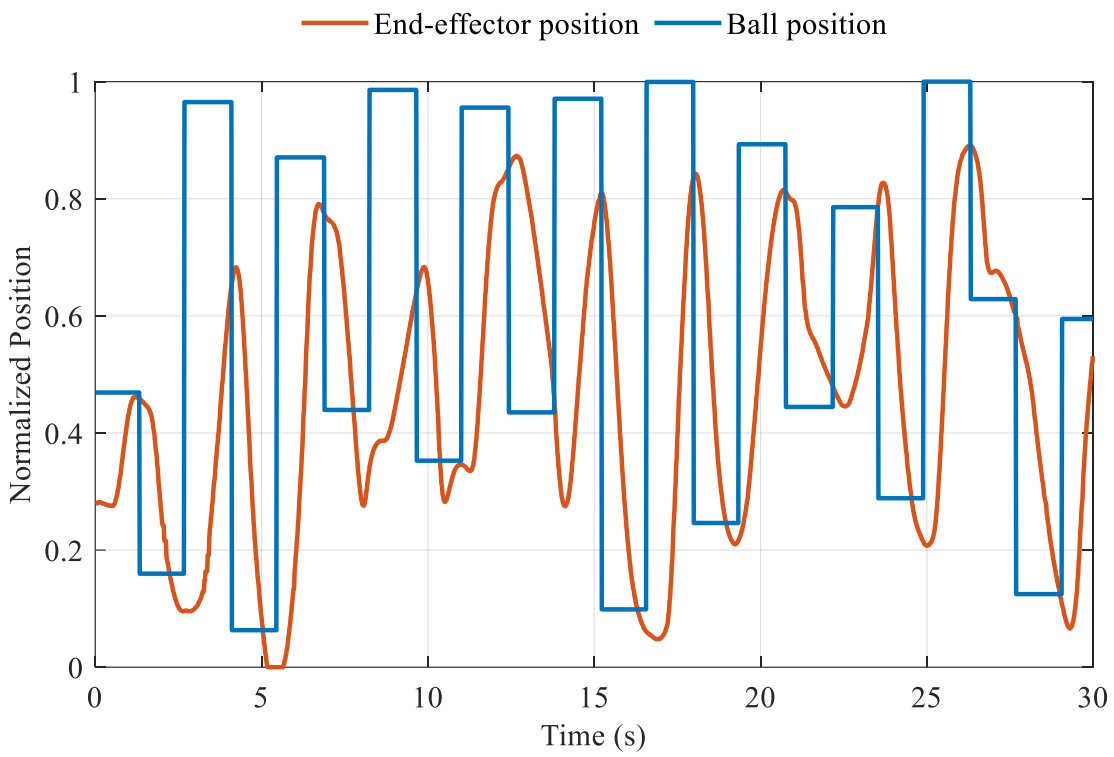

Figure 7.3: Normalized paddle and ball position - Resistive Gain mode.

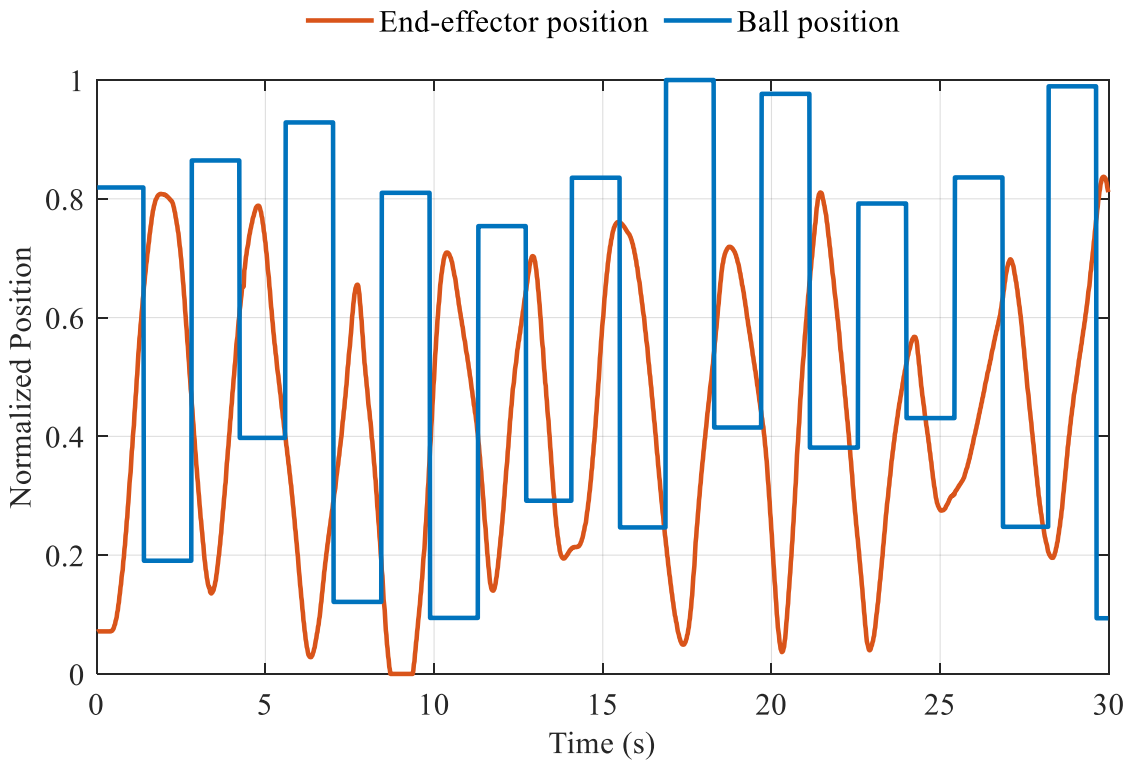

Figure 7.4: Normalized paddle and ball position - Resistive Random mode.

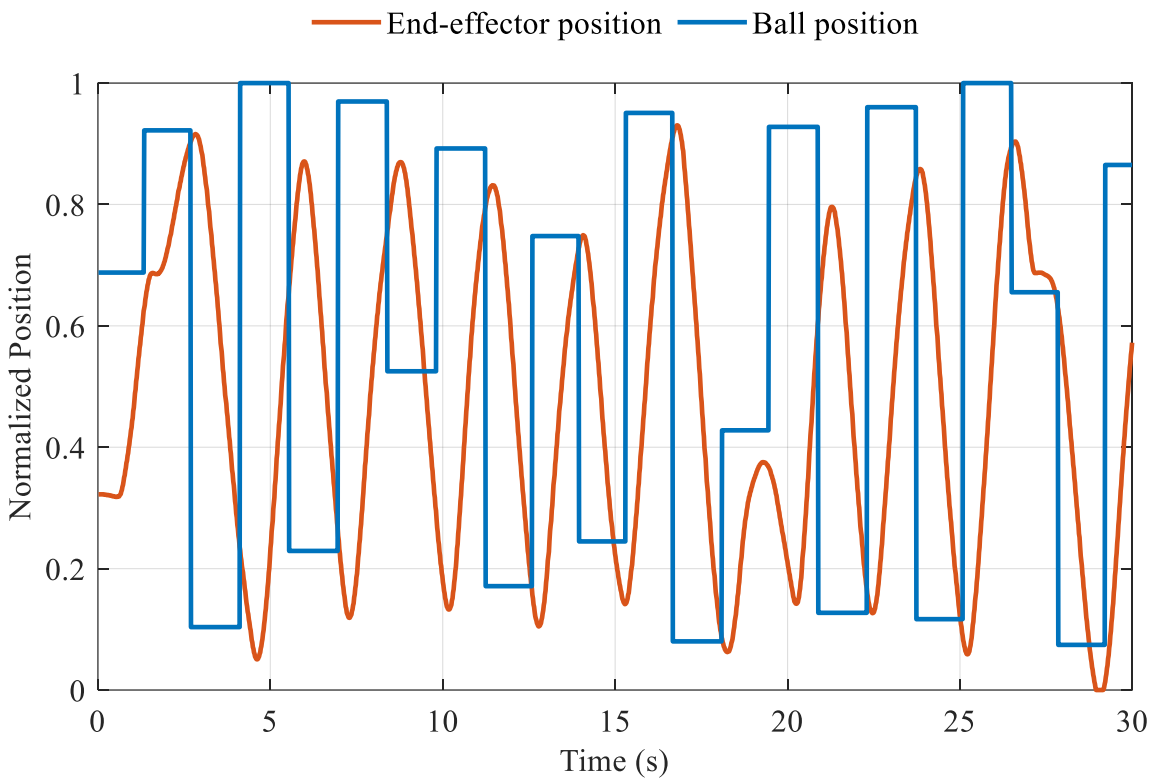

Figure 7.5: Normalized paddle and ball position - Passive mode.

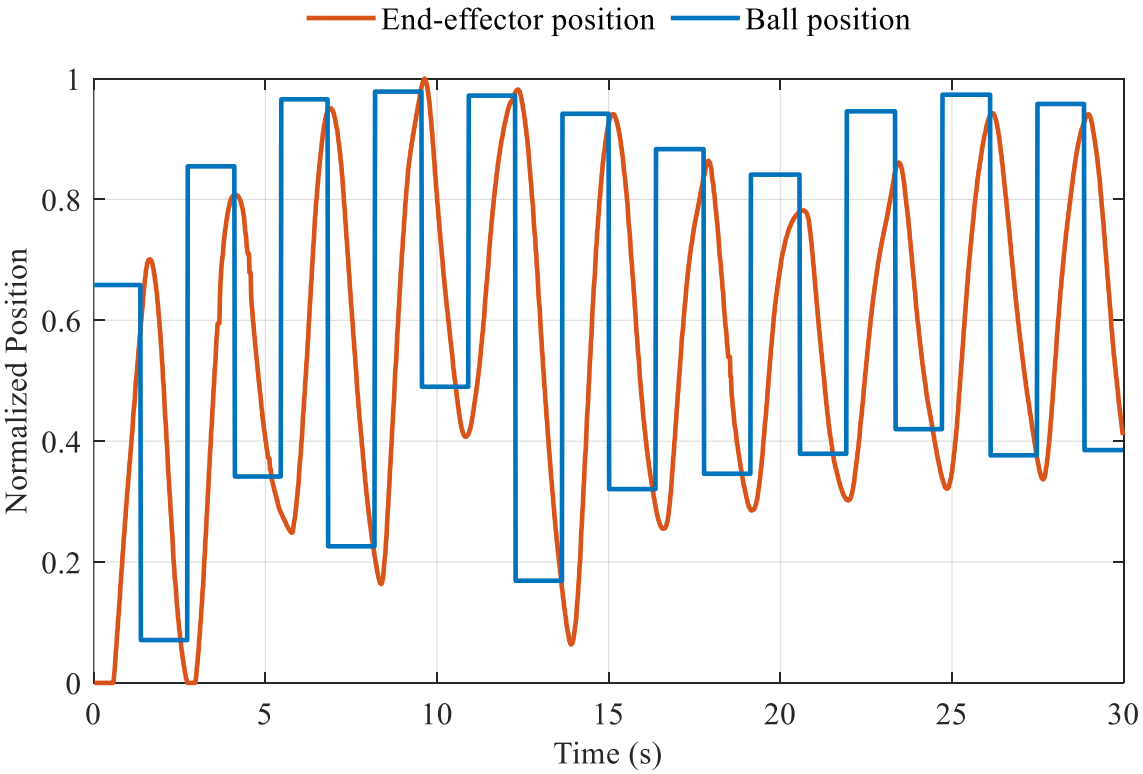

Figure 7.6: Normalized paddle and ball position - Assistive Gain mode.

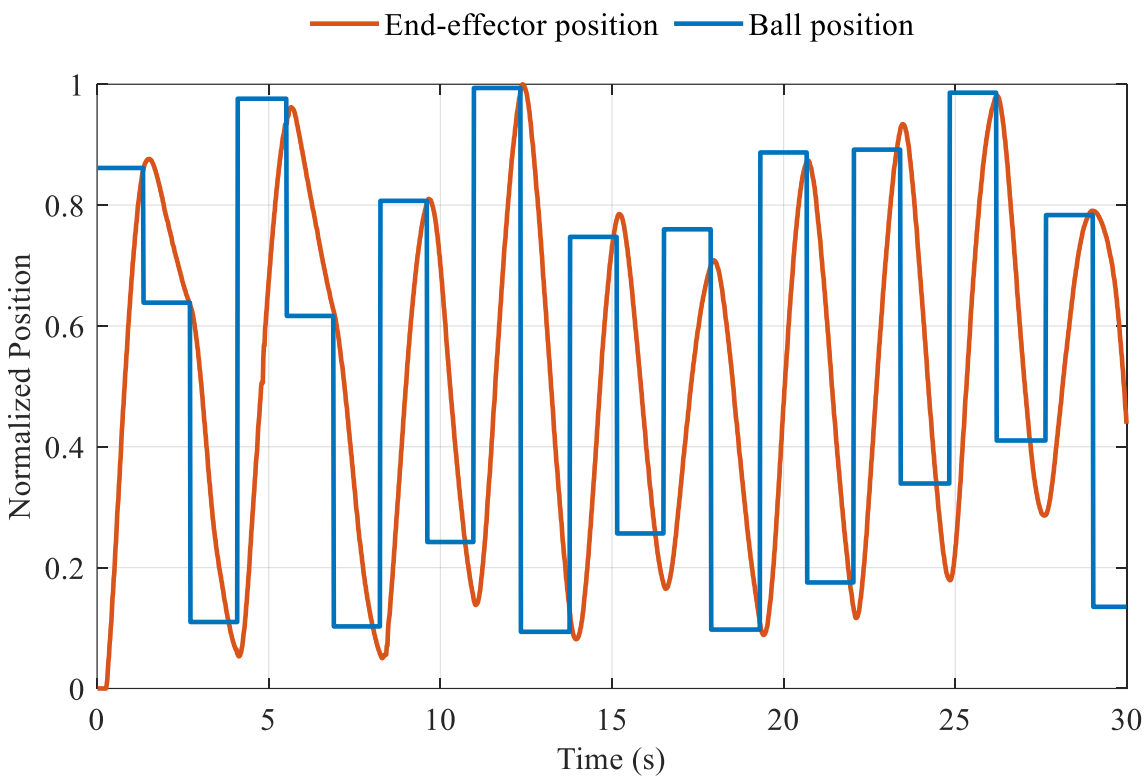

Figure 7.7: Normalized paddle and ball position - Assistive PD-based mode.

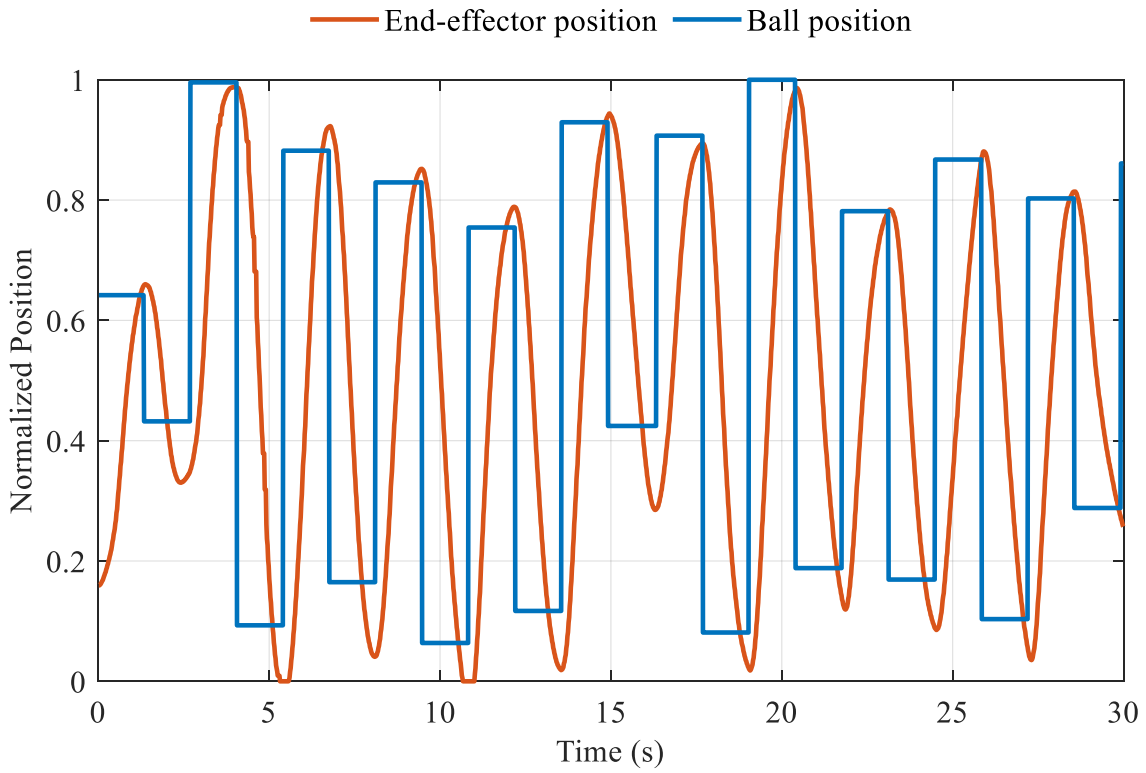

Figure 7.8: Normalized paddle and ball position - Assistive Gain-PD-based mode.

Figures 7.9 to 7.15 illustrate the paddle's displacement from the moment the ball appears on the screen until it either contacts the paddle or is missed and exits the screen. This plot serves as the basis for comparing participants' performances across different game modes. Movements are categorized into two primary types: upward and downward. In this analytical framework, the initial position of the paddle is designated as zero. When the ball's position is above the paddle, it is assigned a positive value; conversely, when below, it is assigned a negative value. The ball speed remains constant throughout the game, resulting in consistent ball travel durations. Trajectories with a total duration of 1.4 s correspond to missed interactions.

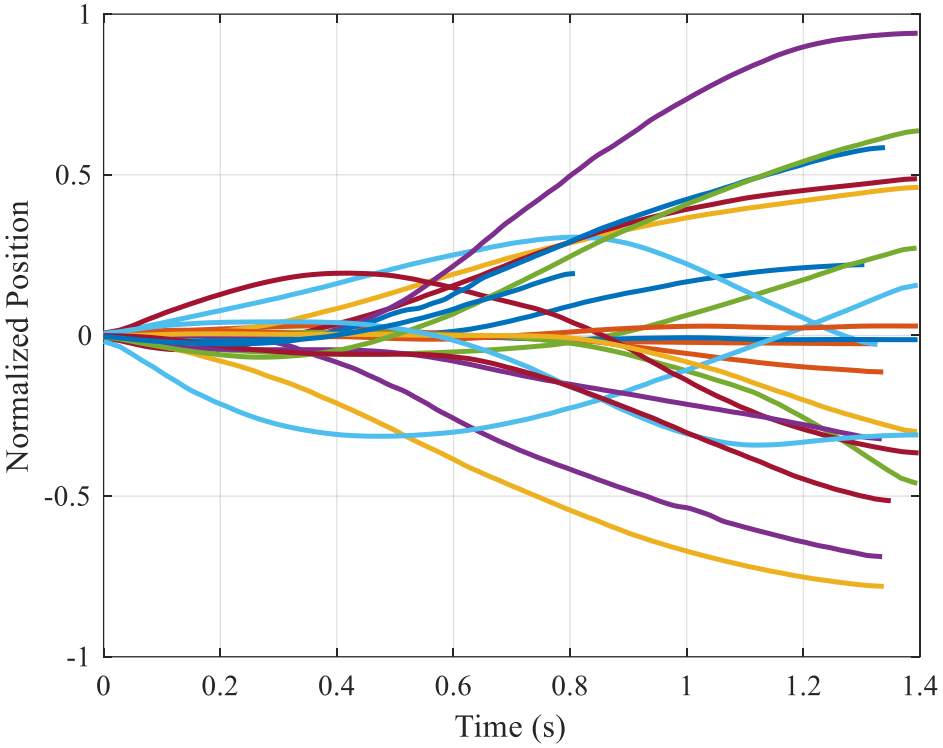

Figure 7.9: Normalized game traces - Resistive Gain-andom mode.

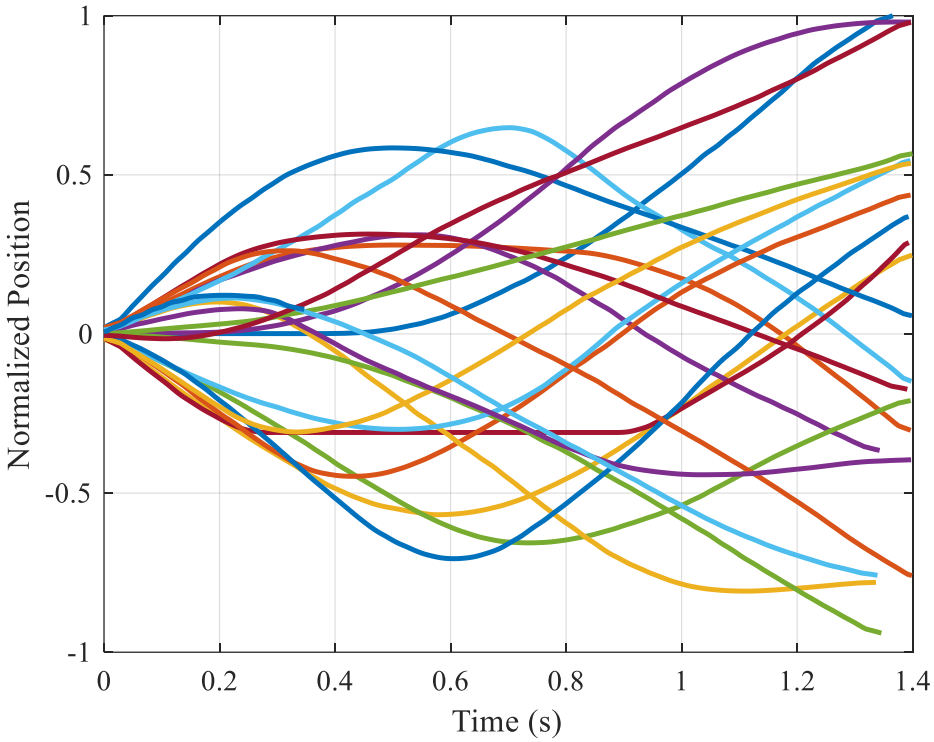

Figure 7.10: Normalized game traces - Resistive Random mode.

