TECHNICAL UNIVERSITY OF CRETE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Control of Humanoid Robot through Eye Tracking Interface for People with Moving Disabilities

Athanasios-Iakovos Apostolopoulos

Thesis Committee
Professor Michail G. Lagoudakis (ECE)
Professor Katerina Mania (ECE)
Professor Panagiotis Partsinevelos (MRE)

Chania, March 2024

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Έλεγχος Ανθρωποειδούς Ρομπότ μέσω Διεπαφής Παρακολούθησης Βλέμματος για Άτομα με Κινητική Δυσκολία

Αθανάσιος-Ιάκωβος Αποστολόπουλος

Εξεταστική Επιτροπή
Καθηγητής Μιχαήλ Γ. Λαγουδάκης (ΗΜΜΥ)
Καθηγήτρια Κατερίνα Μανιά (ΗΜΜΥ)
Καθηγητής Παναγιώτης Παρτσινέβελος (ΜΗΧΟΠ)

Χανιά, Μάρτιος 2024

Abstract

In the field of assistive technologies for the mobility impaired people, this diploma thesis embarks on a journey motivated by the profound aspiration to enhance the quality of life for individuals facing total mobility impairment. Confronted with day-to-day challenges that often necessitate reliance on others for assistance, the quest for alternative methods to provide accessibility becomes imperative. This journey involves a meticulous exploration of diverse technologies, tools, and assets to serve the specific needs of individuals with tetraplegia within the broader domain of assistive technologies. More specifically, our research addresses the overarching problem by combining robotic systems and eye-tracking technology, seeking to empower individuals with limited autonomy through innovative and intuitive solutions and to bridge the social gap by providing some alternative autonomy. Our approach centers on integrating eye-tracking technology into a Unity-created application and establishing a communication protocol through the Robot Operating System (ROS) framework to control a robotic system in simulation. The result is an intuitive, lean, and user-friendly application to control a humanoid robot. While several challenges emerged in the course of the thesis, such as compatibility issues and software limitations, the end system lays the groundwork for the development of future applications in the field of assistive technologies. A multifaceted user evaluation of the system's performance is also discussed. Overall, our approach encapsulates a pioneering effort in assistive technologies, illuminating a path toward a more inclusive and accessible future.

Περίληψη

Στον τομέα των υποστηρικτικών τεχνολογιών για άτομα με κινητικά προβλήματα, η παρούσα διπλωματική εργασία ξεκινά ένα ταξίδι με κίνητρο τη βαθιά επιδίωξη να βελτιώσει την ποιότητα ζωής ατόμων που αντιμετωπίζουν ολική κινητική αναπηρία. Αντιμέτωποι με καθημερινές προκλήσεις που συχνά απαιτούν την εξάρτηση από άλλους για βοήθεια, η αναζήτηση εναλλακτικών μεθόδων για την παροχή προσβασιμότητας καθίσταται επιτακτική. Αυτό το ταξίδι περιλαμβάνει μια σχολαστική εξερεύνηση διαφορετικών τεχνολογιών, εργαλείων και πόρων για την εξυπηρέτηση των ειδικών αναγκών των ατόμων με τετραπληγία στον ευρύτερο τομέα των υποστηρικτικών τεχνολογιών. Πιο συγκεκριμένα, η έρευνά μας αντιμετωπίζει το γενικότερο πρόβλημα συνδυάζοντας ρομποτικά συστήματα και τεχνολογία παρακολούθησης βλέμματος, επιδιώκοντας να ενδυναμώσει άτομα με περιορισμένη αυτονομία μέσω καινοτόμων και διαισθητικών λύσεων και να γεφυρώσει το κοινωνικό χάσμα παρέχοντας κάποια εναλλακτική αυτονομία. Η προσέγγισή μας επικεντρώνεται στην ενσωμάτωση της τεχνολογίας παρακολούθησης βλέμματος σε μια εφαρμογή που δημιουργήθηκε στο περιβάλλον Unity και στην ανάπτυξη ενός πρωτοκόλλου επικοινωνίας μέσω του πλαισίου Robot Operating System (ROS) για τον έλεγχο ενός ρομποτικού συστήματος σε προσομοίωση. Το αποτέλεσμα είναι μια διαισθητική, λιτή και φιλική προς το χρήστη εφαρμογή για τον έλεγχο ενός ανθρωποειδούς ρομπότ. Ενώ προέκυψαν αρκετές προκλήσεις κατά τη διάρκεια της εργασίας, όπως ζητήματα συμβατότητας και περιορισμοί λογισμικού, το τελικό σύστημα θέτει τις βάσεις για την ανάπτυξη μελλοντικών εφαρμογών στον τομέα των υποστηρικτικών τεχνολογιών. Συζητείται επίσης μια πολύπλευρη αξιολόγηση των επιδόσεων του συστήματος από τους χρήστες. Συνολικά, η προσέγγισή μας περικλείει μια πρωτοποριακή προσπάθεια στις υποστηρικτικές τεχνολογίες, που αναδεικνύει μια πορεία προς ένα μέλλον πιο προσβάσιμο, χωρίς αποκλεισμούς.

Acknowledgments

First of all, I would like to thank the thesis committee, Professor Michail G. Lagudakis, Professor Katerina Mania, and Professor Panagiotis Partsinevelos for their guidance and support.

I would like to thank the SenceLab research team for their support and providing a great and inspiring environment for innovation and research. I especially thank Angelos Antonopoulos, a good friend and colleague, for his guidance, support, and inspiration all those years.

I would like to thank the Surreal research team for their advice. Specifically to Minas Katsiokalis for his guidance.

Last, but not least, I am grateful to my family, my cornerstone, for their love, faith, and sacrifices.

Athanasios-Iakovos Apostolopoulos, March 2024

Table of Contents

List of Figures

List of Tables

List of Abbreviations

POV	Point of View
ROS	Robotic Operating System
GUI	Graphical User Interface
SDK	Software Development Kits
API	Application Programming Interfaces
PR2	Personal Robot 2
URL	Uniform Resource Locator
HTTP	HyperText Transfer Protocol
PCCR	Pupil Central Cornwall Reflection

Chapter 1 Introduction

Individuals with mobility impairments navigate daily existence fraught with challenges far beyond physical limitations. The simple tasks that many take for granted, such as moving from one place to another or accessing basic amenities, become arduous undertakings fraught with obstacles. Dependence on external support is frequently an undeniable truth, gradually diminishing individual autonomy and self-reliance. However, the availability of personnel to provide necessary aid is only sometimes guaranteed, leaving individuals grappling with a profound sense of dependency and restricted freedom. In this context, the quest for alternative methods that can empower individuals with mobility impairments to reclaim control over their lives takes on paramount importance. Within this domain of assistive technologies, the pursuit of innovative solutions unfolds, driven by the imperative need to alleviate daily struggles and enhance the quality of life for those with mobility challenges.

The field of robotics has witnessed remarkable advancements, particularly in accessibility support for individuals with mobility impairment. These innovations aim to enhance the quality of life for people facing partial or complete mobility challenges, offering them newfound independence and opportunities for interaction with their surroundings. One significant avenue within this domain involves the integration of humanoid robots, bridging the gap between individuals with limited mobility and the environment.

The motivation behind this thesis stems from the deep desire to empower individuals facing mobility challenges. In the pursuit of providing telepresence capabilities and self-assistance, integrating robotics becomes a promising avenue. Enabling users to interact with their surroundings, navigate spaces, and perform tasks remotely has the potential to redefine accessibility standards. By leveraging robotic technologies, particularly humanoid robots, we strive to bring a transformative change that enhances the overall well-being of individuals with partial or complete mobility impairment.

Despite the advancements in robotics for accessibility support, one persistent challenge remains—the limited means of user input for controlling humanoid robots. Conventional interfaces may not adequately address the diverse needs and capabilities of individuals with mobility impairment. This creates a barrier to the seamless interaction between users and robots, restricting the full potential of these technologies.

In this thesis, we propose a novel approach to address the aforementioned challenge. We harness the capabilities of an eye tracker to serve as an intuitive control interface. Through the integration of an eye-controlled user interface developed in Unity, users can effortlessly send commands to a PR2 (Personal Robot 2) robot. The robot, operating within the Robot Operating System (ROS) framework and simulated in the Gazebo environment, responds to these commands, offering a tailored solution to the limited user input problem. This

innovative approach opens avenues for telepresence and self-assistance, redefining the accessibility landscape in robotics.

1.1 Thesis Contribution

The primary contribution of this thesis is developing a cohesive system that leverages eye-tracking capabilities to control a cursor within a Unity-based GUI, subsequently interfacing with the PR2 robot through the ROS Noetic system. The approach involves meticulous design considerations, addressing challenges in message type compatibility between Unity and ROS, and refining the eye-tracking implementation for enhanced user experiences. The results demonstrate a functional system, showcasing the successful integration of these diverse technologies to enable more natural and efficient human-robot interactions. This research contributes to the advancement of assistive technologies and the broader field of robotics, where intuitive control mechanisms play a pivotal role in shaping the future of human-robot collaboration.