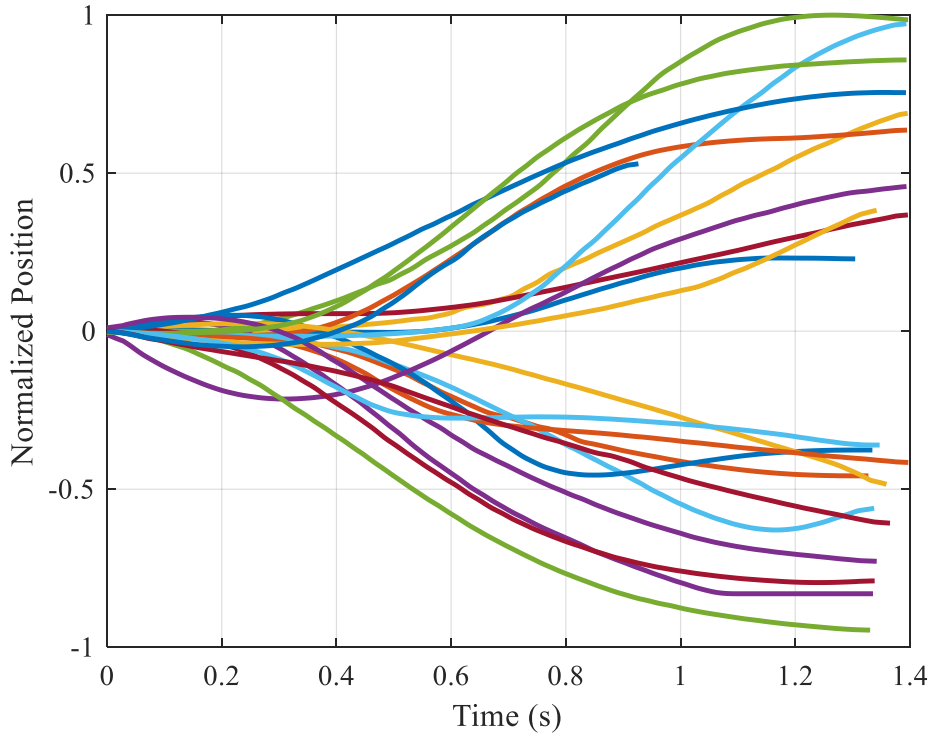

Figure 7.11: Normalized game traces - Resistive Gain mode.

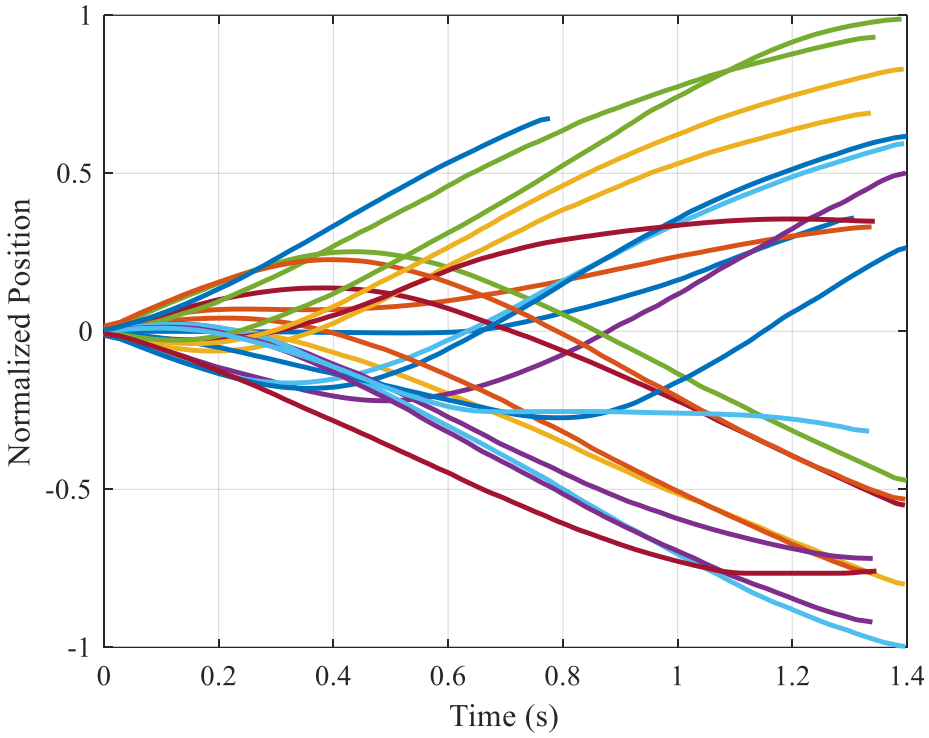

Figure 7.12: Normalized game traces - Passive mode.

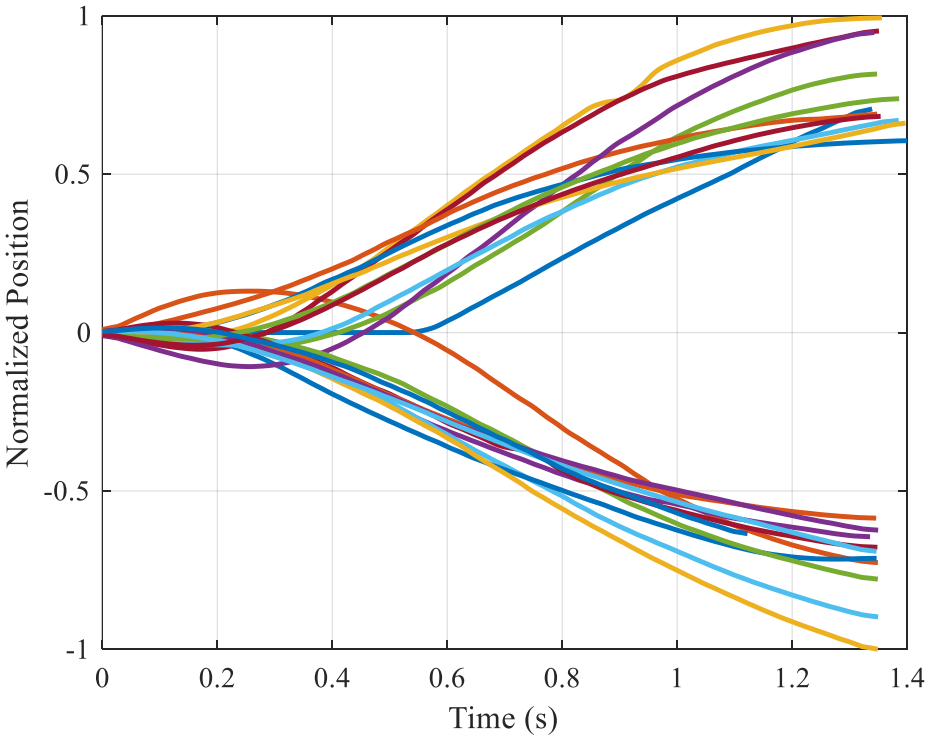

Figure 7.13: Normalized game traces - Assistive Gain mode.

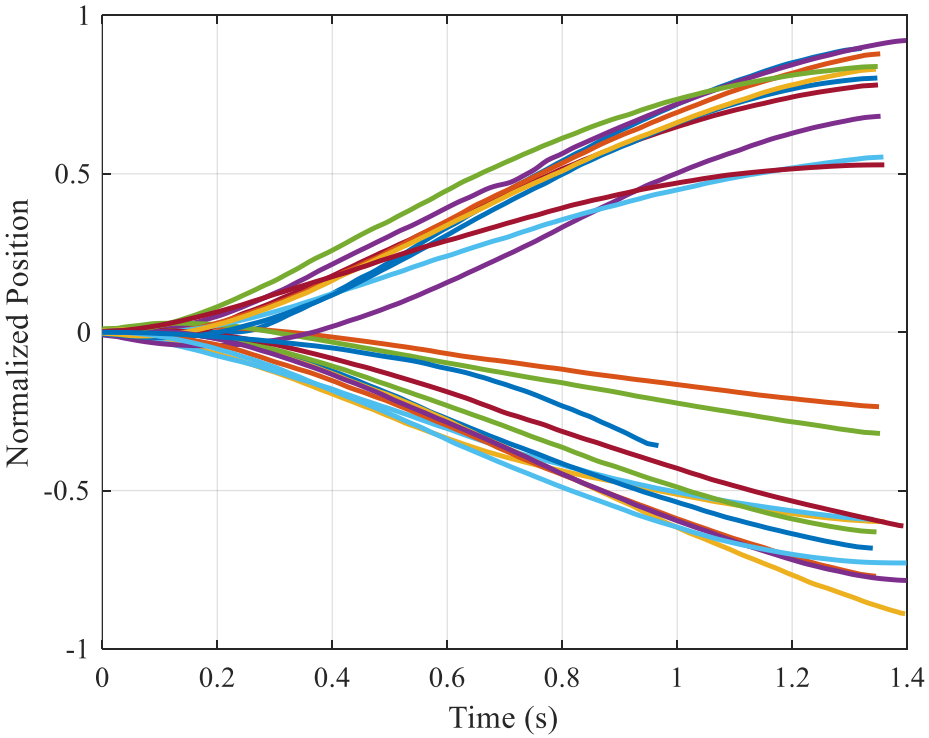

Figure 7.14: Normalized game traces - Assistive PD-based mode.

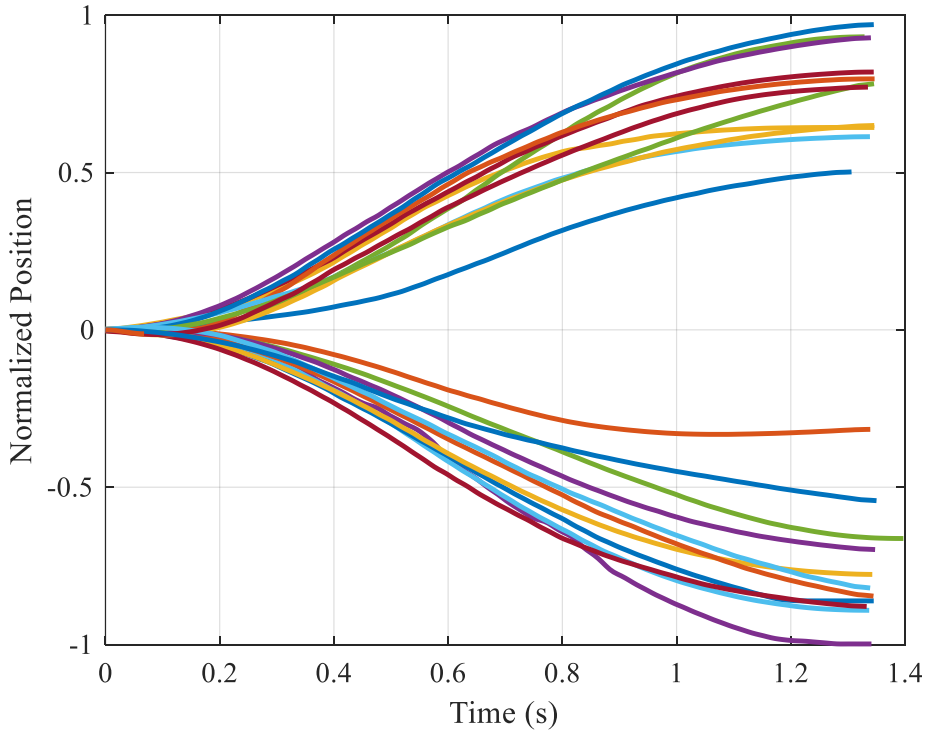

Figure 7.15: Normalized game traces - Assistive Gain-PD-based mode.

Figures 7.16 to 7.22 illustrate the corresponding position difference between the ball and paddle movements for the sessions shown in Figures 7.9 to 7.15. Across Figures 7.16 to 7.22, the position difference gradually converges to zero.

Comparing the Passive mode in Figure 7.19 with the Resistive modes shown in Figures 7.16 to 7.18, it is evident that convergence of the position difference occurs more slowly under resistive interaction. In contrast, Figures 7.20 to 7.22 demonstrate that the system assists the user in reducing the position difference more rapidly than in the other modes.

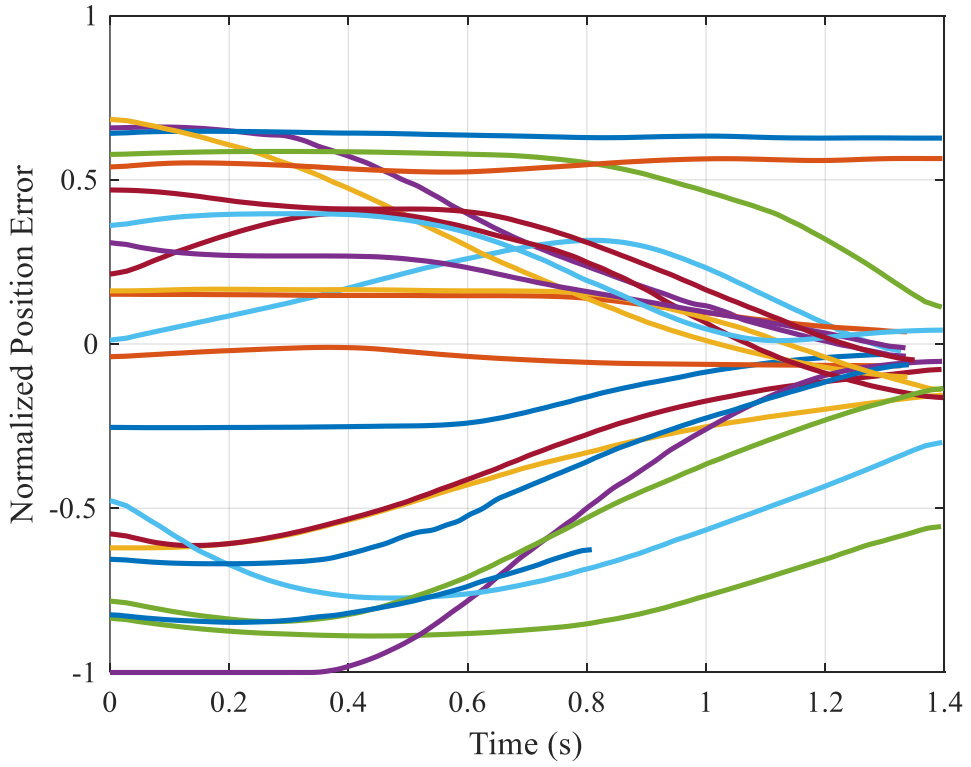

Figure 7.16: Normalized paddle-ball position difference - Resistive Gain-random mode.

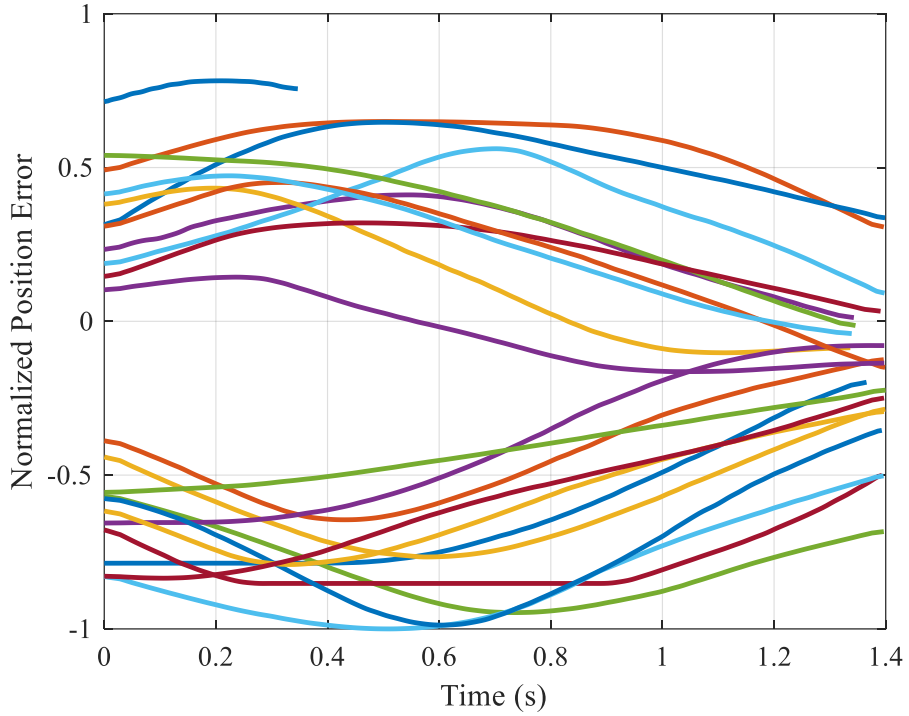

Figure 7.17: Normalized paddle-ball position difference - Resistive Random mode.

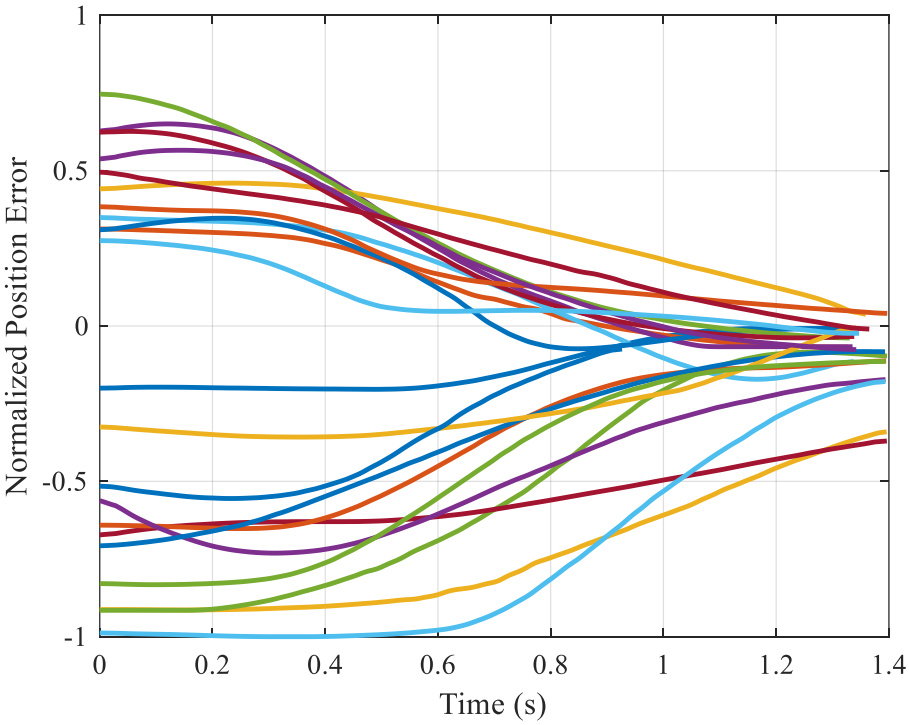

Figure 7.18: Normalized paddle-ball position difference - Resistive Gain mode.

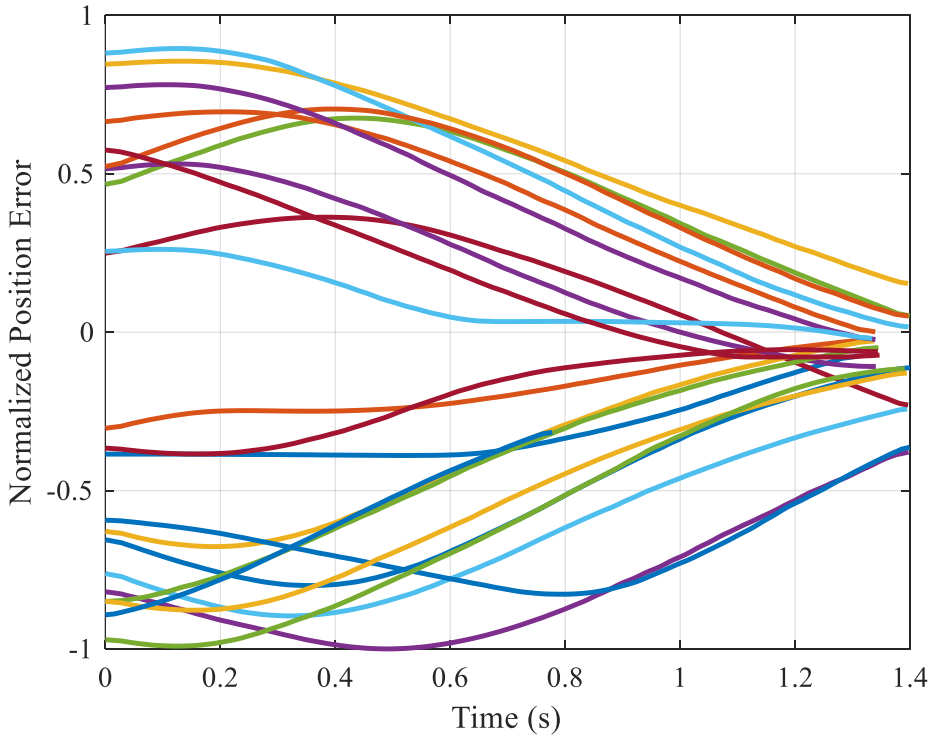

Figure 7.19: Normalized paddle-ball position difference - Passive mode.

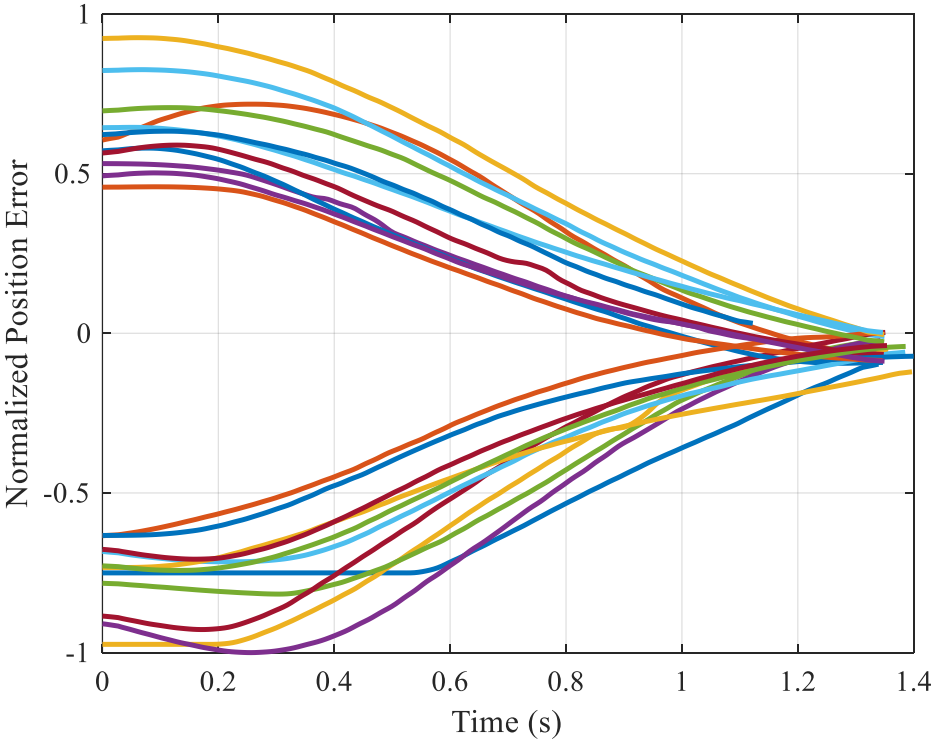

Figure 7.20: Normalized paddle-ball position difference - Assistive Gain mode.

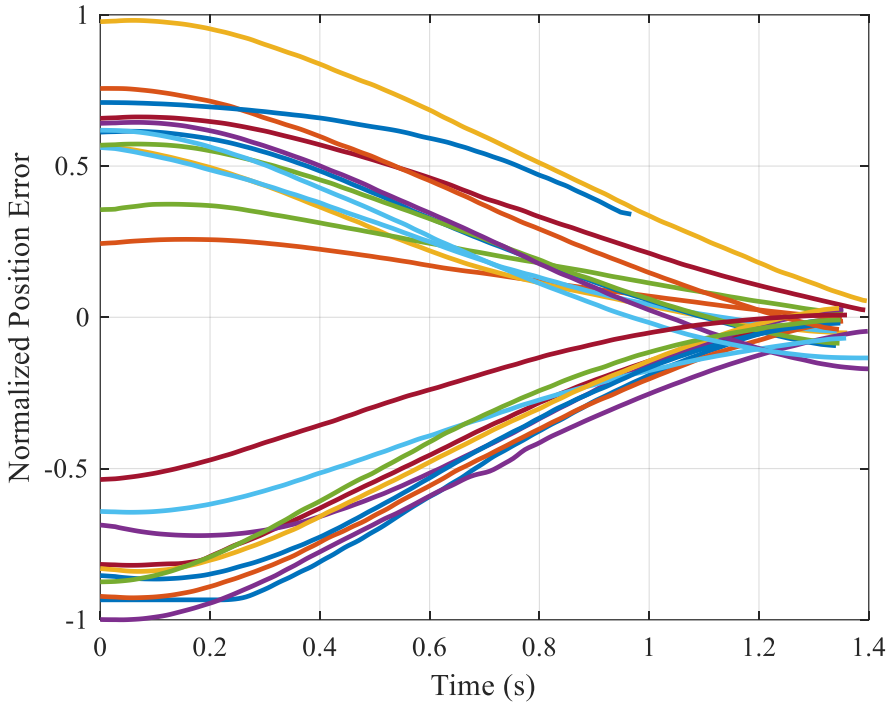

Figure 7.21: Normalized paddle-ball position difference - Assistive PD-based mode.

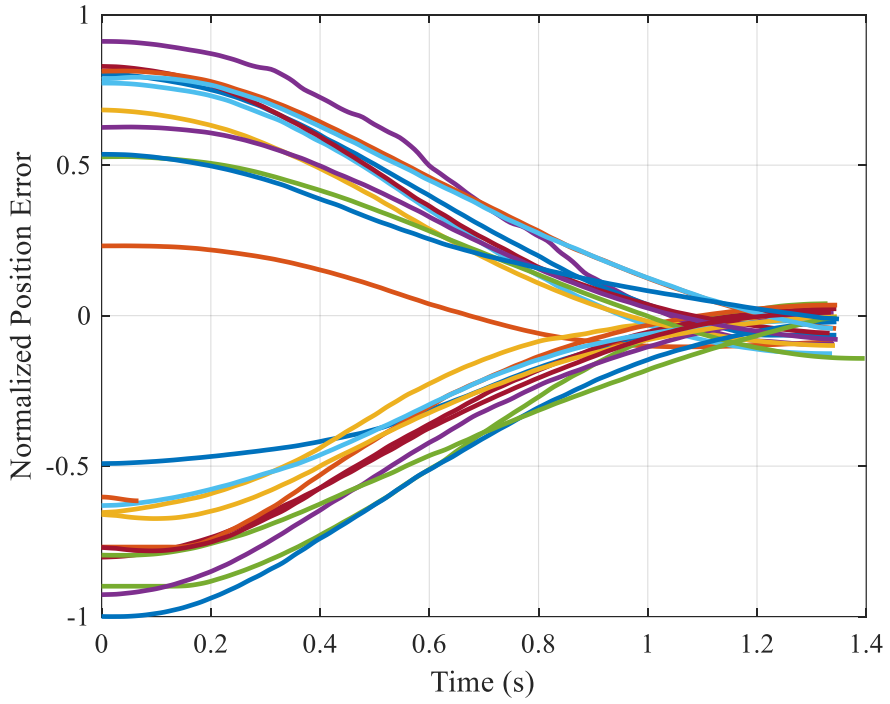

Figure 7.22: Normalized paddle-ball position difference - Assistive Gain-PD-based mode.

Another factor to analyze is torque. The torques applied by the robot and the participants produce valuable insight into the system's effectiveness. Passive mode torques, during the sessions in each joint of the robot, are depicted in Figures 7.23 and 7.24. Figure 7.23, reveals the torque applied by the participant to the robot’s joints. Lastly, Figure 7.24 captures the total torques generated by the robot in each joint, combining both gravity compensation and resistance or assistance torques.

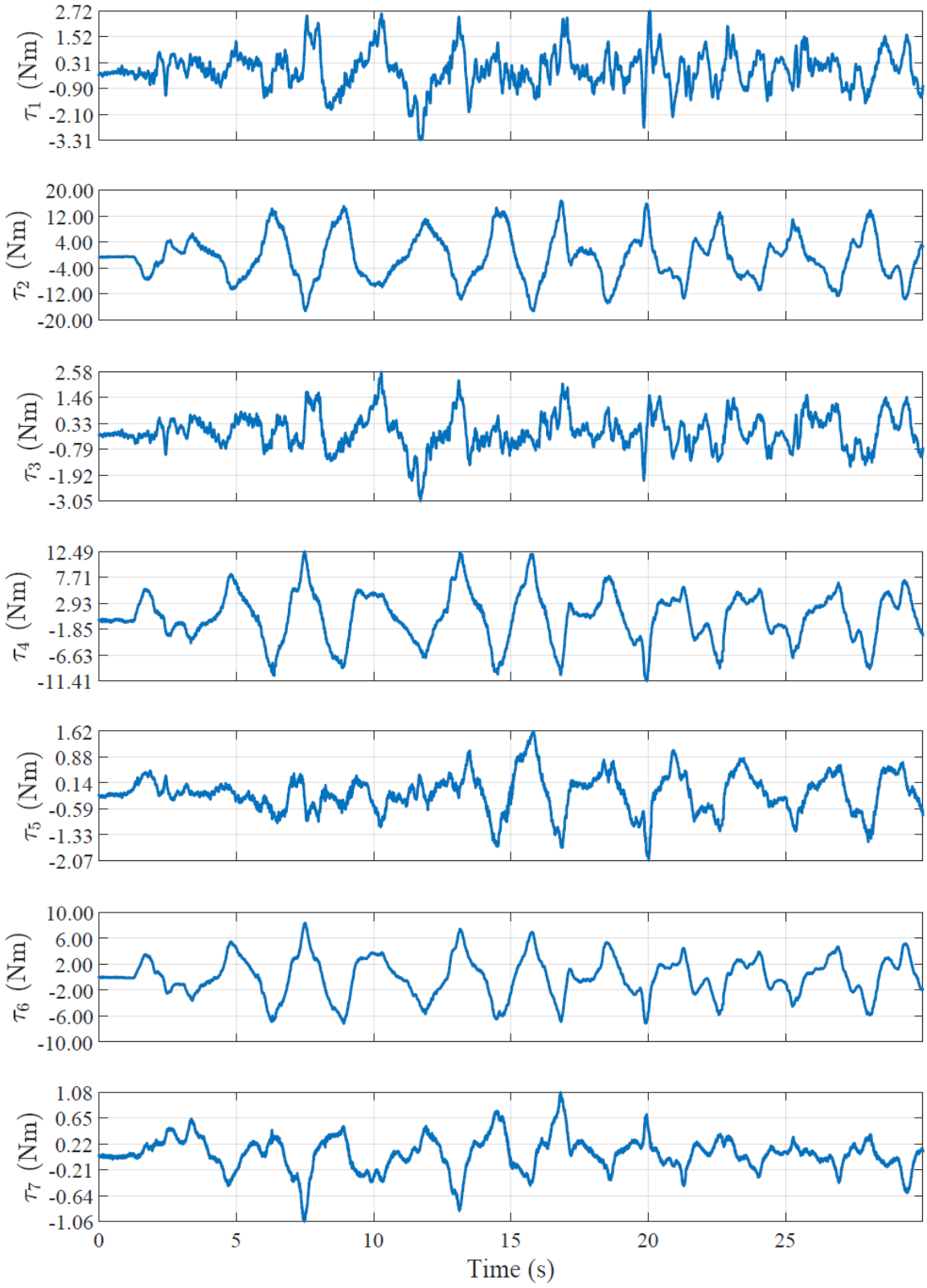

Figure 7.23: Participant-generated torques - Passive mode.

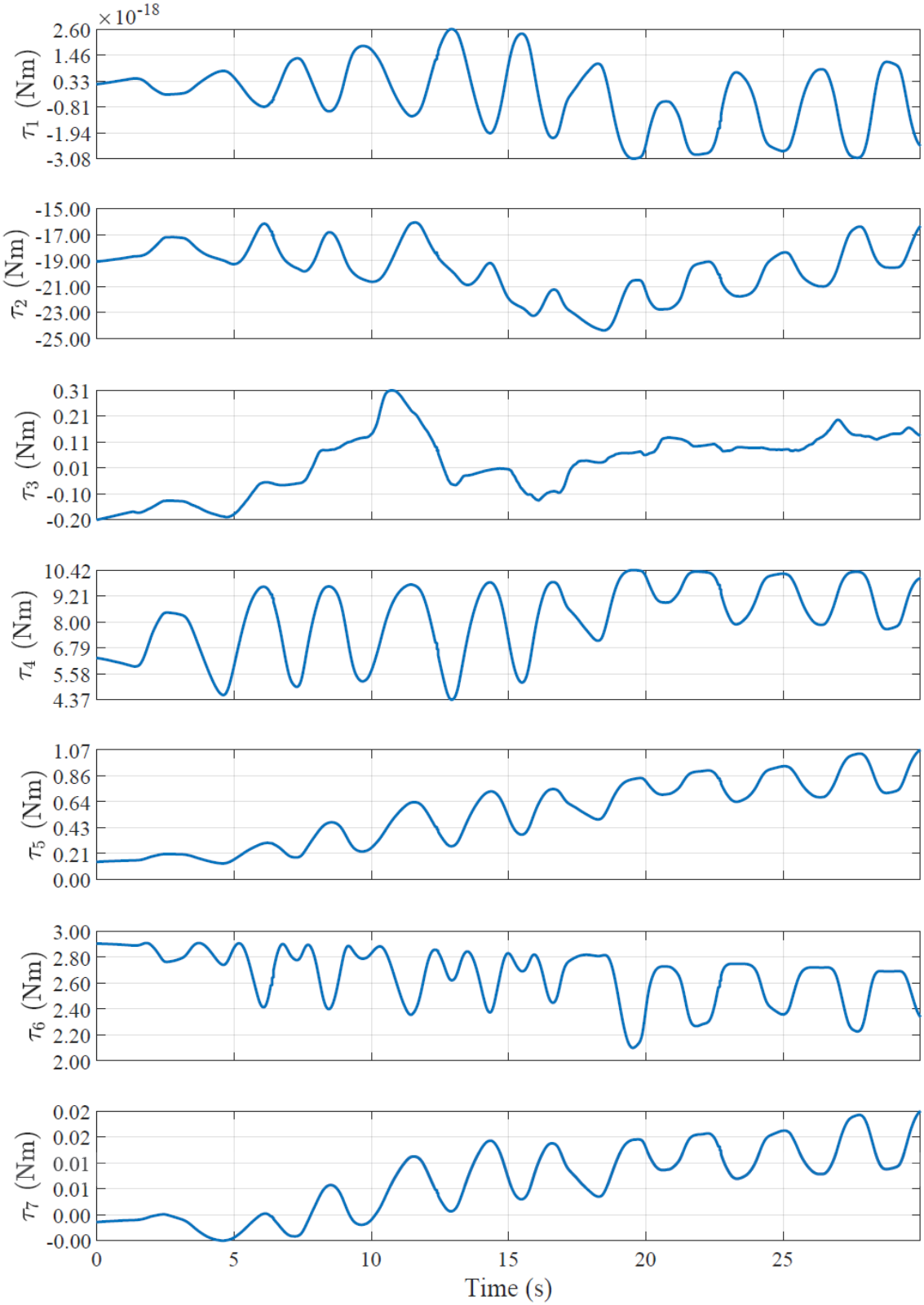

Figure 7.24: Robot-generated torques - Passive mode.

To compare torques across different modes, the human-generated torques for each mode are grouped by joint and plotted together to enable direct comparison. This provides a clear understanding of system behaviour in different modes. These plots are shown in Figures 7.25 to 7.31.

Torque profiles across seven joints vary significantly between assistive, passive, and resistive modes. Assistive modes display minimal torque generated by the participant, indicating adequate support and reduced effort. Passive modes exhibit a moderate amount of torque generated by the participant compared to Resistive and Assistive modes, reflecting natural movement without assistance. In Resistive modes, the participant applies a higher torque than in other modes, indicating greater engagement during the session. Overall, torque increases from assistive to resistive modes, demonstrating how each control strategy impacts physical demand.

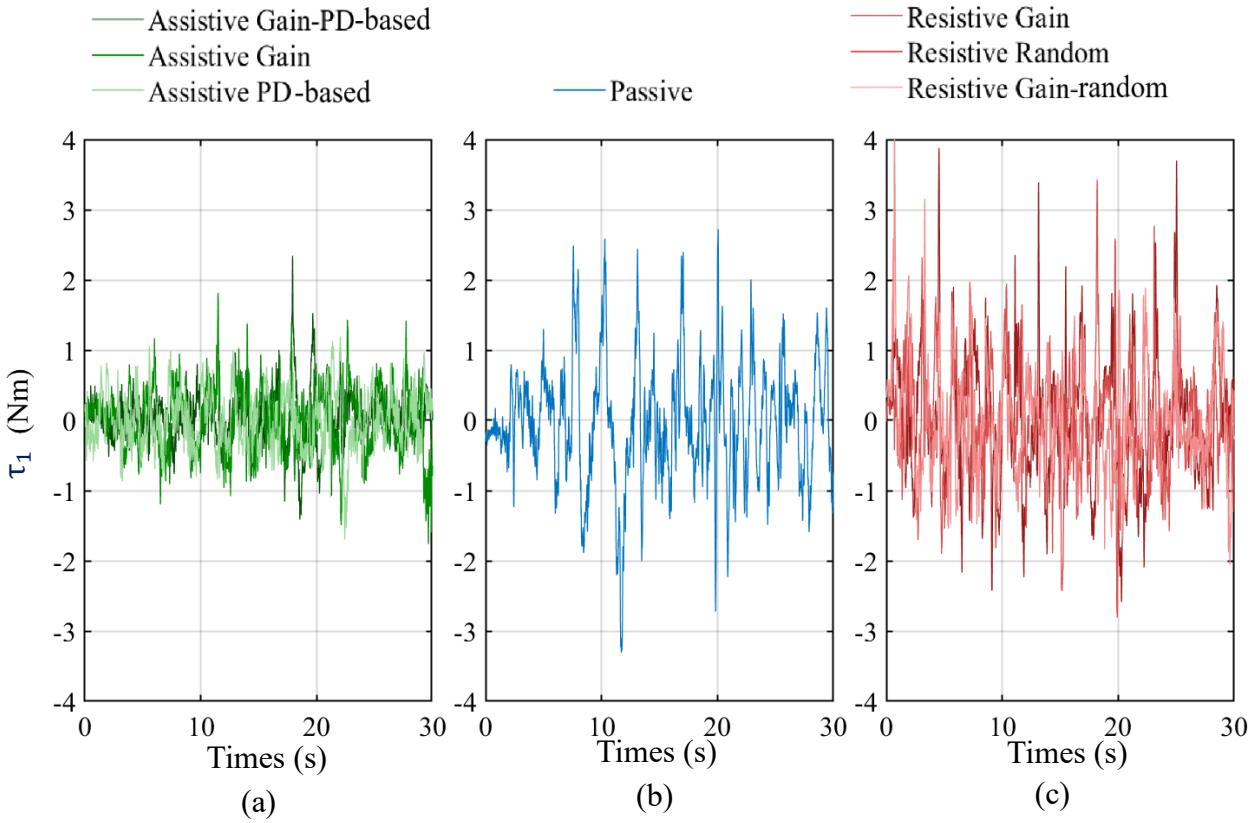

Figure 7.25: Human-generated torques applied to joint 1 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

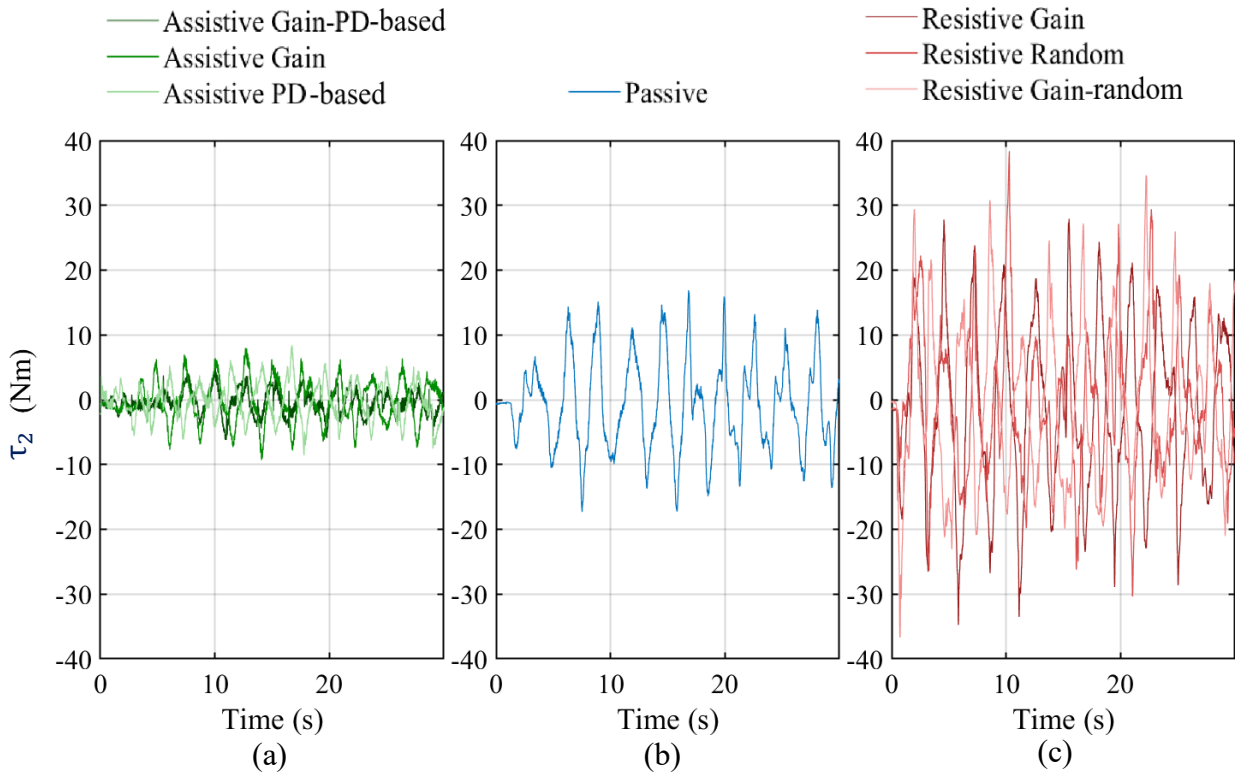

Figure 7.26: Human-generated torques applied to joint 2 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

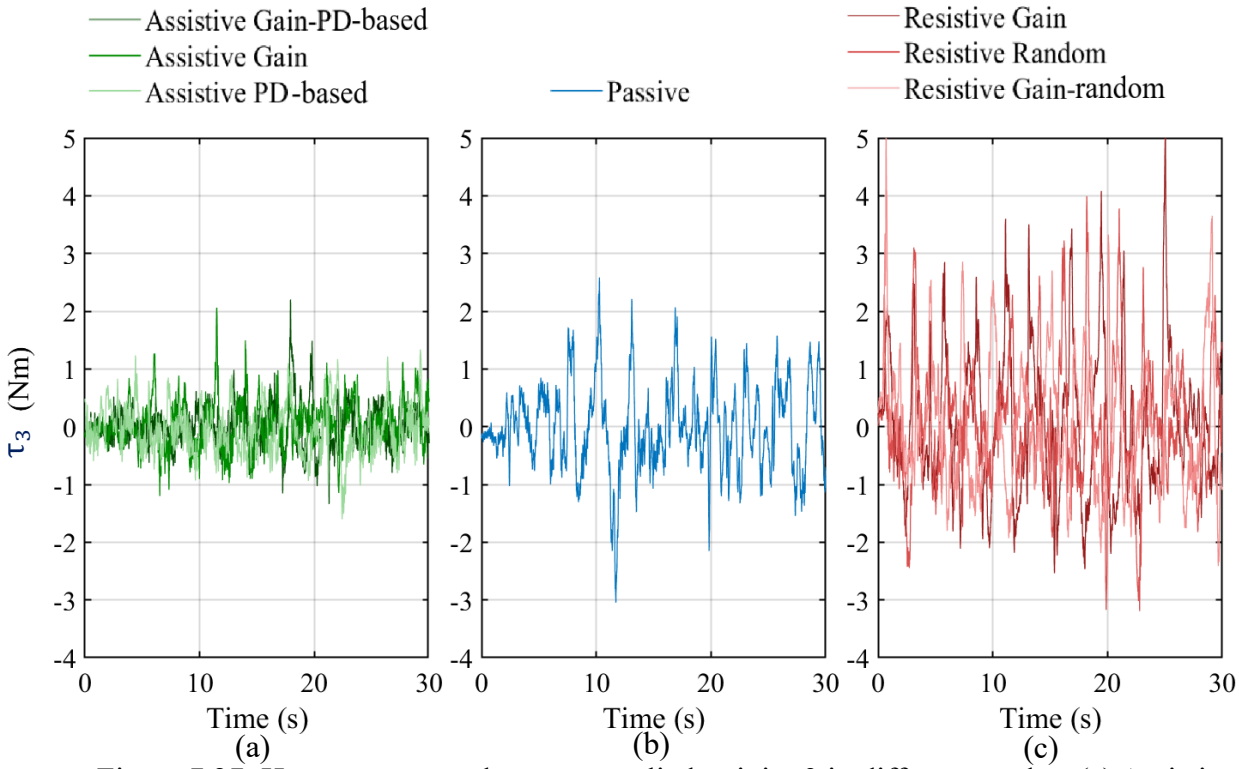

Figure 7.27: Human-generated torques applied to joint 3 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