1.2 Thesis Overview

- **Chapter 2 - Theoretical Background:** In this chapter, the presentation of all the essential background knowledge needed for this thesis is included. An introduction to all technologies, tools, and assets used in the implementation process. Finally, a brief description of Tetraplegia and the domain of Assistive Technologies is presented. It gives an insight into the challenges addressed in this thesis.
- **Chapter 3 - Problem Statement:** This chapter identifies and articulates the core issues and challenges that form the basis of the research in this thesis. A detailed analysis of the problems faced by individuals with mobility impairments and the limitations of existing assistive technologies is presented. This chapter sets the stage for the subsequent chapters, framing the context for the proposed solutions.
- **Chapter 4 - Approach:** This chapter provides a comprehensive insight into the methodologies, strategies, and approaches adopted to tackle the identified problems. It outlines the systematic process employed to design and implement the solution, offering a transparent view of the thought process and decision-making that led to the development of the system.
- **Chapter 5 - Results:** This chapter presents the results of a user study conducted to evaluate the eye-tracking-controlled application. Two rounds of testing were performed with four participants, focusing on intuitiveness, usability, aesthetics, response time, and identifying any issues. The results indicated that the system achieved its goal of providing an intuitive, streamlined, and user-friendly interface, especially benefiting scenarios where a standard mouse is impractical.
- **Chapter 6 - Conclusion:** The final chapter consolidates the entire thesis, providing a cohesive summary of the journey from problem identification to solution implementation. It includes a discussion of the results, their implications, and their contribution to the broader field of assistive technologies. The concluding chapter

also paves the way for future work, reflecting on lessons learned and offering insights for continued advancements in the domain.

Chapter 2 Background

2.1 Eye Tracking Technologies

Eye-tracking technologies [1] have revolutionized human-computer interaction by capturing and analyzing individuals' gaze behavior. Employing various techniques, these technologies monitor the movement and focus of a user's eyes, providing valuable insights into visual attention and cognitive processes.

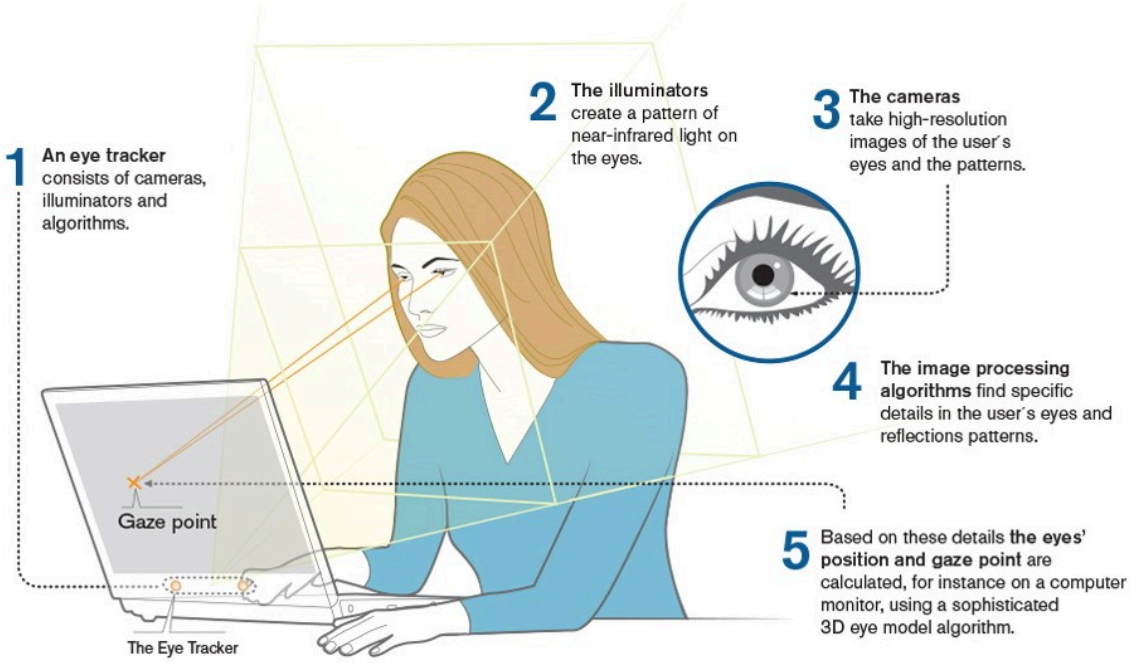

Figure 2.1: Tobii screen-based eye trackers

The most common method is called pupil central corneal reflection (PCCR). As shown in Figure 2.1, it involves using infrared light to illuminate the eyes and capture reflections of the cornea and retina. This data is then processed to determine the point of gaze, enabling precise tracking of eye movements. Eye tracking finds applications across various domains, from usability studies in design to assistive technologies for individuals with disabilities. In the context of this thesis, eye tracking serves as a pivotal interface, allowing users to control a user interface and, consequently, a humanoid robot, adding a different solution to accessible and intuitive interaction for individuals with mobility impairments.

2.2 Unity

Unity [2] is a cross-platform game engine developed by Unity Technologies. Originally designed as a game engine, Unity offers robust capabilities for rendering graphics,

handling physics, managing animations, and creating immersive 3D and 2D environments. Unity provides a broad set of tools and features that empower developers to build applications for diverse platforms. Unity's Asset Store provides a vast marketplace where developers can find and share assets, plugins, and tools to enhance their projects. Development accelerates by offering pre-built components and functionalities. Unity's versatility and feature-rich environment make it a preferred choice for developers creating various applications, including UI-centric ones.

2.3 Robot Operating System

Robot Operating System (ROS) [3] is an open-source middleware framework designed for the development, integration, and operation of robotic systems. Despite its name, ROS t name, ROS is not an operating system but a collection of tools, libraries, and templates that facilitate the development of robotic systems. It was initially developed by the Stanford Artificial Intelligence Laboratory in collaboration with Willow Garage and later maintained by the Open Source Robotic Foundation (OSRF). ROS has evolved into a standard in the robotics community, fostering collaboration and innovation across a plethora of robotic applications.

Many useful and key features define this tool. ROS is built in a middleware architecture, enabling seamless communication among distributed robotic communications. This communication middleware facilitates the exchange of data between modules, allowing for the construction of complex robotic systems composed of a variety of sensors, actuators, and computational units.

ROS adopts a distributed computing approach, breaking down robotic systems into modular units called nodes. Nodes are lightweight processes that communicate asynchronously with each other using a publish-subscribe messaging system. This way the nodes exchange data seamlessly and operate independently of each other.

In ROS, topics are a fundamental communication mechanism that enables different parts of a robotic system to exchange data. ROS topics facilitate the publish-subscribe communication. This decoupled communication architecture enhances modularity, flexibility, and scalability in robotic systems.

Through its service framework, remote procedure calls (RPC) are also supported in the form of requests and replies. A service provider node registers a service while service clients make requests and wait for replies. Standardized message types enable nodes to communicate uniformly.

Functionality in ROS is Organized into packages, which are directories containing configuration files, libraries, and executables. Packages encapsulate specific functionalities, making code reuse and collaboration more manageable.

The ROS Master registers all of the nodes in the ROS system. It tracks publishers and subscribers to topics as and as services. It enables individual ROS nodes to locate one another. Once these nodes have located each other, they communicate with each other peer-to-peer.

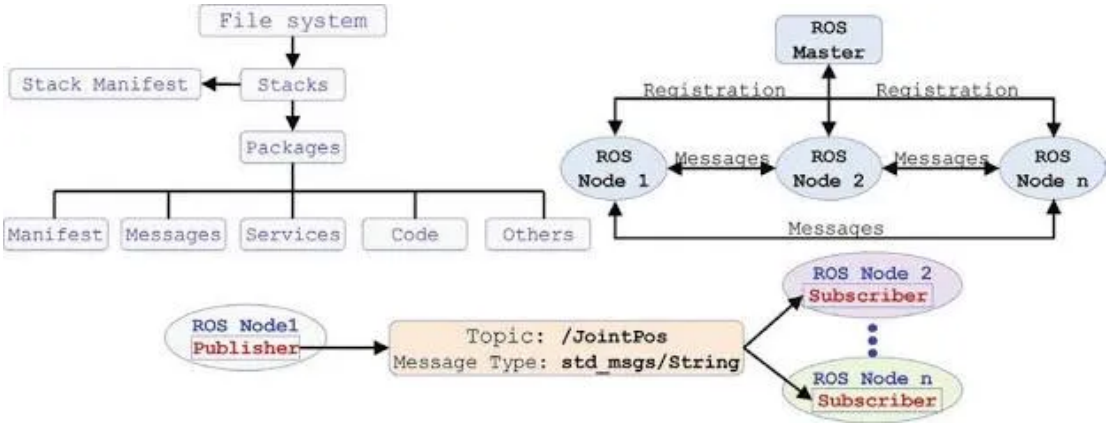

Figure 2.2: ROS File System, The Computation Graph Level, The Community Level

2.4 Gazebo

Gazebo [4] is an open-source 3D simulation environment widely used in robotics for simulating the dynamics and interactions of robots in diverse environments. It serves as a powerful tool for testing and validating robotic algorithms, controllers, and systems before deploying them on physical hardware. Gazebo is part of the ROS ecosystem, seamlessly integrating with other ROS tools to create a comprehensive robotic development platform.

Gazebo facilitates the development and testing of robot algorithms, including navigation, perception, and control. Researchers and developers can iterate quickly in the simulated environment, refining algorithms before deploying them on physical robots.