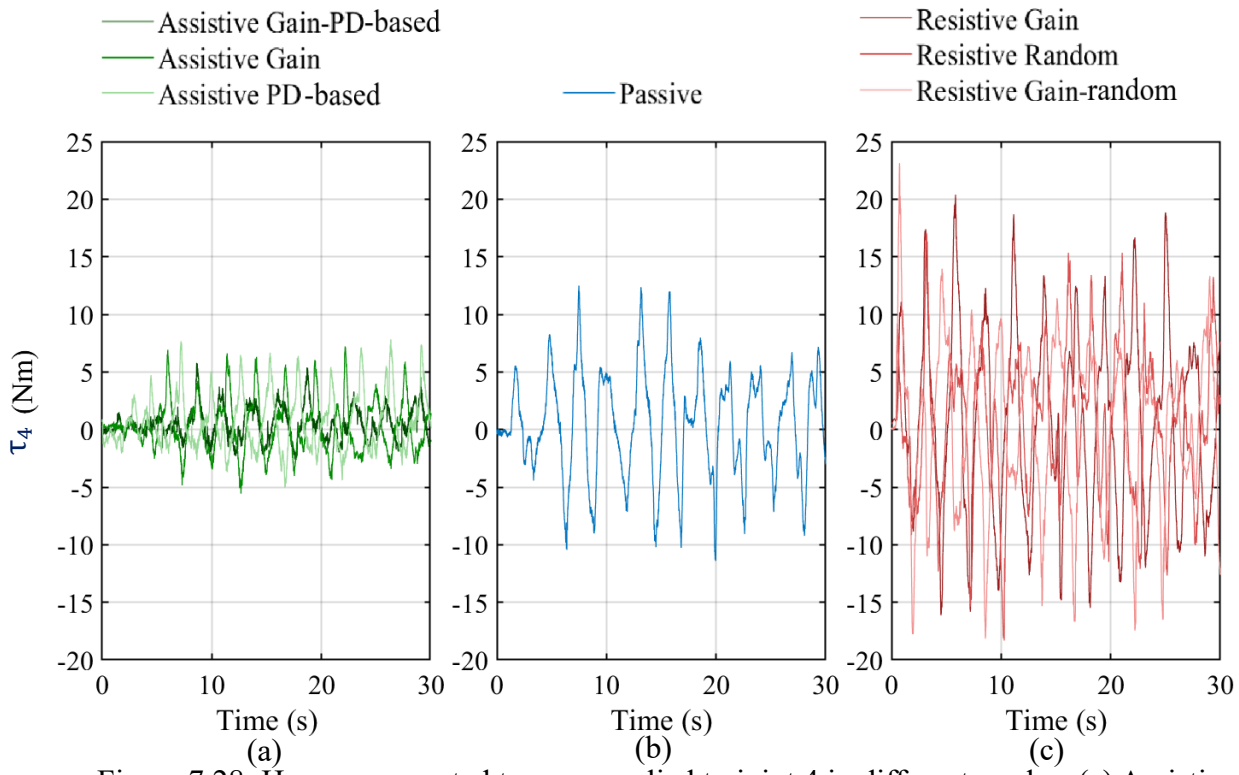

Figure 7.28: Human-generated torques applied to joint 4 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

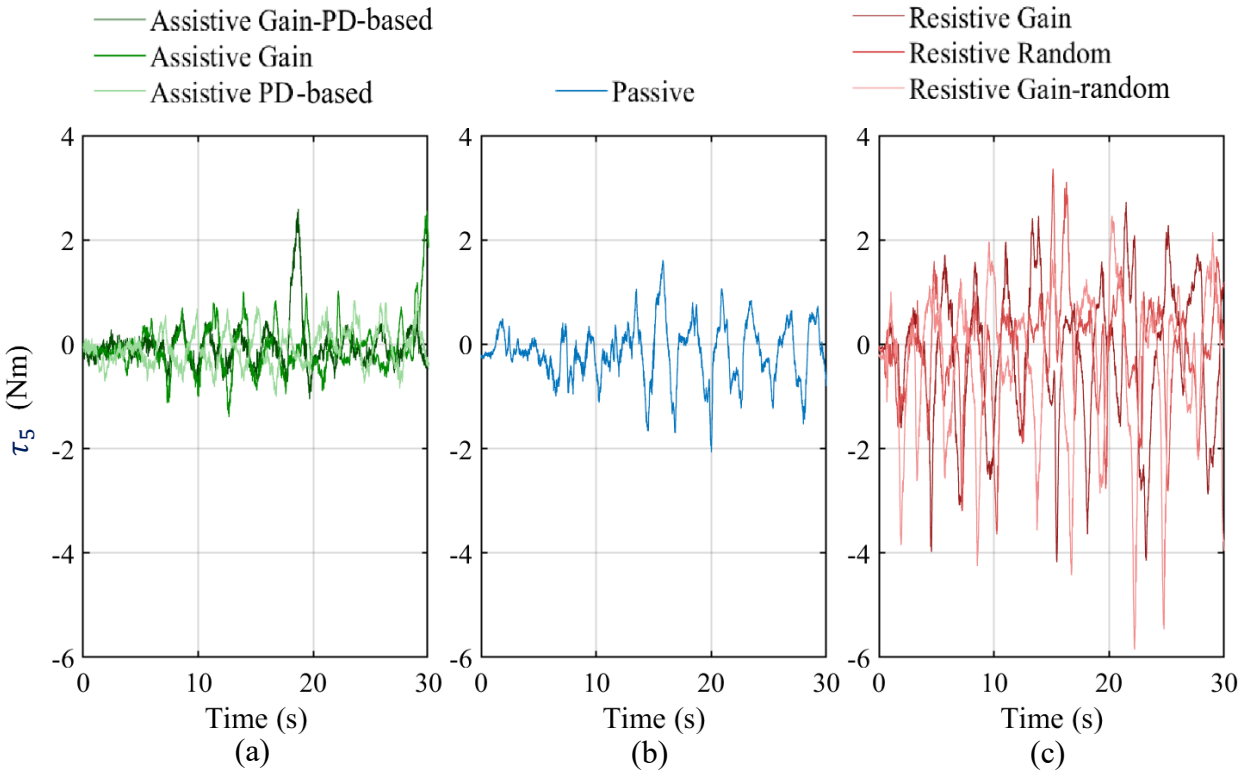

Figure 7.29: Human-generated torques applied to joint 5 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

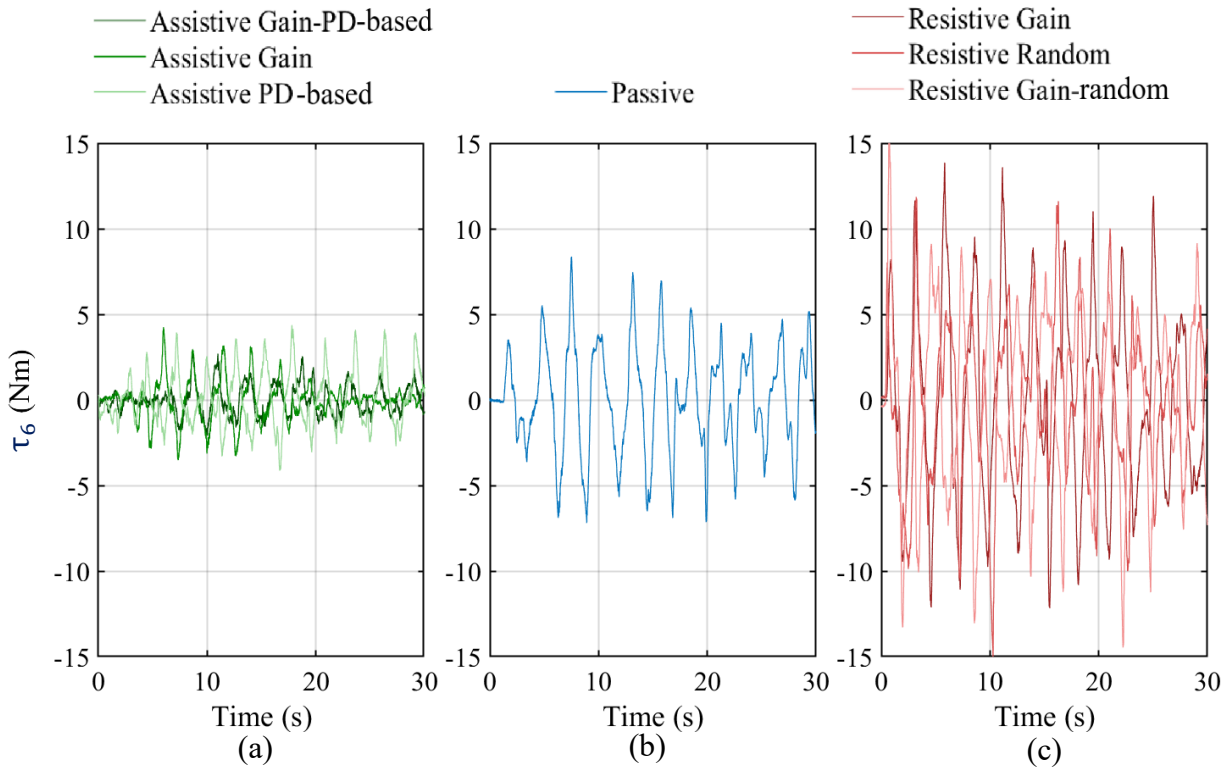

Figure 7.30: Human-generated torques applied to joint 6 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

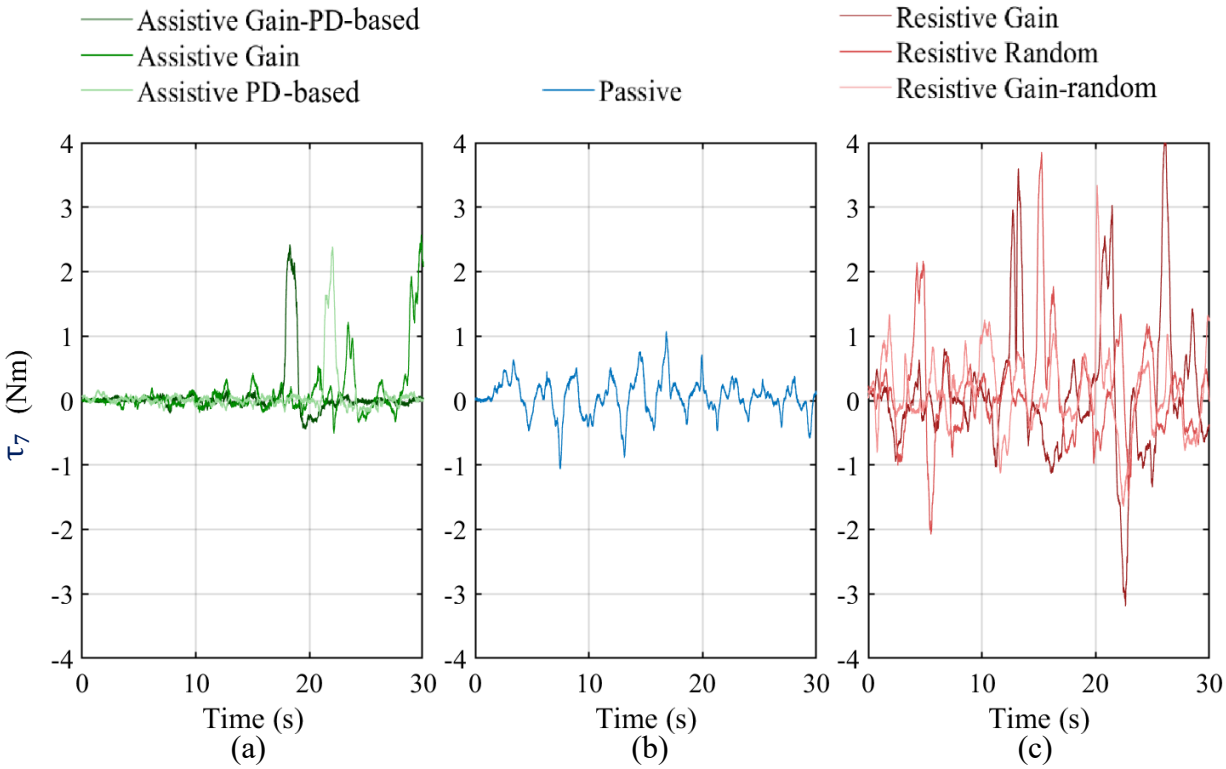

Figure 7.31: Human-generated torques applied to joint 7 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

In Figure 7.32 Standard Deviation (STD) and the mean of each torque signal from Figures 7.25 to 7.31 are depicted for even more precise comparison between modes. Assistive modes demonstrate consistently low torque magnitudes with minimal variability, indicating stable, low-effort interaction. Conversely, resistive modes show significantly higher variability, particularly in Joints 2, 4, and 6, reflecting increased effort and inconsistent user responses. Passive mode exhibits moderate variability, positioned between assistive and resistive extremes. Overall, resistive modes produce larger torque fluctuations, while assistive modes offer greater stability across all joints.

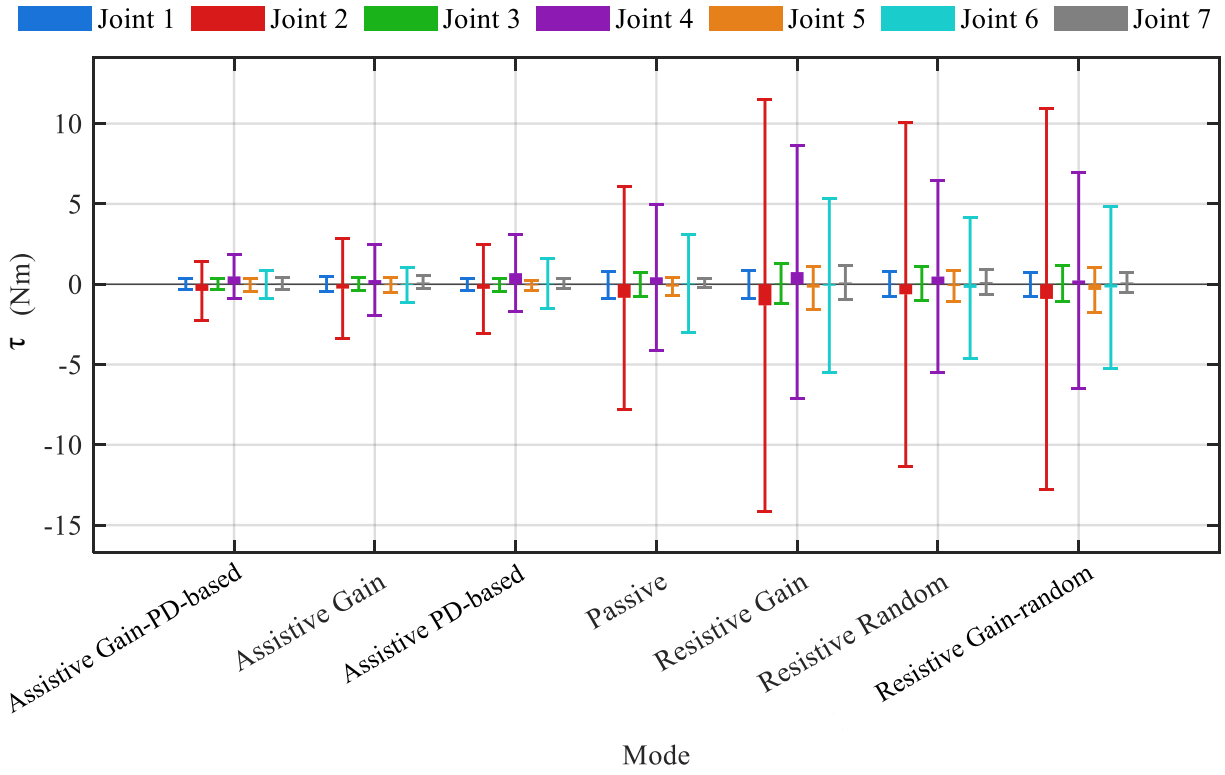

Figure 7.32: STD (thin bars) and Mean (thick bars) of Human-generated torques in each mode on every joint.

Figures 7.33 to 7.39 depict robot-generated torques across joints, reflecting their control objectives. Assistive modes produce smooth, low magnitude torques to reach targets and aid voluntary movement with minimal oscillations. High-load joints (2, 4, and 6) show increased yet stable torques. Passive mode maintains posture with minimal torque variation. Resistive modes generate larger, fluctuating torques, with the Random mode introducing high-frequency disturbances. These patterns validate the control framework's ability to deliver supportive, minimal, or challenging forces suited to various rehabilitation scenarios.

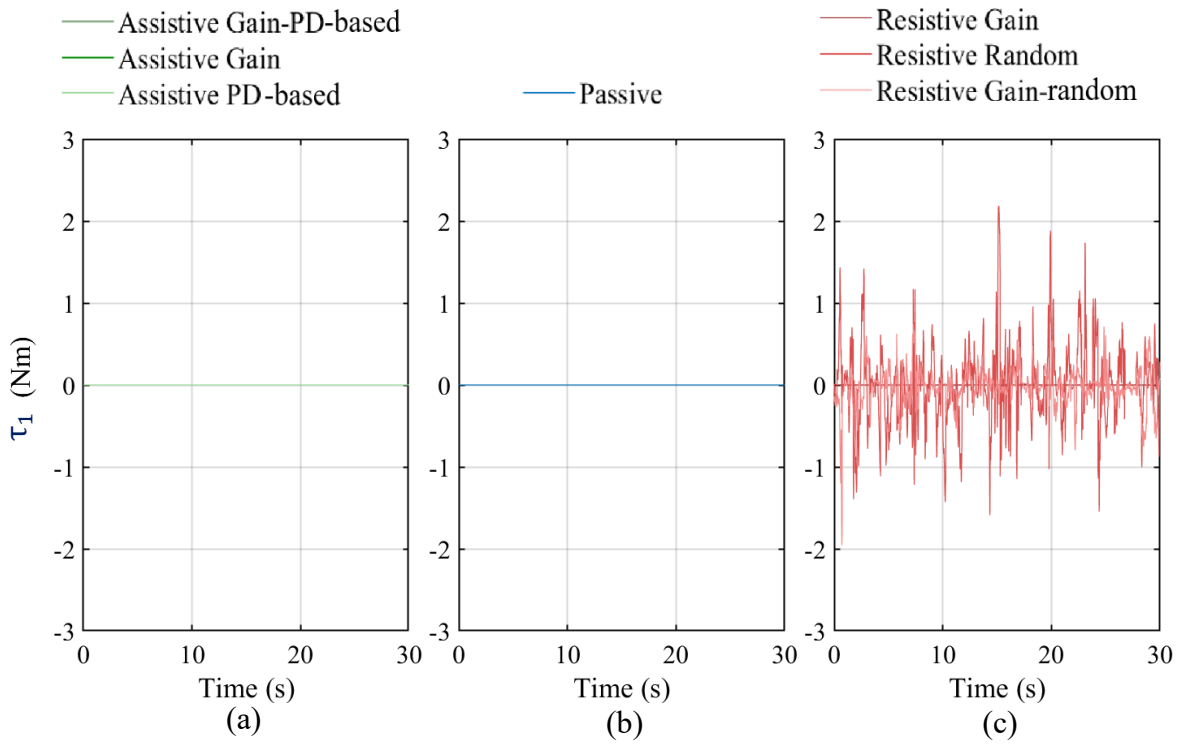

Figure 7.33: Robot-generated torques applied to joint 1 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