2.5 Tetraplegia

Tetraplegia (preferred to “quadriplegia”) [5] [6], is a paralysis condition that results in the loss of sensory and motor function in all four limbs and typically the torso. It is caused by injury or damage to the spinal cord, usually in the cervical region (neck), affecting the nerves that control movement and sensation throughout the body. Individuals with tetraplegia experience varying degrees of impairment, ranging from partial to complete paralysis, depending on the level and extent of spinal cord injury. This condition often leads to challenges in mobility, self-care, and overall physical independence. Rehabilitation, assistive technologies, and support services play crucial roles in helping individuals with tetraplegia adapt to and navigate their daily lives.

2.6 Assistive Technologies

Assistive technology [7] refers to devices, tools, software, or equipment designed to enhance the functional capabilities and independence of individuals with disabilities or limitations. These technologies aim to mitigate barriers and facilitate access to information, communication, education, employment, and daily activities. Assistive technologies can cover a broad spectrum, including mobility aids, communication devices, screen readers, adaptive computer peripherals, and other specialized tools tailored to address specific challenges associated with various disabilities. The primary goal is to empower individuals with disabilities, promote inclusivity and improve their overall quality of life by enabling greater social participation.

Chapter 3 Problem Statement

3.1 Control of a Humanoid Robot by People with Mobility Impairment

People with mobility impairments face substantial challenges interacting with their environment and accessing digital applications. Traditional interfaces often prove inadequate for individuals with limited or no physical mobility. The primary problem addressed by this thesis is the need for a practical, intuitive, and comfortable interface catering to this demographic. Benefiting from eye-tracking technology, this project aims to provide a lean solution to enable users to interact with the application through gaze, offering a pathway for those with limited physical capabilities to access and control the digital interface effortlessly.

Furthermore, the lack of accessible means for individuals with mobility impairments to navigate and engage with their surroundings remains a significant concern. This thesis extends its focus to address this broader challenge by entering the field of robotics. The ability to remotely control a humanoid robot opens new possibilities for individuals with mobility impairments to engage with the world, overcome physical limitations, and foster a sense of independence. This multifaceted approach tackles key problems faced by individuals with mobility impairments, providing comprehensive solutions to enhance their daily lives.

3.2 Related Work

In accessibility support, assistive technologies have emerged as invaluable tools designed to empower individuals with mobility impairments, fostering independence and inclusion. These technologies span a diverse range, employing various means and methods to address these unique challenges. There are three basic principles to achieve seamless integration of the user's capabilities. They are improving the assistive technology mechanics. Improving the user-technology physical interface. Sharing of control between the user and the technology. Powered wheelchairs, prosthetic limbs, functional electrical stimulation, wearable exoskeletons, and telepresence robots are a few great and diverse examples that this field encompasses.

In this study [8], a wheelchair-mounted 6DOF assistive robot is controlled by an Eye-gaze interface as shown in Figure 3.1. The objective is to design an eye-tracking assistive xArm 6 robot control system for individuals with a disability. The developed method is meant to enable self-assistance activities in daily living. The graphical user interface is designed and integrated with the developed control architecture to achieve the goal.

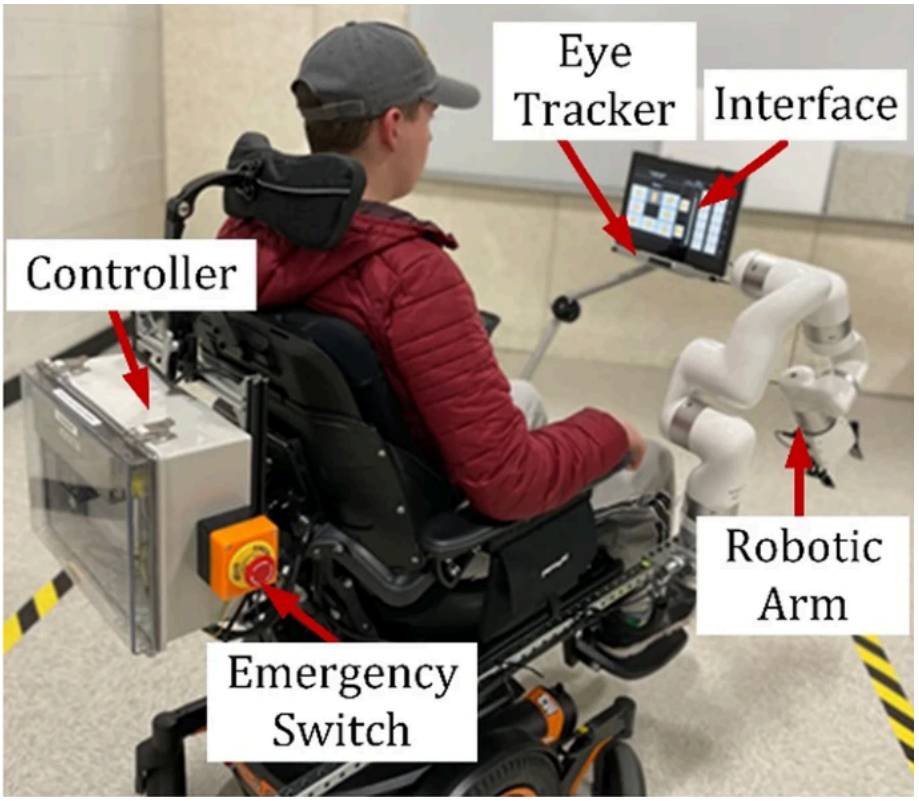

Figure 3.1: A user is sitting in a wheelchair and using the system

Another interesting study [9] on the matter adopted a different approach. It utilizes a VR headset to accomplish the eye-tracking to give control to the user. On the other side of the communication a simple robot mounted with a 360-degree camera as a telepresence device. This system is also targeted at People with Motor Disabilities to provide opportunities for social interaction and participation in public events. Figure 3.2 is the overview of Eye-gaze-controlled telepresence systems a telepresence robot with a VR headset.

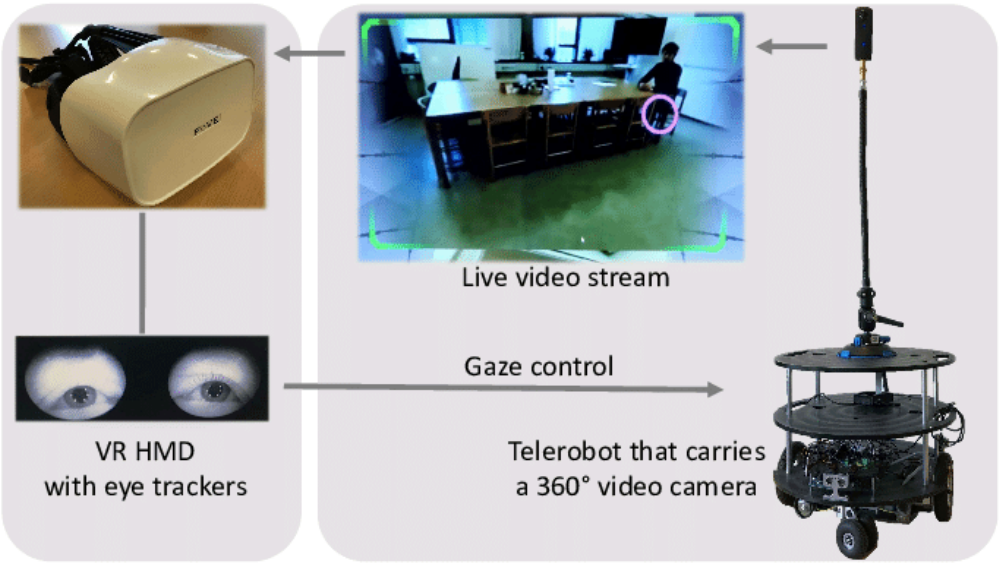

Figure 3.2: Eye-gaze-controlled telepresence systems

Chapter 4 Our Approach

4.1 Eye Tracking Camera

4.1.1 Hardware Overview

The Tobii Eye Tracker 5 [10] is a cutting-edge eye-tracking device designed for integration with various applications and systems. It is a compact and unobtrusive hardware unit equipped with advanced infrared sensors capable of precisely capturing the user's eye movements. The device is designed to be easily mounted on a computer monitor, offering a non-intrusive and seamless eye-tracking experience. With its high sampling rate and accuracy, the Tobii Eye Tracker 5 provides detailed data on gaze points, allowing developers to unlock a new dimension of interactivity in their applications.

Sensor	IS5 with custom Tobii NIR sensor (850nm)
Field of view	40 x 40 degrees
Support screen size	15” to 27” [16:9] or 30” [21:9]
Head tracking	CPU + Network (CNN) combined / 6DoF
Image sampling rate and gaze frequency	133Hz non-interlaced gaze at 33Hz
Illuminator	33Hz
Software	Tobii Experience

Table 4.1.1: Tobii Eye Tracker 5 Characteristics

Figure 4.1: Tobii Eye Tracker 5

4.1.2 Capabilities

Leveraging state-of-the-art technology, the Tobii Eye Tracker 5 boasts impressive capabilities that extend beyond traditional input methods. It can accurately track head and eye movements, including gaze direction and fixation points, enabling precise interaction with digital content. The device supports dynamic calibration, ensuring accurate tracking for users with varying eye characteristics. Additionally, it incorporates features like gaze awareness, enabling applications to respond intelligently to the user's attention, and enhancing the overall user experience. With these capabilities, the Tobii Eye Tracker 5 opens doors to innovative user interfaces and immersive experiences.