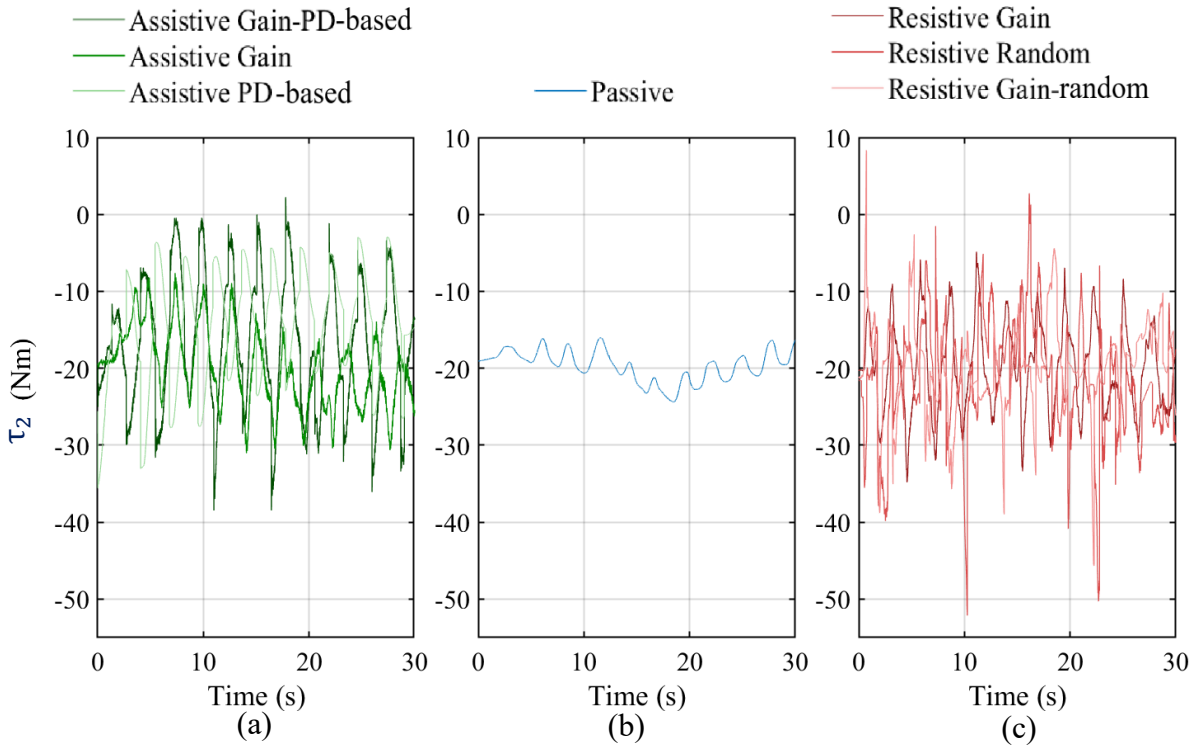

Figure 7.34: Robot-generated torques applied to joint 2 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

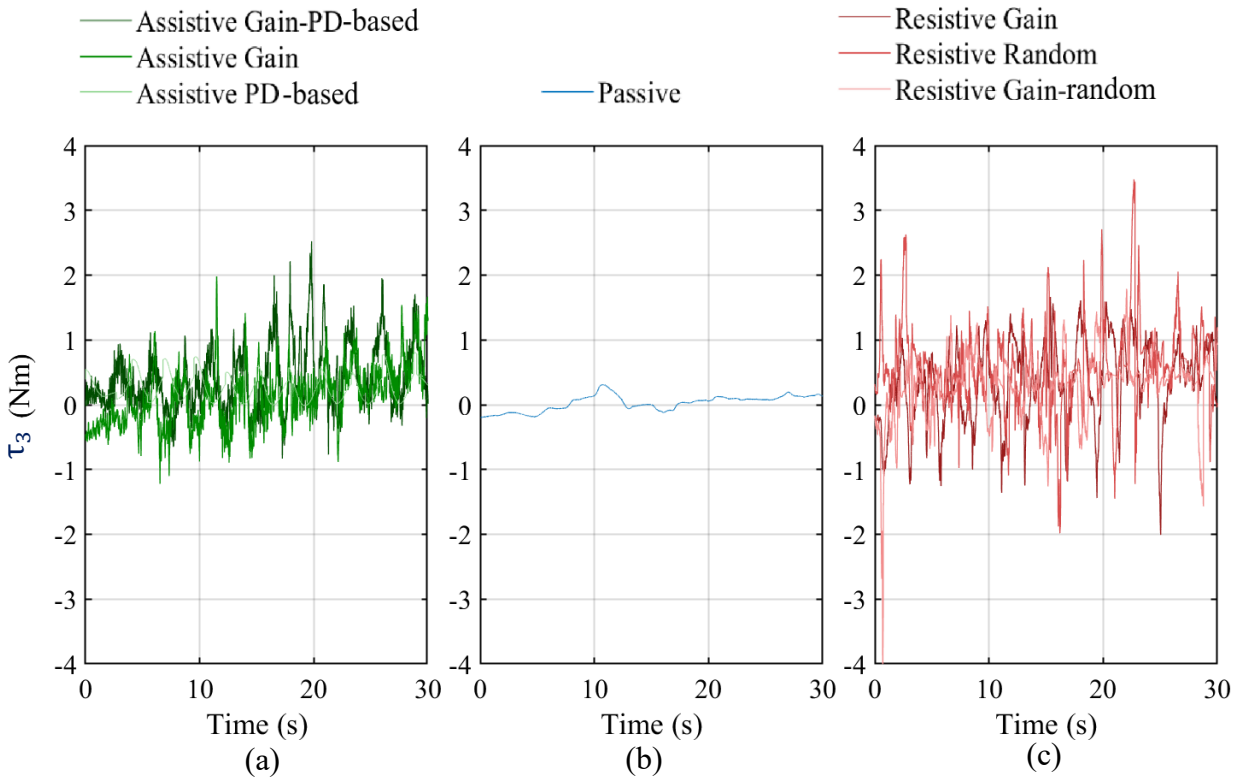

Figure 7.35: Robot-generated torques applied to joint 3 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

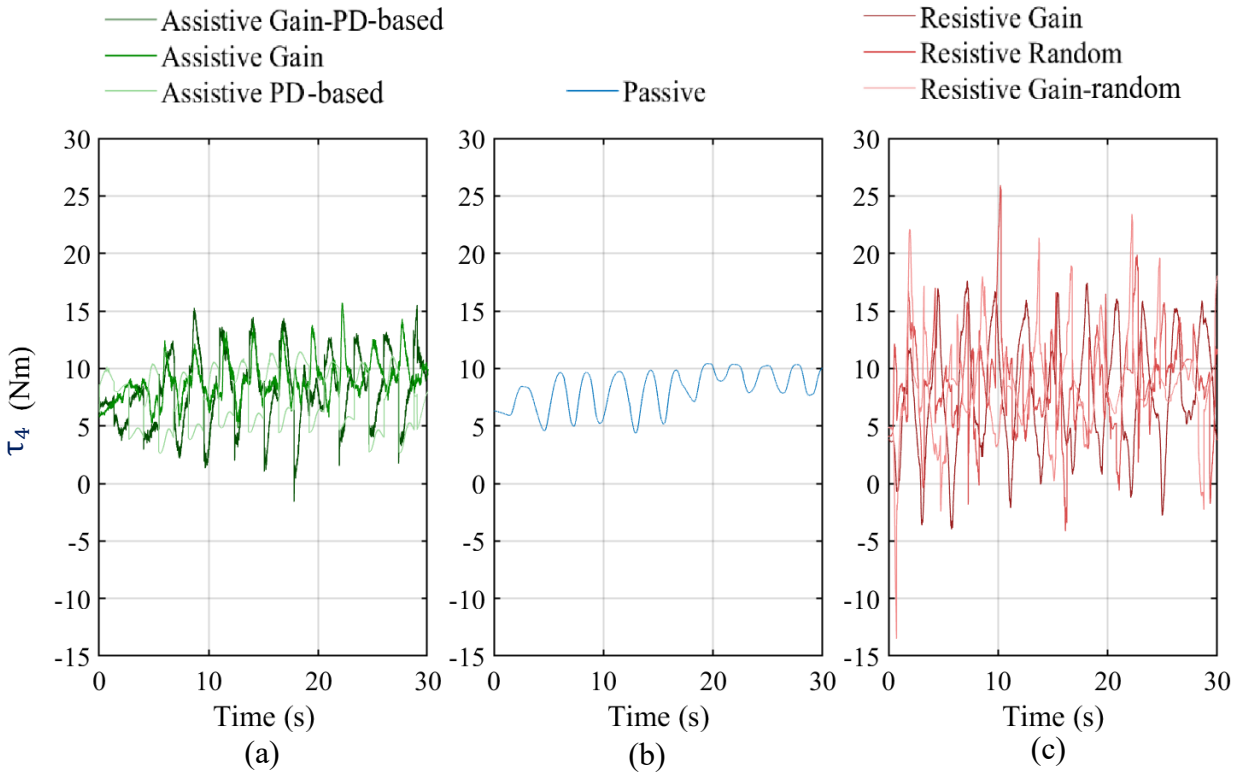

Figure 7.36: Robot-generated torques applied to joint 4 in different modes: (a) Assistive mode: (b) Passive mode; and (c) Resistive mode.

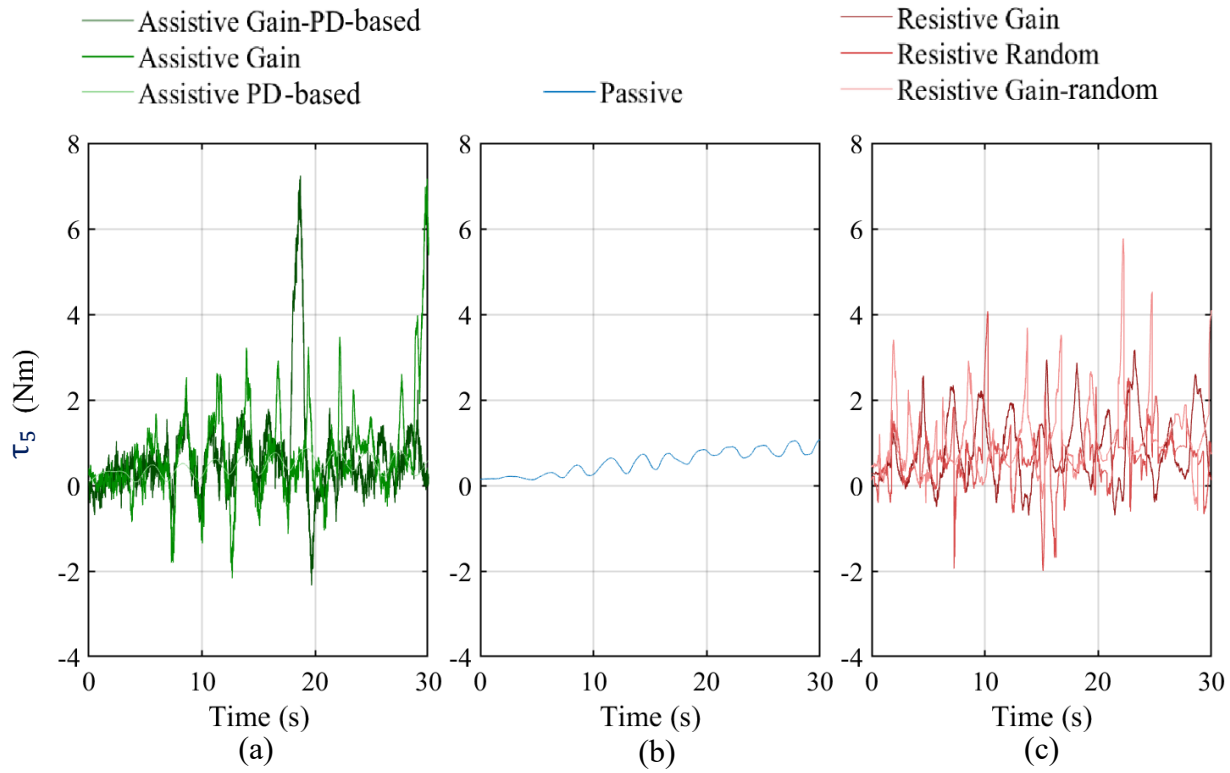

Figure 7.37: Robot-generated torques applied to joint 5 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

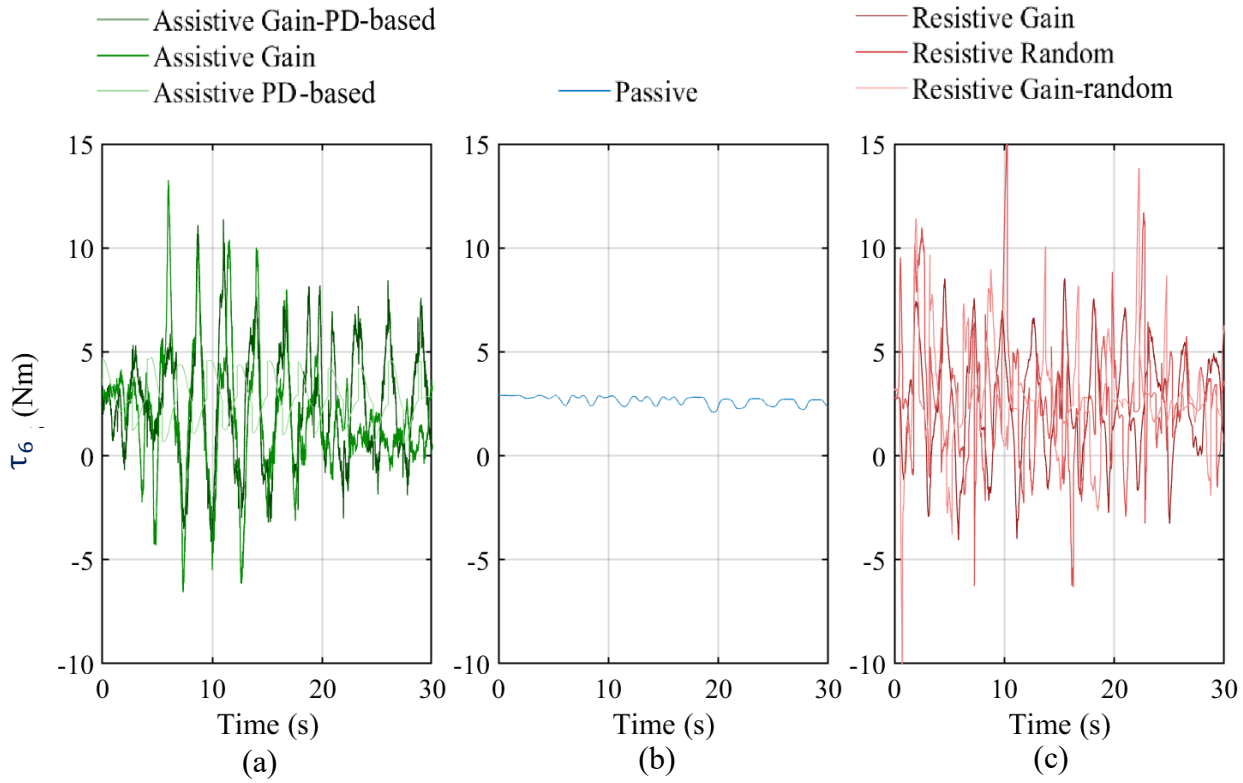

Figure 7.38: Robot-generated torques applied to joint 6 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

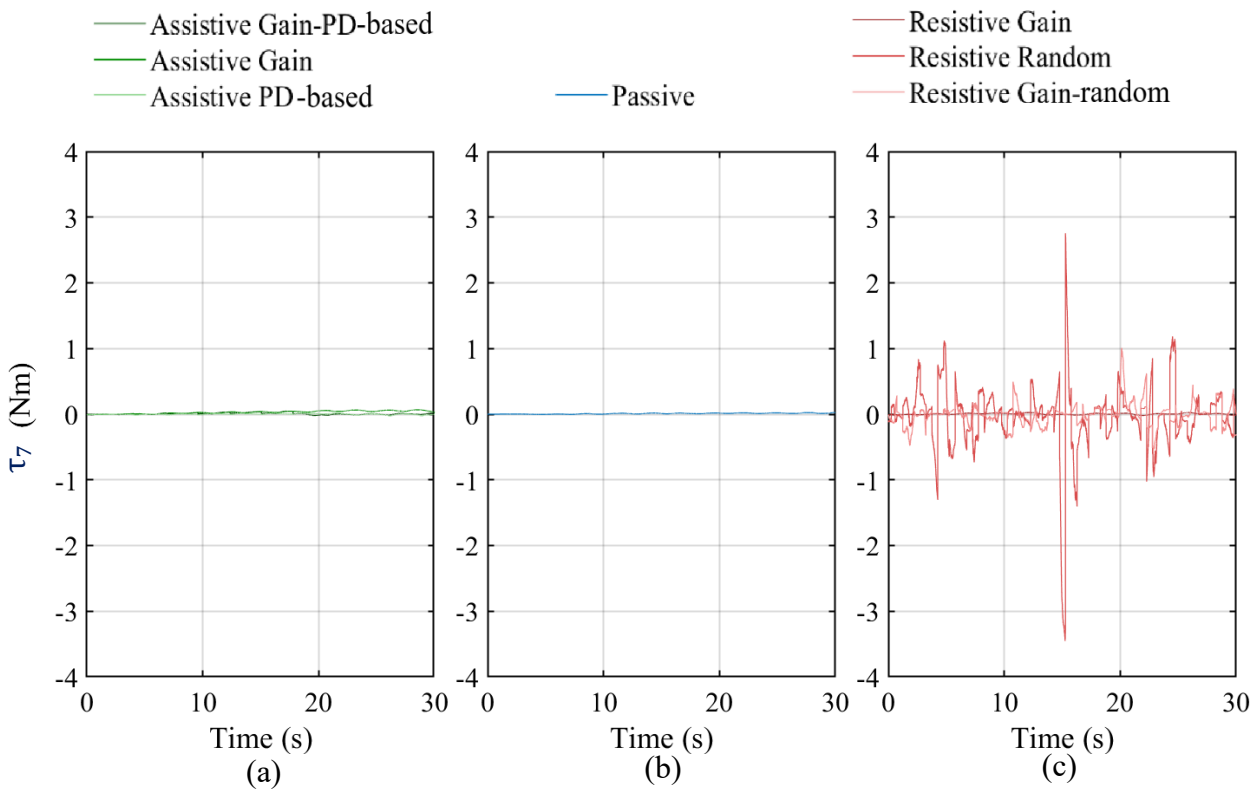

Figure 7.39: Robot-generated torques applied to joint 7 in different modes: (a) Assistive mode; (b) Passive mode; and (c) Resistive mode.

Figure 7.40 shows the STD and mean of robot-generated torques across all joints and modes. Most joints exhibit small mean torque values, as the task trajectory follows a periodic motion pattern that results in alternating torque demands over time. Joint 2 consistently shows the largest torque variance because the robot must counteract greater gravitational and dynamic loads at this joint. Assistive and Passive modes exhibit low variability, reflecting stable, supportive robot behavior. In contrast, Resistive modes generate larger torque fluctuations as the controller intentionally injects opposing or random forces to challenge the participant.

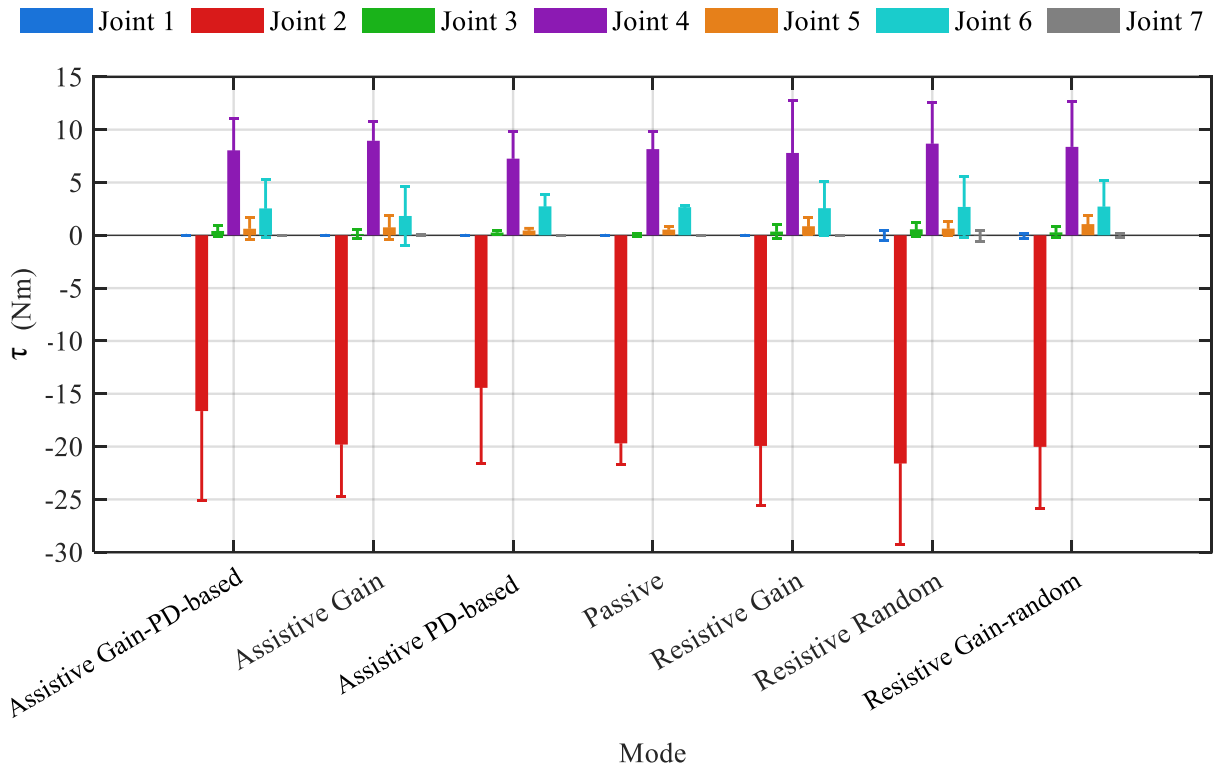

Figure 7.40: STD (thin bars) and mean (thick bars) of robot-generated torques in each mode on every joint of the robot.

Overall, these plots demonstrate that in resistive modes, the robot effectively applies controlled resistance, prompting the user to generate significant torque while performing consistent, repeatable joint movements. Also, show that in assistive modes, the robot effectively applies controlled assistance and reduces the user's contribution to the task during the interaction. This supports the system's goal of engaging users in active rehabilitation through targeted resistance.

To better understand human movement during gameplay and its impact on robotic interaction, a detailed analysis of upper body joint positions (over a 30-second session) was conducted using data from 3D vision-based tracking, as shown in Figure 7.41. Also, in Figure 7.42 (a) X-Z view, and (b) Y-Z view of Figure 7.41 are shown for better understanding. The joints of interest include the shoulder, elbow, wrist, and hip, and their positions were analyzed along the X, Y, and Z axes, with a primary focus on Z, as it directly maps to the paddle movement in the game. Each dot represent a snapshot of joint positions at 16 ms intervals.

With reference to Figure 7.41 the wrist joint (red dots) shows the most extensive vertical range (Z direction), as expected, since it plays the most direct role in controlling the paddle. The elbow (green dots) also demonstrates a significant vertical contribution, while the shoulder (blue dots) maintains a more centralized position with limited displacement. The hip joint (pink dots) remains relatively static, serving as a spatial reference point and confirming minimal lower body movement, which aligns with the upper-body-focused gameplay.