4.1.3 Tobii Experience Application

The Tobii Experience [11] application complements the Tobii Eye Tracker 5, providing users with a platform to explore the device's capabilities. This application serves as a demonstration of eye-tracking technology, allowing users to understand its potential in real-world scenarios. This application acts as a gateway for users to familiarize themselves with the technology's capabilities before integrating it into their preferred applications. This application also provides two calibration tools to help increase accuracy. The “Improve calibration” feature configures the tracker and creates a personalized calibration profile. The “Set up display” feature sets the ratio between the camera and screen scale.

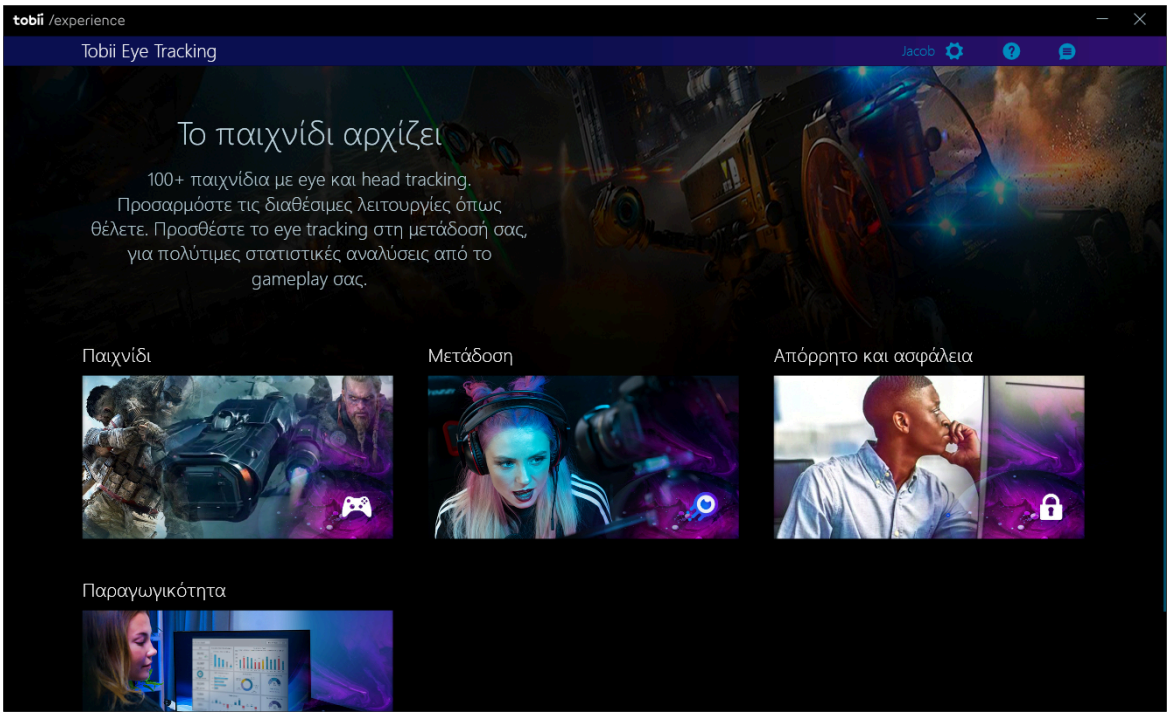

Figure 4.2: Tobii Experience application

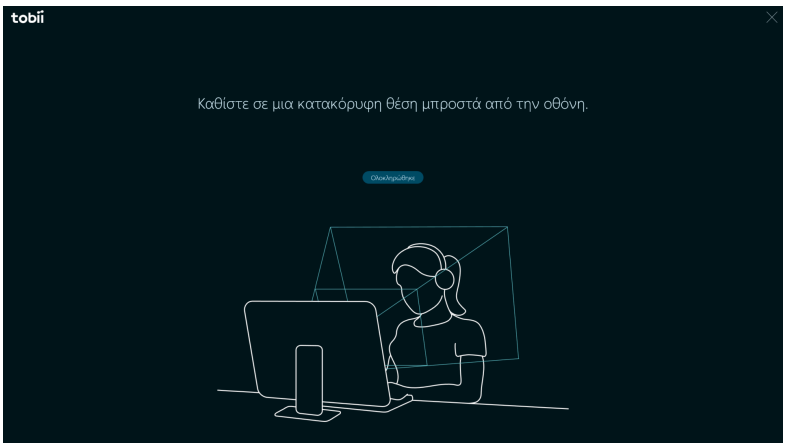

Figure 4.3: Camera calibration feature

Figure 4.4: Set up display feature

4.1.4 Unity SDK and Functionality

Tobii offers a robust Unity SDK [12] that facilitates seamless integration of the Tobii Eye Tracker 5 into Unity-based projects. The SDK provides an API and tools to access eye-tracking data within the Unity environment. Developers can effortlessly incorporate gaze data into their applications, allowing for gaze-based interactions and insights into user behavior. The Unity SDK also supports features like dynamic object mapping, enabling developers to create immersive experiences that respond intelligently to the user's gaze. With straightforward integration and comprehensive documentation, the Tobii Unity SDK empowers developers to harness the power of eye-tracking technology in their projects.

The SDK offers a native C++ API called Tobii Game Integration API as well as support for the game engine Unity. TobiiAPI is a static API that provides functionality like the collection of gaze point data, head pose data, and User presence. Finally, the Tobii G2OM tool is a machine-learned selection algorithm that accurately predicts what the user is looking at. It helps developers focus on creating great eye-tracking experiences, while also giving users a more consistent experience.

4.2 Development of Graphical User Interface

4.2.1 GUI Design