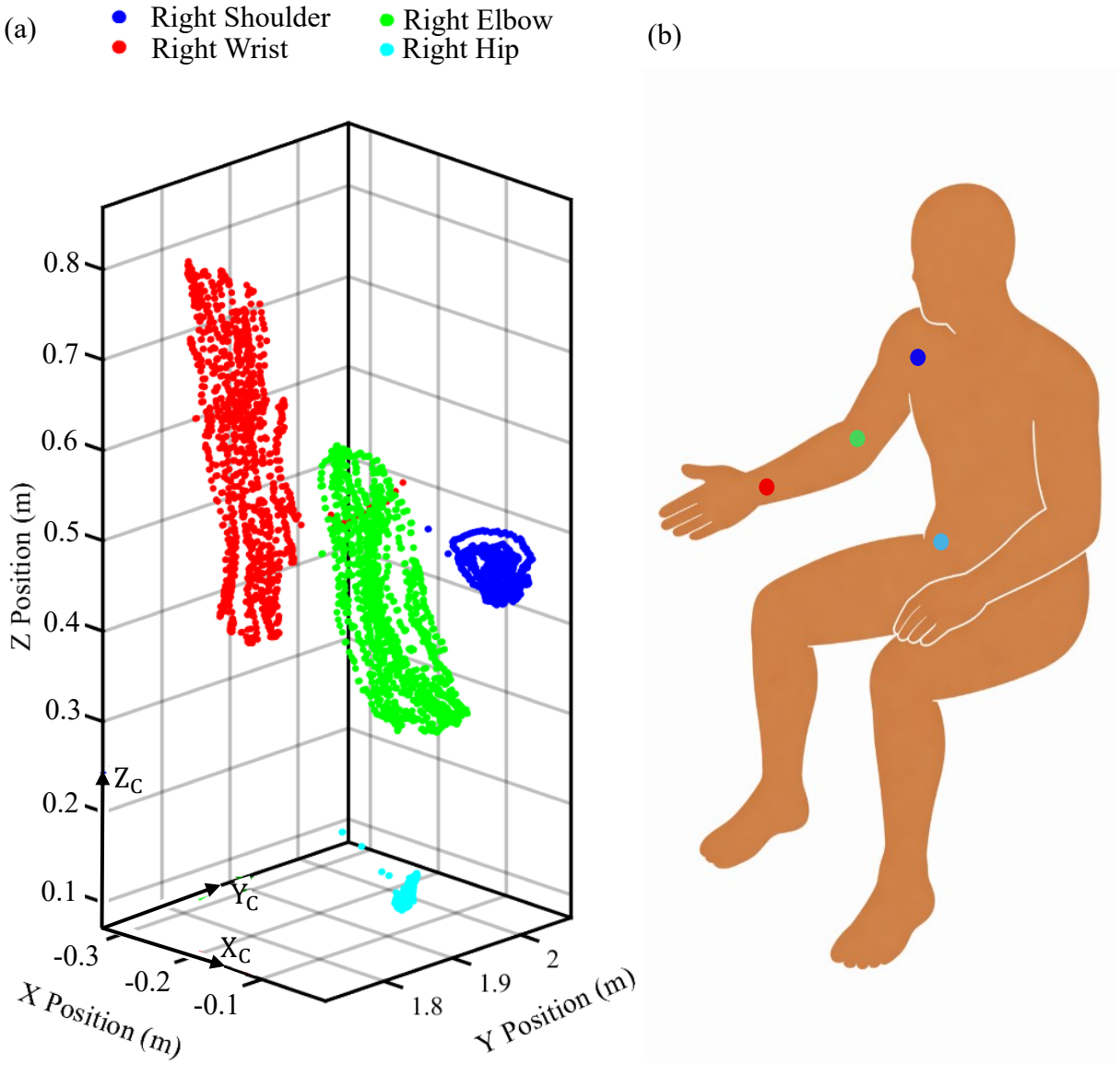

Figure 7.41: (a) Joints XYZ - 3D view; and (b) Targeted body keypoints of the participant.

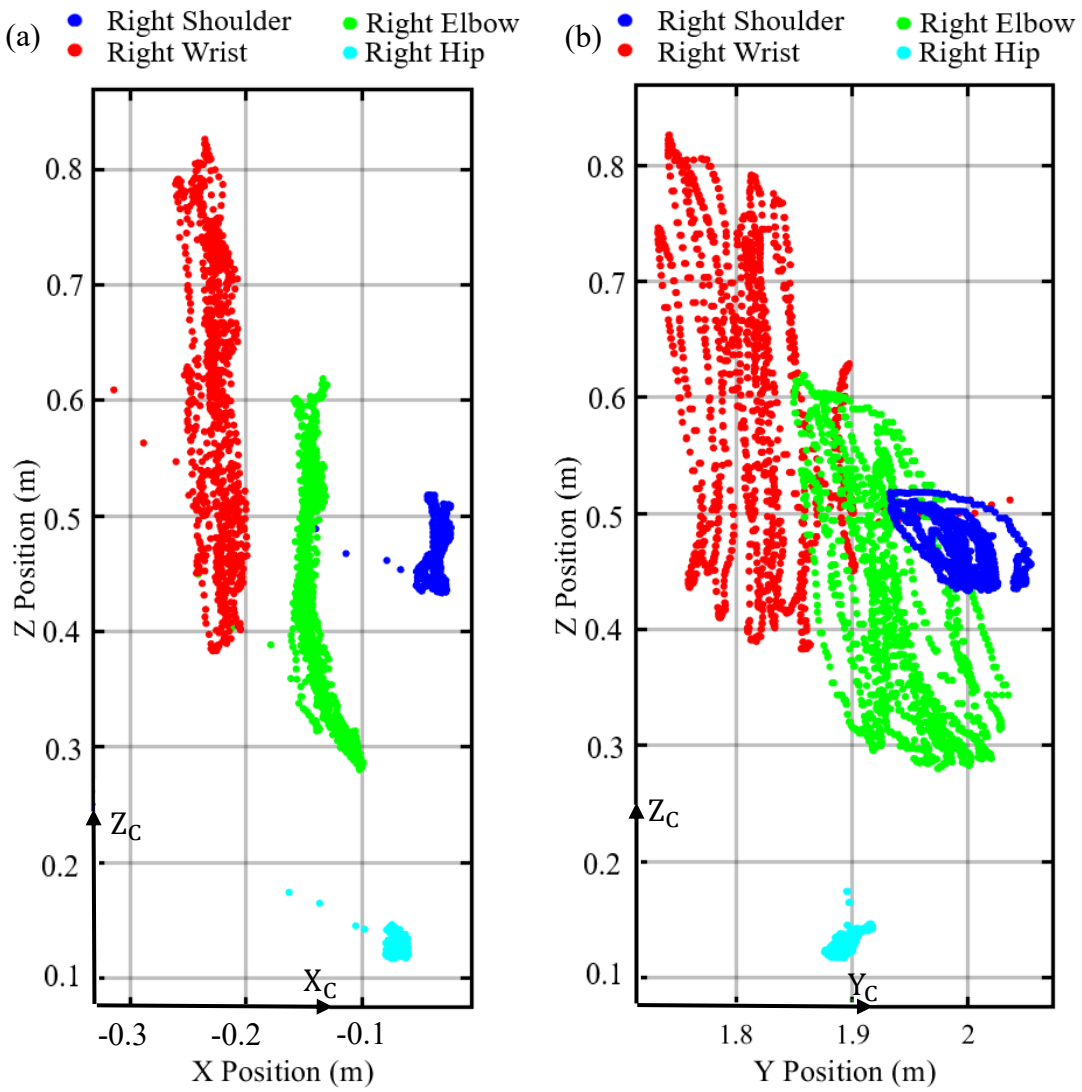

Figure 7.42: (a) Joints Z-X positions, (b) Joints Z-Y positions.

In Figures 7.43 to 7.49, the Z positions of the right hip, shoulder, elbow, wrist, and paddle calibrated position in the game are plotted over time in different modes, giving a clearer view of the dynamic movement patterns. Initially, the calibration of the paddle pixel position is conducted to align with a functional end-effector position along the Z axis, thereby demonstrating the integrity of wrist and paddle motion. In the plots, the wrist shows the greatest frequency and amplitude changes, mirroring the rhythmic nature of the game interaction as the participant attempts to catch the ball with the paddle. The elbow and shoulder follow similar periodic trends but with smaller amplitudes. The hip joint remains nearly flat, with negligible movement. This analysis confirms that the participant mainly engaged in shoulder flexion-extension and elbow-wrist coordination throughout the session, which are essential for vertical paddle control.

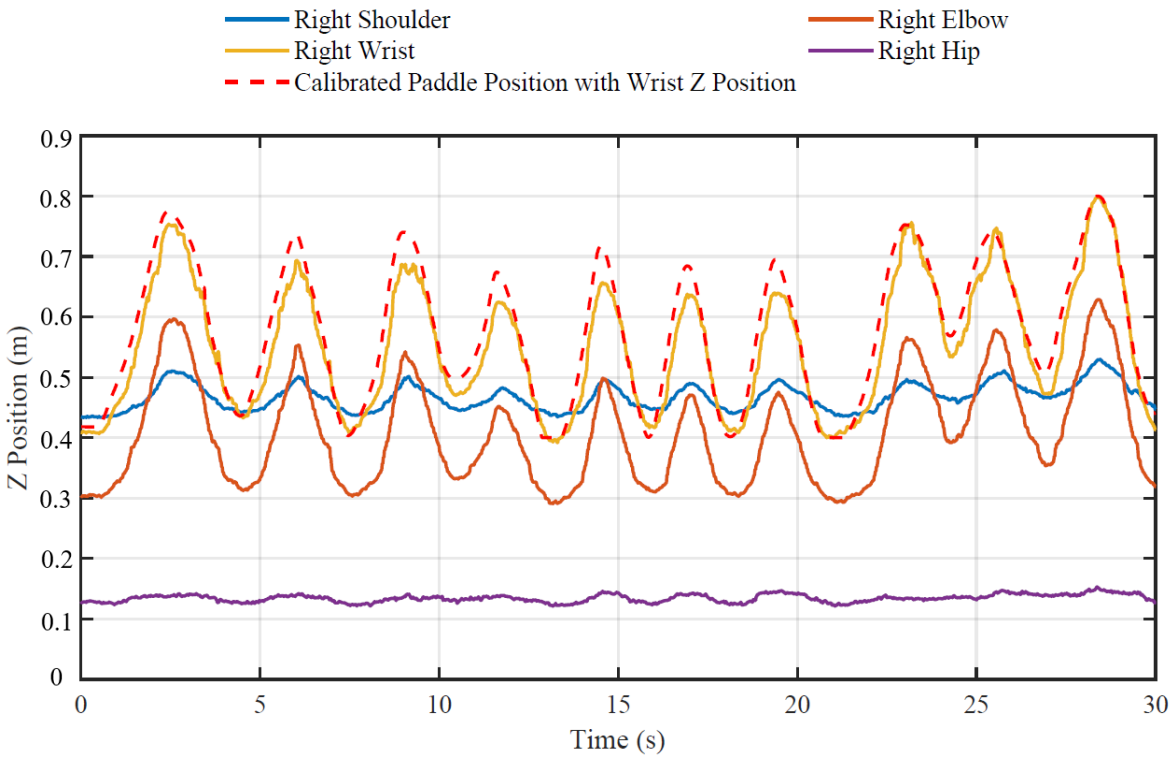

Figure 7.43: Body joints Z position over time, Resistive gain random mode.

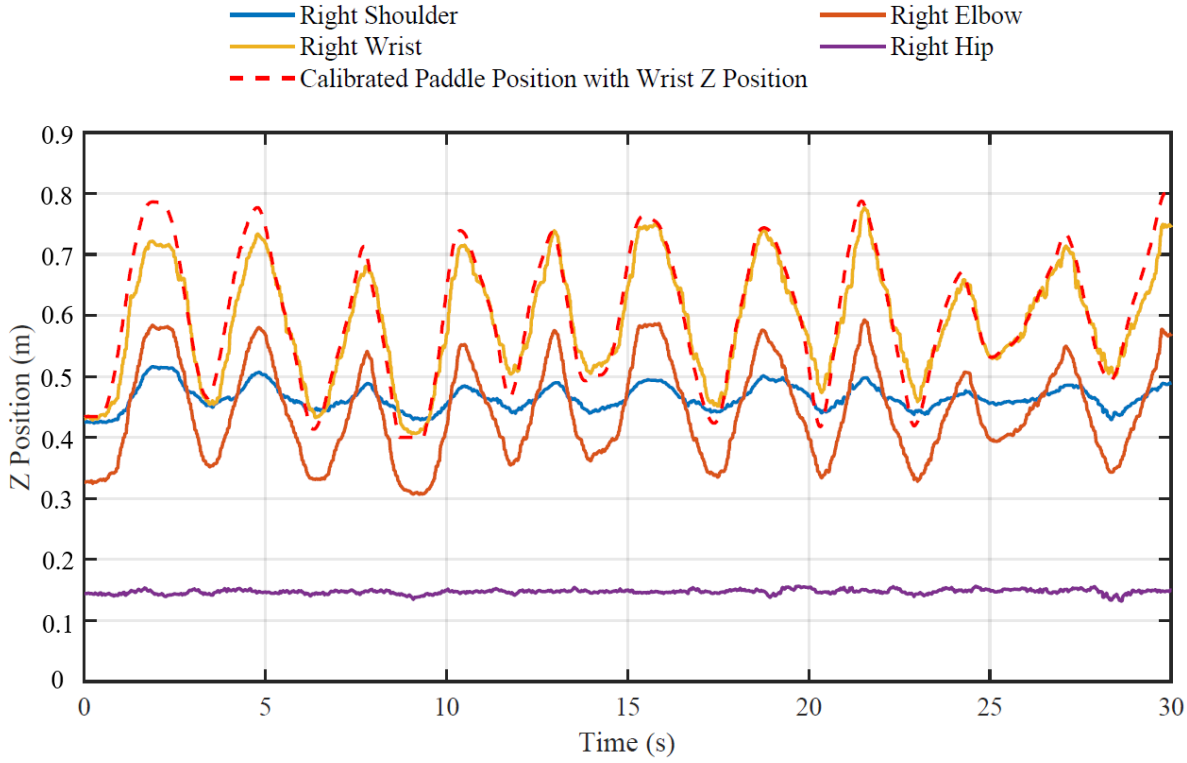

Figure 7.44: Joints Z-time - Resistive Random mode.

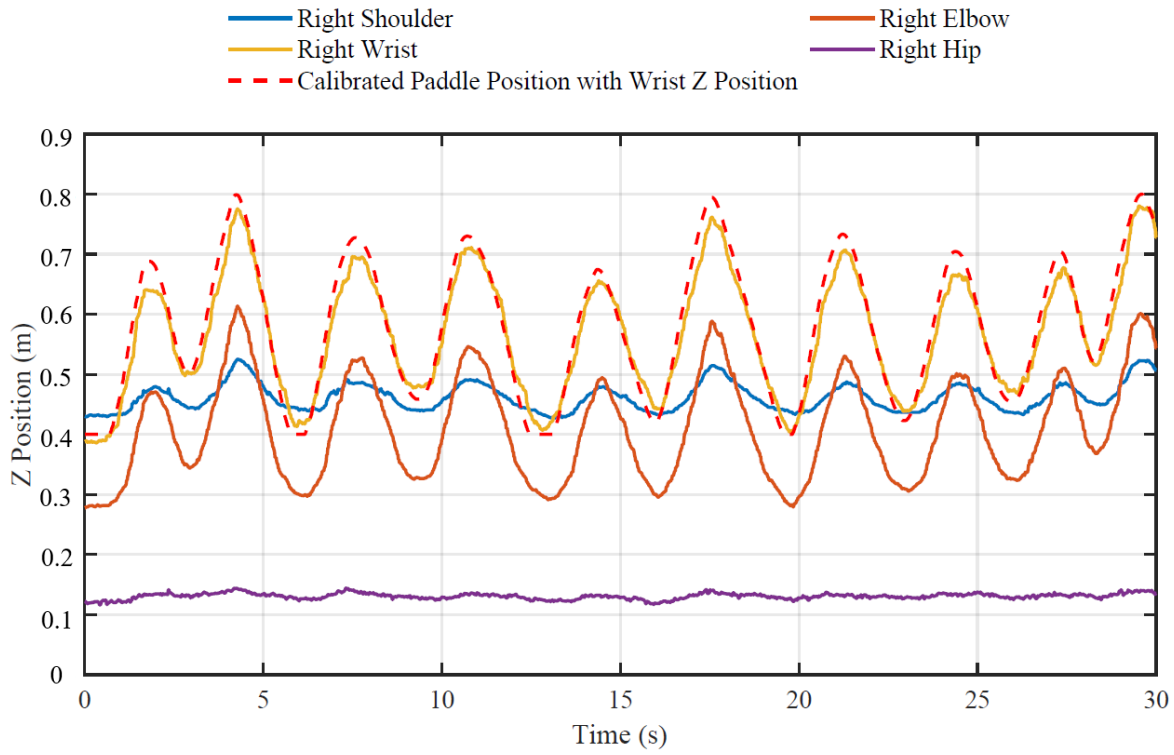

Figure 7.45: Joints Z-time - Resistive Gain mode.

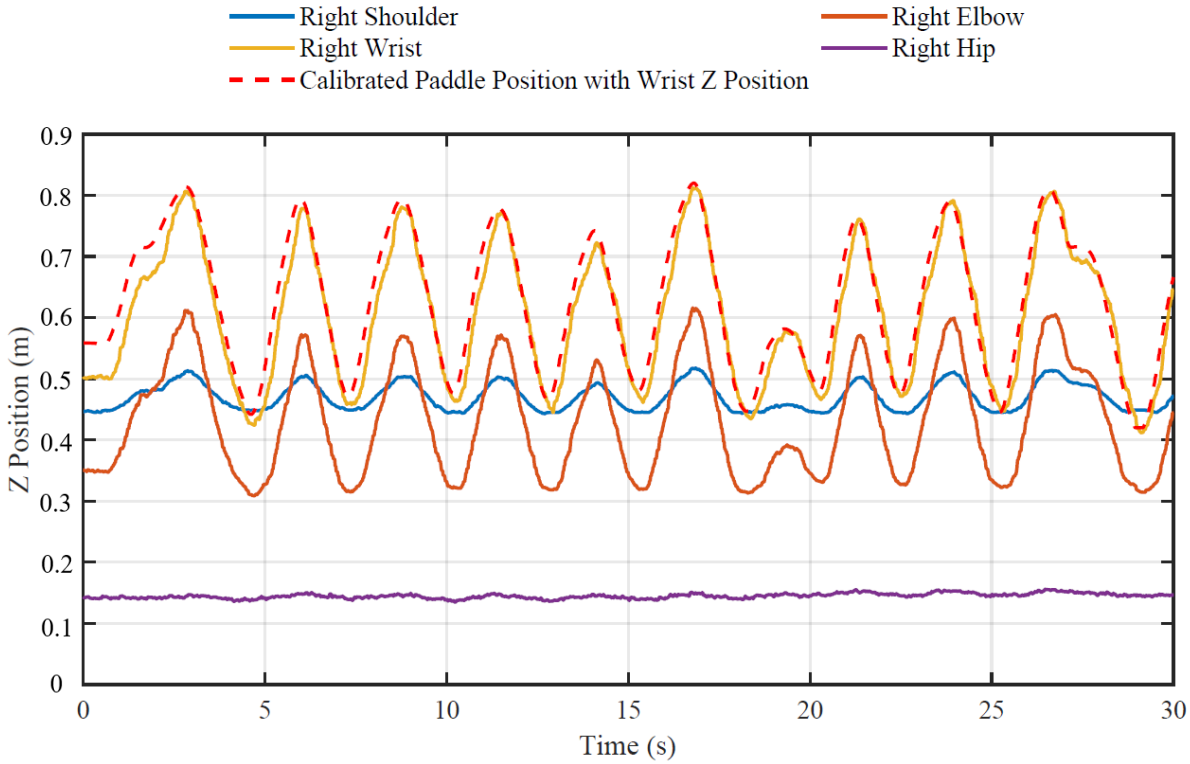

Figure 7.46: Joints Z-time - Passive mode.

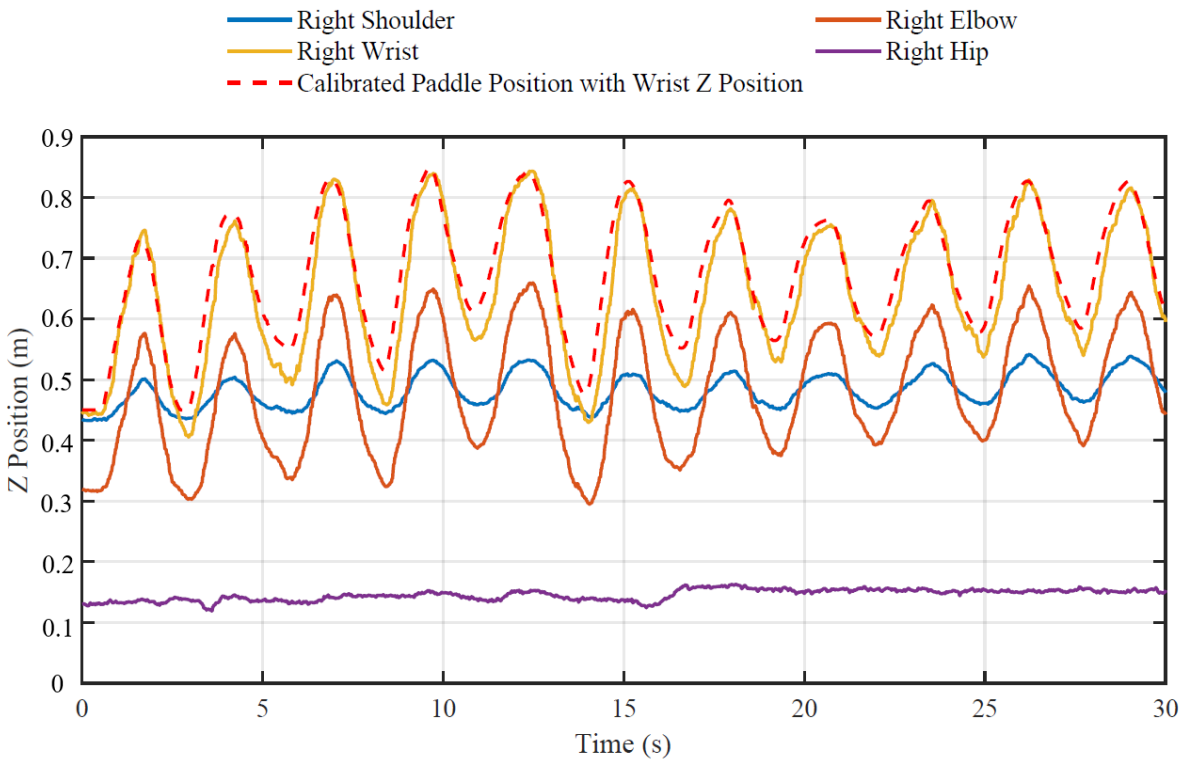

Figure 7.47: Joints Z-time - Assistive Gain mode.

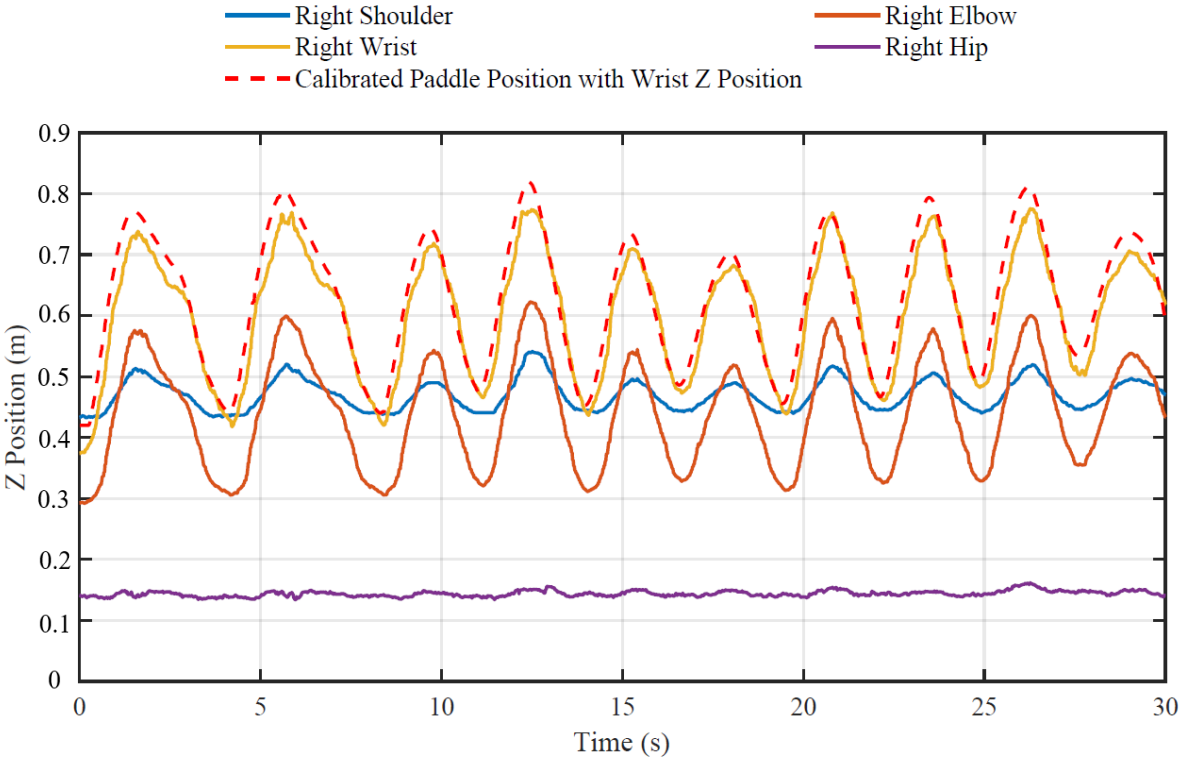

Figure 7.48: Joints Z-time - Assistive PD mode.

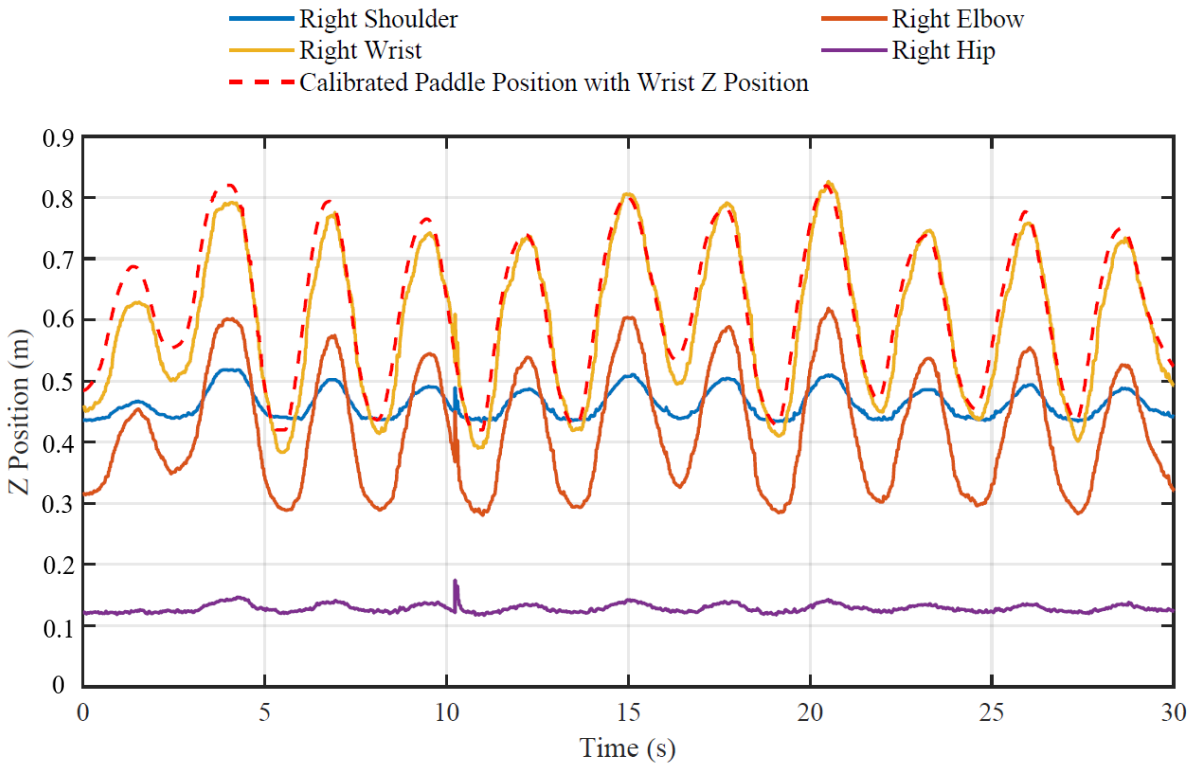

Figure 7.49: Joints Z-time - Assistive Gain-PD-based mode.

7.2.2 Quantitative Analysis

In this section, the performances of all seven participants are analyzed. Since game performance is often subjective, it's difficult to fairly compare users based on a single trial. Individual differences, such as reflexes or familiarity with the game, can make direct comparisons between subjects unreliable. A better, more objective approach is to analyze average performance across all users. However, this kind of comparison requires a larger dataset collected from multiple users, across different control modes, and ideally over repeated sessions, to ensure the analysis reflects general trends rather than isolated performances.

To study and quantify user performance in each control mode more clearly, a few key performance metrics were defined and calculated:

I. Response Time: This is the time between the appearance of a new target and the initiation of paddle movement toward the ball. For each ball, the response time is recorded and then averaged across all movements.

II. Movement Time: This measures the time from the start of the paddle movement until the paddle gets within 10% of the initial distance to the target ball. It reflects how quickly the user approaches the ball once they begin moving.

III. Reach Time: This measures the duration it takes for the paddle to reach a defined proximity to the ball's position, starting from the moment the ball is generated.

IV. Success Rate: This refers to the percentage of total target balls that were successfully caught by the paddle during one game trial.

V. Average Absolute Error: This calculates the average distance between the paddle and the target ball until the moment the ball disappears, either by being missed or caught. This metric provides insight into how close the user was to the target even if the ball was missed, unlike the success rate, which only counts complete catches.

Figure 7.50 shows the average response times of participants in each mode. Based on the results, the value across all assistive modes during movements was smaller than in the other modes. This reduction in response time highlights the assistive mode's supportive role, in which the actuation force helps the user move the paddle more quickly toward the target. In contrast, the resistive modes showed a higher average response time, as the opposing force applied by the robot challenges the user and slows down the initiation of movement. These trends confirm that assistive control helps accelerate paddle movement, while resistive control slows it down.

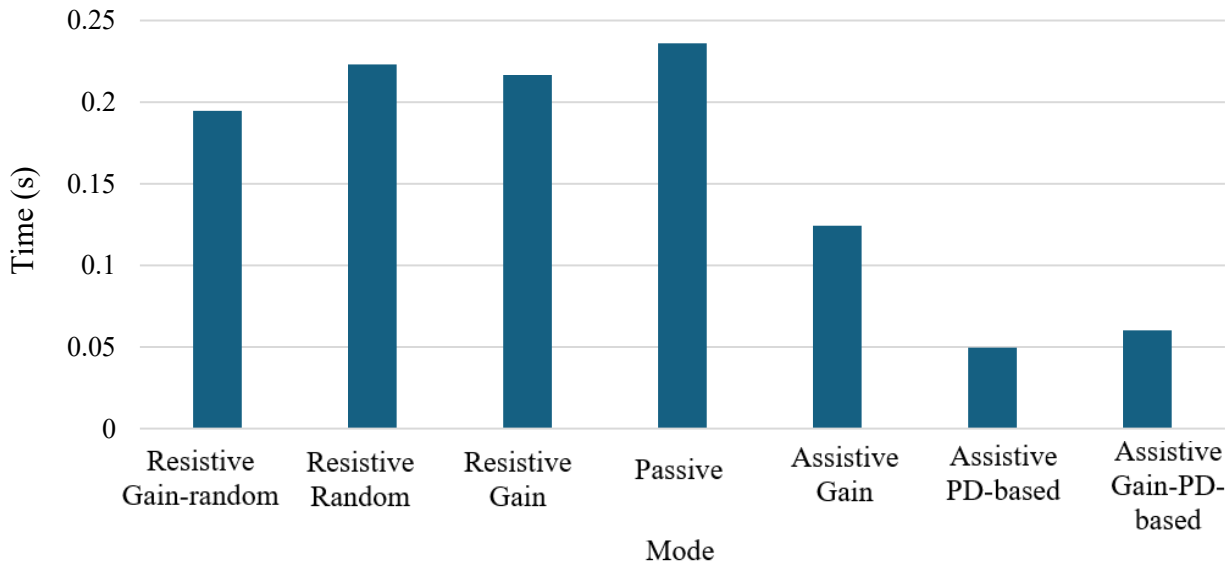

Figure 7.50: Average response time for different modes.

Figure 7.51 presents the average movement time. This metric isolates the motion phase after the response has started. According to the graph, the Resistive and Passive modes yield the highest movement times, all hovering around 0.55 to 0.57 seconds. This indicates that users in these modes experience slower progress toward the target once movement begins, possibly due to resistance or the absence of supportive control. In contrast, the Assistive modes show significantly lower movement times, nearly half as long, demonstrating that assistive control mechanisms, especially those combining gain and PD elements, enable quicker and more efficient motion toward the ball.

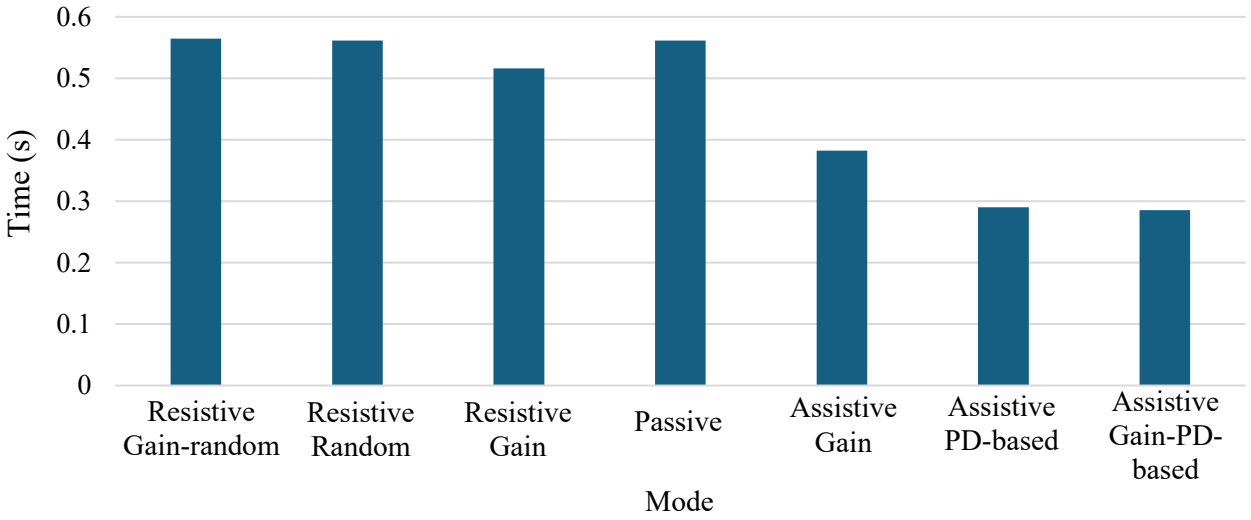

Figure 7.51: Average movement time for different modes.

Figure 7.52 depicts the average reach time. The results show that the Resistive Gain Random mode has the longest reach time, approximately 1.8 seconds, suggesting a delayed response and slower movement under this condition. Also, the Passive and Resistive modes demonstrate relatively high values. On the other hand, all Assistive modes lead to noticeably reduced reach times. The Assistive Gain mode, Assistive PD mode, and Assistive Gain PD mode all maintain reach times between 1.0 and 1.15 seconds. This consistent reduction implies that assistive modes not only help users move faster but also enhance their initial responsiveness to target appearance.

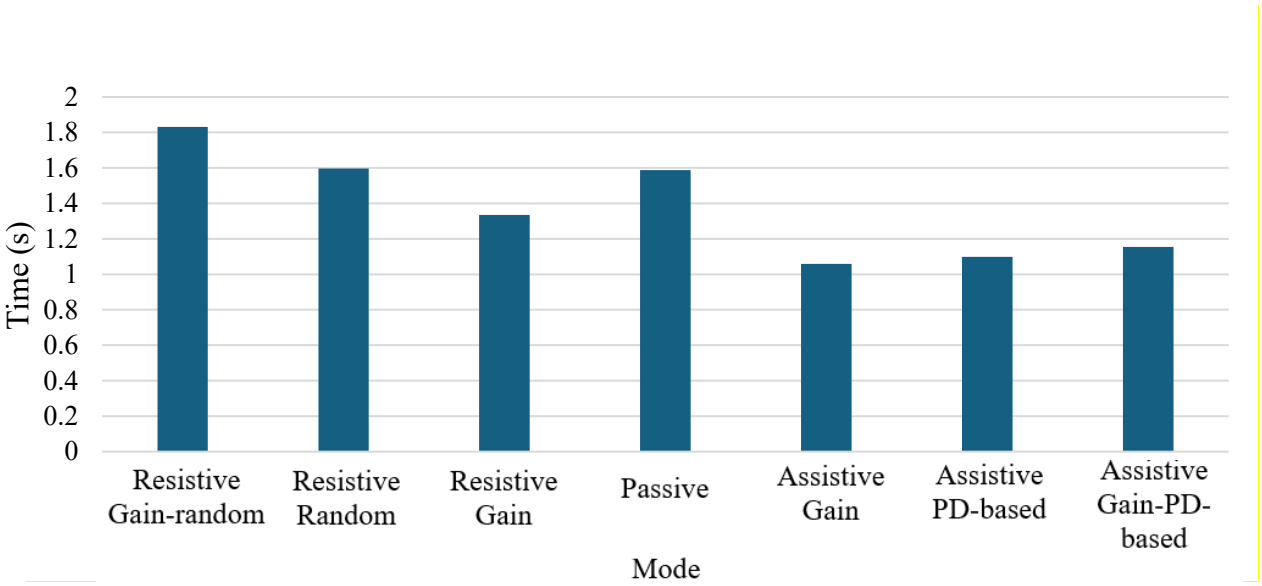

Figure 7.52: Average reach time for different modes.

To better compare the impact of different control algorithms, the average success rate and average absolute error were plotted separately for each assistive and resistive mode in Figure 7.53. In terms of success rate, results show that the Assistive mode resulted in a higher success rate than other modes. On the other hand, the resistive mode led to a significantly lower success rate, confirming that it posed a considerable challenge and effectively increased task difficulty.

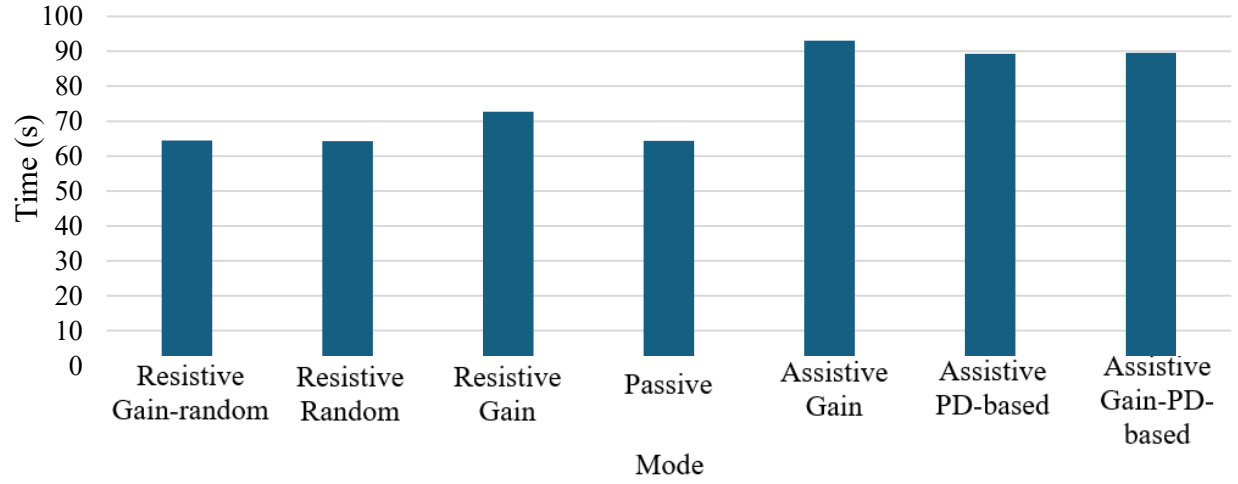

Figure 7.53: Average success rate for different modes.

Figure 7.54 shows the normalized absolute error. A lower normalized error indicates the user stayed closer to the target during the interaction. Among all modes, the Passive Mode results

in the highest error (~0.41), suggesting that without any active Assistance or control adaptation, users tend to deviate more from the target. The Resistive modes show slightly lower errors than the Passive mode, with values around 0.38 to 0.39. Notably, all Assistive modes outperform the others in terms of accuracy, with the Assistive Gain PD Mode showing the lowest error (~0.30). These results highlight the impact of assistive control in not only enhancing speed but also improving the user's precision in following the target path throughout the task.

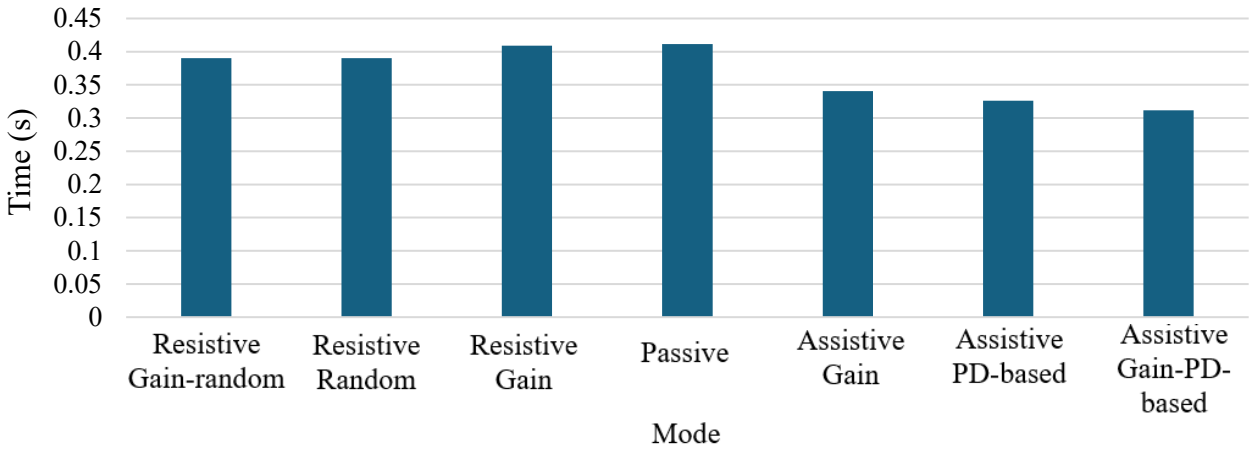

Figure 7.54: Normalized absolute error average for different modes.

7.3 User Inputs

During prototype testing, participants provided several comments on potential improvements or adjustments to enhance the performance, usability, and overall usefulness of the developed system for future implementation in clinical neurological rehabilitative assessments and exercises. Many suggestions were gathered during the evaluation phase, each helping to refine the system. Some of the notable suggestions included:

I. Human model storing: Currently, the system requires building the human kinematic model at the beginning of each session, which adds unnecessary setup time. To address this, the human model should be saved and loaded automatically at the start of each session. This modification would eliminate the need for repeated initialization and provide a more efficient, user-friendly experience.

II. Smoother coefficient transitions in resistive random mode: In the current implementation, the Resistive mode applies sudden, step-like changes to the resistance coefficient, which

can result in abrupt and unnatural force transitions. To enhance the realism and comfort of interaction, the coefficient should be adjusted with smooth, continuous transitions, enabling more natural changes in resistance levels during gameplay.

III. Playable game dimension upgrade: The current game operates in a one-dimensional space, limiting the movement and interaction to vertical control. Transforming the game into a two-dimensional environment would introduce horizontal dynamics, adding complexity, realism, and greater rehabilitation potential, especially for patients with broader mobility goals.

IV. Ball generation algorithm improvement: The existing algorithm for generating ball positions often results in a sinusoidal or highly random pattern, which can reduce the predictability and learning effectiveness for the user. A revised algorithm should be implemented to provide more predictable, structured trajectories, enhancing training benefits while still offering sufficient variability for engagement.

V. Handle redesign for improved grasp: The current robot handle could be redesigned to support a more ergonomic and secure grip, especially for users with limited hand strength or dexterity. A better handle design would reduce fatigue, increase comfort, and promote more natural and effective interaction with the robotic system.

These improvements aim to enhance usability, rehabilitation quality, and system responsiveness, making the platform more robust and adaptable to diverse patient needs.

7.4 Summary

This chapter presented a comprehensive evaluation of the proposed HRI system through a series of controlled experimental trials within the rehabilitation game environment. The system was tested across three modes: Passive, Resistive, and Assistive, each designed to be helpful for participants with different conditions. During the sessions, multiple joints of the human upper body (right hip, right shoulder, right elbow, and right wrist) were tracked in real-time using vision-based body keypoints detection. Key performance metrics, including success rate, response time, movement time, reach time, and average absolute error, were collected and analyzed. These metrics provided objective insights into the user's interaction with the robot, the system's responsiveness, and the effectiveness of each control strategy.

The analysis showed that the system enabled accurate and timely detection of joint movements. Furthermore, a significant achievement was the successful real-time visualization of both the human kinematic model and the 7 degree-of-freedom (DoF) robot within the same environment, visually inspecting their spatial proximity and interactions. This feature provided an intuitive visualization and debugging tool while ensuring safe interaction zones. The results also revealed performance differences across control modes, validating the platform's flexibility and therapeutic potential. Despite minor challenges such as joint occlusion, sensor noise, and some improvements required across different modes, the system proved to be a reliable foundation for rehabilitation.

8 Conclusions

This thesis presented a vision-based Human-robot Interaction (HRI) framework for rehabilitation, implemented using a 7 DoF Kinova robot. The system integrates real-time computer vision to detect and display human body keypoints and movement in real-time, for safety considerations and rehabilitation analysis. The experimental setup also includes an engaging interactive game that serves both as a rehabilitation task and a performance evaluation tool. The system has shown potential to support various modes of interaction between humans and robots, from assistive to resistive, while capturing quantitative metrics such as success rate and response time, and recording 63 human key points for future use.

8.1 Contributions

This thesis contributes to the field of Human-robot Interaction and rehabilitation robotics through several key innovations:

I. Real-Time Vision-based HRI System: A complete framework was developed that integrates real-time pose detection with a robot, allowing smooth HRI. A real-time visualization environment was created to show both the human skeleton and the robot in proximity, paving the way for analysis of interaction dynamics in a shared 3D space.

II. Design and Implementation of Control Algorithms: Multiple control modes were implemented, including assistive control using Impedance, DTC and PD control algorithms.

III. Multi-Mode Rehabilitation Framework: The system supports three modes of interaction Passive, Assistive, and Resistive, allowing flexible adaptation to different rehabilitation needs and difficulty levels.

IV. Quantitative Performance Metrics: Objective evaluation criteria such as Success-rate, Movement-time, Response-time, Reach-time, and average absolute error were developed to assess user performance and track rehabilitation progress.

8.2 Key Findings

I. The proposed system can effectively detect and track human joint movements in real-time using vision-based methods, despite some noise in depth estimation.

II. HRI can be gamified for rehabilitation purposes, allowing engaging exercises with quantifiable performance metrics.

III. The robot can successfully assist or challenge the human user depending on the selected mode, providing a flexible rehabilitation platform adaptable to different therapy needs.

IV. Metrics such as Response-time and Movement-time offer valuable insight into patient behavior and motor control performance.

8.3 Limitations of the Designed System

I. Joint Occlusion: The system may fail to detect or accurately track joints when they are occluded by other body parts (e.g., one arm covering the other) or by external objects, which affects the continuity and reliability of the detected skeleton.

II. Limited Interaction Complexity: The current system focuses on relatively simple motion tasks involving single-joint or end-effector movements. More complex full-body motions or multi-joint coordination are not yet supported.

III. Lack of Personalized Feedback: The system does not yet provide real-time feedback to the user about incorrect or suboptimal body posture or configuration. For example, if the user's body is not in the correct position or alignment, the system does not inform them or prompt corrections.

IV. Static Control Strategies: The robot's behavior is based on pre-defined logic rather than learning or adapting dynamically to the user’s responses, fatigue levels, or performance variations.

V. No Longitudinal Adaptation: There is no embedded logic to assess long-term performance trends or personalize the therapy based on improvement.

8.4 Future Works

Future work could extend the current system in several impactful directions:

I. Advanced Rehabilitation Scenarios: Supporting complex motions involving multiple joints, such as shoulder-elbow coordination, or full-body tasks like reaching and balance training

II. Real-Time Feature Extraction and Adaptation: Extracting biomechanical features (e.g., speed, range of motion, fatigue patterns) from the human skeleton in real-time to enable intelligent adaptation of robot behavior or therapy difficulty.

III. Dynamic Safety Zones: Defining and updating collision-avoidance safety boundaries around humans in real-time to ensure safe operation of the robot in shared workspaces.

IV. Adaptive Control Based on Performance: Implementing learning algorithms or reinforcement learning to adjust control strategies based on patient progression or session history.

V. Integration with Wearables: Combining vision data with wearable sensors (e.g., IMUs or EMG) to increase system robustness and improve motion intent inference.

VI. Cloud-Based Data Logging and Analysis: Enabling remote monitoring, data aggregation, and machine learning-driven insights for clinicians to personalize and evaluate rehabilitation strategies over time.

References

[1] L. Benos, V. Moysiadis, D. Kateris, A. Tagarakis, P. Busato, S. Pearson and D. Bochtis, "Human–Robot Interaction in Agriculture: A Systematic Review," *Sensors,* vol. 23, no. 15, p. 6776, 2023.

[2] C. Wong, E. Yang, X.-T. Yan and D. Gu, "Autonomous robots for Harsh Environments: A holistic overview of current solutions and ongoing challenges," *Systems Science & Control Engineering,* vol. 6, no. 1, pp. 213-219, 2018.

[3] I. Maurtua, A. Ibarguren, J. Kildal, L. Susperregi and B. Sierra, "Human–robot collaboration in Industrial Applications," *International Journal of Advancmsed Robotic Syste,* vol. 14, no. 4, p. 172988141771601, 2017.

[4] H. Su, W. Qi, J. Chen, C. Yang, J. Sandoval and M. Laribi, "Recent Advancements in Multimodal Human–Robot Interaction," *Frontiers in Neurorobotics,* vol. 17, 2023.

[5] A. Mohebbi, "Human-Robot Interaction in Rehabilitation and Assistance: A Review," *Current Robotics Reports,* vol. 1, no. 3, pp. 131-144, 2020.

[6] E. B. Küçüktabak, S. J. Kim, Y. Wen, K. Lynch and J. L. Pons, "Human-Machine-Human Interaction in Motor Control and Rehabilitation: A Review," *Journal of NeuroEngineering and Rehabilitation,* vol. 18, no. 1, 2021.

[7] H. Krebs, J. Palazzolo, L. Dipietro, M. Ferraro, J. Krol, K. Rannekleiv, B. Volpe and N. Hogan, "Rehabilitation robotics: performance-based progressive robot-assisted therapy," *Auton Robot,* vol. 15, no. 1, p. 7–20, 2003.

[8] N. Stepanenko and A. Dubko, "Motion tracking systems in rehabilitation: a comparative analysis of sensor technologies, algorithmic methods, and clinical relevance," *International Science Journal of Engineering & Agriculture,* vol. 4, pp. 72-100, 2025.

[9] J. He, M. K. Farlie and P. Carreno-Medrano, "Use of Socially Assistive Robots in Physiotherapy: Scoping Review," *JMIR Rehabil Assist Technol,* vol. 12, p. e69908, 2025.

[10] J. P. Vasconez, G. A. Kantor and F. A. Auat Cheein, "Human-Robot interaction in agriculture: A survey and current challenges," *Biosystems Engineering,* vol. 179, p. 35–48, 2019.

[11] A. Sharkawy and P. N. Koustoumpardis, "Human–Robot Interaction: A review and analysis on variable admittance control, safety, and perspectives," *Machines,* vol. 10, no. 7, p. 591, 2022.

[12] H. Kress, K. Eder, G. Hoffman, H. Admoni, B. Argall, R. Ehlers, C. Heckman, N. Jansen, R. Knepper, J. Křetínský, S. Levy-Tzedek, J. Li, T. Murphey, L. Riek and D. Sadigh, "Formalizing and guaranteeing human-robot interaction," *Communications of the ACM,* vol. 64, no. 9, pp. 78-84, 2021.

[13] J. Schmidtler, V. Knott, C. Hölzel and K. Bengler, "Human centered assistance applications for the working environment of the future," *Occupational Ergonomics,* vol. 12, no. 3, p. 83–95, 2015.

[14] J. M. Beer, A. D. Fisk and W. A. Rogers, "Toward a framework for levels of robot autonomy in human-robot interaction," *Journal of Human-Robot Interaction,* vol. 3, no. 2, p. 74–99, 2014.

[15] L. Onnasch and E. Roesler, "A taxonomy to structure and analyze human–robot interaction," *International Journal of Social Robotics,* vol. 13, no. 4, p. 833–849, 2020.

[16] T. Kruse, A. K. Pandey, R. Alami and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems,* vol. 61, no. 12, p. 1726–1743, 2013.

[17] B. D. Argall and A. G. Billard, "A survey of tactile human–robot interactions," *Robotics and Autonomous Systems,* vol. 58, no. 10, p. 1159–1176, 2010.

[18] A. Keshvarparast, D. Battini, O. Battaia and A. Pirayesh, "Collaborative Robots In Manufacturing and Assembly Systems: Literature Review and Future Research Agenda," *Journal of Intelligent Manufacturing,* vol. 35, no. 5, pp. 2065-2118, 2023.

[19] V. Seidita, F. Lanza, A. Pipitone and A. Chella, "Robots as intelligent assistants to face COVID-19 pandemic," *Briefings in Bioinformatics,* vol. 22, no. 2, pp. 823-831, 2020.

[20] H. Yan, M. H. Ang and A. N. Poo, "A survey on perception methods for human–robot interaction in Social Robots," *International Journal of Social Robotics,* vol. 6, no. 1, p. 85–119, 2013.

[21] M. Schwenk and K. O. Arras, "R2-D2 reloaded: A flexible sound synthesis system for sonic human-robot interaction design," *The 23rd IEEE International Symposium on Robot and Human Interactive Communication,* p. 161–167, 2014.

[22] C. Marechal, D. Mikołajewski, K. Tyburek, P. Prokopowicz, L. Bougueroua, C. Ancourt and K. Węgrzyn, "Survey on AI-based multimodal methods for emotion detection," *Lecture Notes in Computer Science,* p. 307–24, 2019.

[23] G. Cheng, E. Dean, F. Bergner, J. Rogelio , Q. Leboutet and P. Mittendorfer, "A comprehensive realization of robot skin: Sensors, sensing, control, and applications," *Proceedings of the IEEE,* vol. Proceedings of the IEEE, no. 10, p. 2034–2051, 2019.

[24] A. Bonarini, "Communication in human-robot interaction," *Current Robotics Reports,* vol. 1, no. 4, p. 279–285, 2020.

[25] T. B. Sheridan, "Human–Robot Interaction," *Human Factors: The Journal of the Human Factors and Ergonomics Society,* no. 4, pp. 525-532, 2016.

[26] L. D. Riek, "Healthcare Robotics," *Communications of the ACM,* vol. 60, no. 11, p. 68–78, 2017.

[27] M. J. Matarić and B. Scassellati, "Socially assistive robotics," *Springer Handbooks,* pp. 1973-1994, 2016.

[28] L. Zhang, S. Guo and Q. Sun, "Development and assist-as-needed control of an end-effector Upper Limb Rehabilitation Robot," *Applied Sciences,* vol. 10, no. 19, pp. 66-84, 2020.

[29] A. Rodríguez-Fernández, J. Lobo-Prat and J. M. Font-Llagunes, "Systematic review on wearable lower-limb exoskeletons for gait training in neuromuscular impairments," *Journal of NeuroEngineering and Rehabilitation,* vol. 18, no. 1, p. 22, 2021.

[30] G. Scivoletto, F. Tamburella, L. Laurenza, M. Torre and M. Molinari, "Who is going to walk? A review of the factors influencing walking recovery after Spinal Cord Injury," *Frontiers in Human Neuroscience,* vol. 8, p. 141, 2014.

[31] J. Rivera , J. Bory Reyes, L. M. Hernández Simón and J. I. Palacios , "Rehabilitation exoskeletons: A systematic literature review," *Revista Mexicana de Ingeniería Biomédica,* vol. 45, no. 2, p. 78–99, 2024.

[32] J. Guo, P. Li and S. Guo, "Design and analysis of a wearable exoskeleton upper limb rehabilitation robot," *2020 IEEE International Conference on Mechatronics and Automation (ICMA),* pp. 1564-1569, 2020.

[33] Y.-L. Park, B.-r. Chen, N. O. Pérez-Arancibia, D. Young, L. Stirling, R. J. Wood, E. C. Goldfield and R. Nagpal, "Design and control of a bio-inspired soft wearable robotic device for ankle–foot rehabilitation," *Bioinspiration & Biomimetics,* vol. 9, no. 1, p. 016007, 2014.

[34] Y.-L. Park, J. Santos, K. G. Galloway, E. C. Goldfield and R. J. Wood, "A soft wearable robotic device for active knee motions using flat pneumatic artificial

muscles," *2014 IEEE International Conference on Robotics and Automation (ICRA),* pp. 4805-4810, 2014.

[35] I. Galiana, F. L. Hammond, R. D. Howe and M. B. Popovic, "Wearable soft robotic device for post-stroke shoulder rehabilitation: Identifying misalignments," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 317-322, 2012.

[36] V. Oguntosin, W. S. Harwin, S. Kawamura, S. J. Nasuto and Y. Hayashi, "Development of a wearable assistive soft robotic device for elbow rehabilitation," *2015 IEEE International Conference on Rehabilitation Robotics (ICORR),* pp. 747-752, 2015.

[37] N. W. Bartlett, V. Lyau, W. A. Raiford, D. Holland, J. B. Gafford, T. D. Ellis and C. J. Walsh, "A soft robotic orthosis for wrist rehabilitation1," *Journal of Medical Devices,* vol. 9, no. 3, p. 030918, 2015.

[38] C.-Y. Chu and R. M. Patterson, "Soft robotic devices for hand rehabilitation and assistance: A narrative review," *Journal of NeuroEngineering and Rehabilitation,* vol. 15, no. 1, 2018.

[39] T. Ridremont, I. Singh, B. Bruzek, V. Erel, A. Jamieson, Y. Gu, R. Merzouki and M. Wijesundara, "Soft robotic bilateral rehabilitation system for hand and wrist joints," *Machines,* vol. 12, no. 5, p. 288, 2024.

[40] J. Laut, M. Porfiri and P. Raghavan, "The present and future of robotic technology in rehabilitation," *Current Physical Medicine and Rehabilitation Reports,* vol. 4, no. 4, pp. 312-319, 2016.

[41] K. Hu, Z. Ma, S. Zou, J. Li and H. Ding, "Impedance sliding-mode control based on stiffness scheduling for rehabilitation robot systems," *Cyborg and Bionic Systems,* vol. 5, 2024.

[42] P. I. Corke, W. Jachimczyk and R. Pillat, Robotics, vision and Control: Fundamental Algorithms in MATLAB®, vol. 73, Cham: Springer International Publishing : Imprint: Springer, 2023.

[43] N. Robinson, B. Tidd, D. Campbell, D. Kulić and P. Corke, "Robotic Vision for human-robot interaction and collaboration: A survey and systematic review," *ACM Transactions on Human-Robot Interaction,* vol. 12, no. 1, pp. 1-66, 2023.

[44] Z. Chuanrui, L. Yonggen, L. Minglei, Q. Minghan and . W. Haoqian, "Category-level Object Detection, Pose Estimation and Reconstruction from Stereo Images," *European Conference on Computer Vision. Springer,,* vol. 2, 2024 .

[45] D. Forsyth and J. Ponce, Computer vision: A modern approach, Pearson Education UK, 2015.

[46] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision,* vol. 57, no. 2, pp. 137-154, 2004.

[47] C. Zhe, S. Tomas, W. Shih-En and S. Yaser, "Real-time Multi-Person 2D Pose Estimation using Part Affinity Fields," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2017.

[48] J. F. Henriques, R. Caseiro, P. Martins and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 37, no. 3, pp. 583-596, 2015.

[49] M. Spitale, M. T. Parreira, M. Stiber, M. Axelsson, N. Kara, G. Kankariya, C. Huang, M. Jung, W. Ju and H. Gunes, "ERR@HRI 2024 challenge: Multimodal detection of errors and failures in human-robot interactions," *International Conference on Multimodel Interaction,* pp. 652-656, 2024.

[50] A. A. Toaiari, "Exploring 3D Human Pose Estimation and Forecasting from the Robot's Perspective: The HARPER Dataset," *preprint,* 2024.

[51] J.-Y. Choi, E. Ha, M. Son, J.-H. Jeon and J.-W. Kim, "Human joint angle estimation using Deep Learning-based three-dimensional human pose estimation for application in a real environment," *Sensors,* vol. 24, no. 12, p. 3823, 2024.

[52] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll and M. J. Black, "SMPL: a skinned multi-person linear model," *ACM Transactions on Graphics,* vol. 34, no. 6, pp. 1-16, 2015.

[53] KINOVA, "www.kinovarobotics.com," [Online]. Available: https://www.kinovarobotics.com/uploads/User-Guide-Gen3-R07.pdf.

[54] T. Mallick, P. P. Das and A. K. Majumdar, "Characterizations of Noise in Kinect Depth Images: A Review," *IEEE Sensors Journal,* vol. 14, pp. 1731-1740, 2014.

[55] K. Khoshelham and S. Oude Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications," *Sensors,* vol. 12, pp. 1437--1454, 2012.

[56] J. J. Craig, Introduction to robotics: Mechanics and control, Harlow, United Kingdom: Pearson Education Limited, 2022.

[57] J. M. Hollerbach and G. Sahar, "Wrist-partitioned, inverse kinematic accelerations and manipulator dynamics," *The International Journal of Robotics Research,* vol. 2, no. 4, pp. 61-76, 1983.

Appendix A: Kinematic Model of the Manipulator

To derive the forward kinematics of the manipulator, the geometric structure of the manipulator is defined in terms of its joint variables and link parameters. Figure 0.1 illustrates the simplified kinematic chain used in this work. Each revolute joint q_1, q_2, q_3 provides a rotational degree of freedom, while the link parameters a_0, a_1, a_2, a_3 represent the distances between consecutive joints.

This kinematic representation enables a systematic mapping from joint space to the end-effector pose and forms the basis for the forward kinematic formulation presented in the main text.

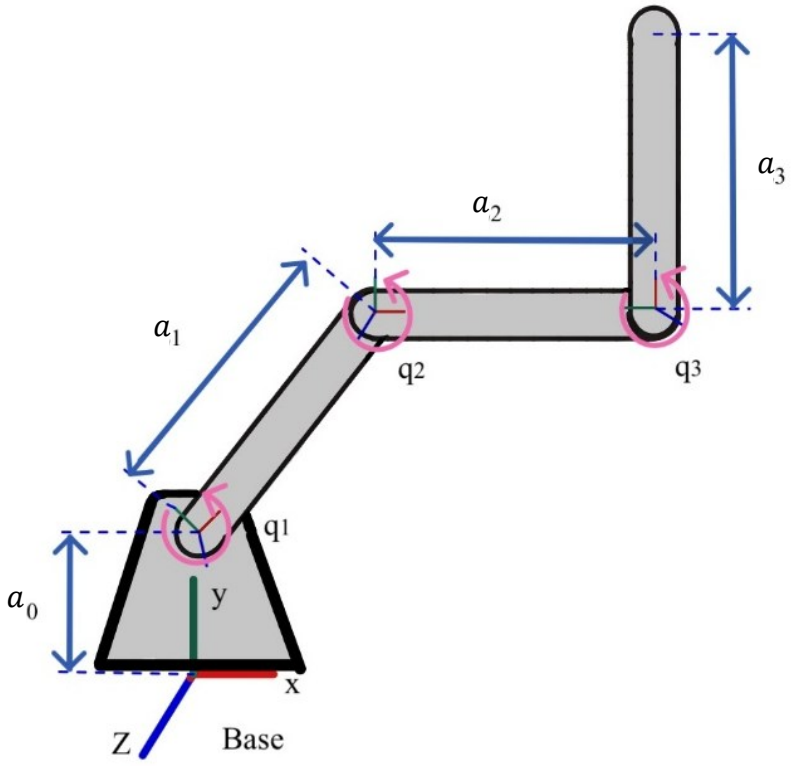

Figure A.1: Kinematic diagram of a 3-DoF planar robotic manipulator showing joint angles and link lengths. The base coordinate frame (x, y, z) is fixed to the robot base, and each revolute joint defines a rotational degree of freedom.

For a robotic manipulator with n joints, the forward kinematics can be expressed as a sequence of homogeneous transformations:

$$T_n^0 = T_1^0 . T_2^1 . T_3^2 \dots T_n^{n-1} \tag{36}$$

Where T_{i+1}^i denotes the homogeneous transformation matrix from frame $i$to frame $i+1$.

Using the Denavit–Hartenberg (DH) convention, the homogeneous transformation matrix for the planar manipulator considered in this work is given by:

$$T_{i+1}^{i} = \begin{bmatrix} \cos q_i & -\sin q_i & 0 & a_i \\ \sin q_i & \cos q_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{37}$$

For the planar manipulator considered in this work, the parameters simplify due to zero link twist and offset angles.

where:

q_i = joint angle for revolute joints,

L_i = i_{th} link length.

d_i = i_{th} link offset (d_i = 0).