In designing the Graphical User Interface (GUI) for the eye-tracking selector, three fundamental principles guided the development process. First and foremost, the emphasis was placed on usability, particularly with the integration of big buttons tailored for easy access by the Tobii Eye Tracker 5. Recognizing the precision of gaze-based interactions, the larger buttons aim to facilitate intuitive selection, aligning with the natural eye movement patterns captured by the eye-tracking hardware. Additionally, a spread layout strategy was employed to minimize the likelihood of incorrect button selection. This approach ensures that buttons are strategically spaced, reducing the risk of inadvertent clicks and enhancing the overall accuracy of user interactions. To further optimize the user experience, the GUI incorporates a categorization system with smaller menus, fostering a lean, clean, and intuitive environment. By organizing buttons into logical groupings, the interface achieves a streamlined design, promoting efficiency and ease of navigation for users interacting with the eye-tracking selector.

4.2.2 GUI Layout

The Graphical User Interface (GUI) of the application is strategically organized into four distinct menus, with a primary menu serving as the central hub and three specialized menus dedicated to the control of specific robot movements. The core menu acts as the focal point, providing users with easy access to the various functionalities of the application. Within this core menu, users can seamlessly navigate to the specialized menus tailored for precise control. The 'Head Movement Menu' facilitates intuitive manipulation of the robot's head, allowing users to direct their gaze with precision. The 'Hand Control Menu' empowers users to interact with the robot's hand movements, enabling seamless coordination of dexterous actions. Lastly, the 'PR2 Movement Menu' focuses on the overall locomotion and positioning of the PR2 robot, providing users with comprehensive control over its movements. This modular approach not only ensures a clean and organized user interface but also allows for efficient and intuitive management of diverse robot functionalities through a well-structured menu hierarchy.

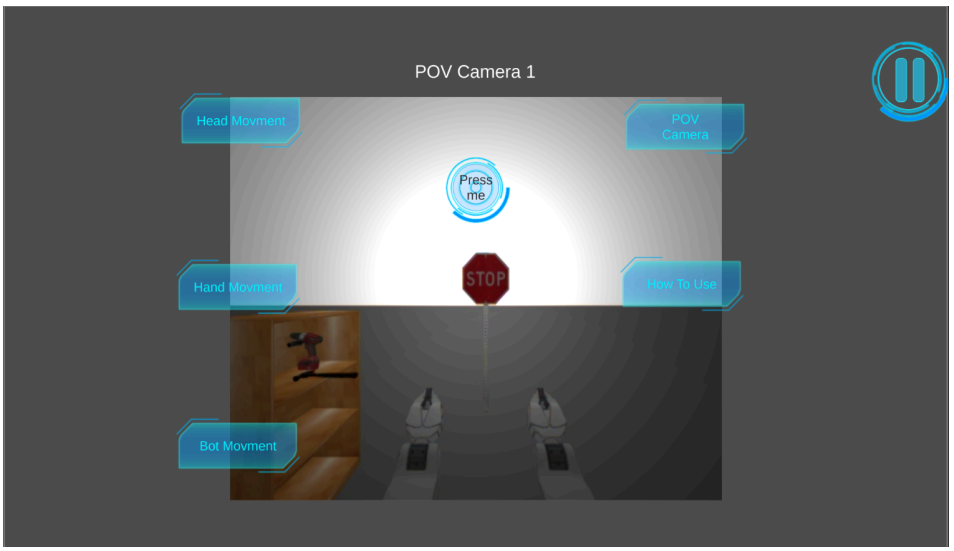

Figure 4.5: Primary Menu

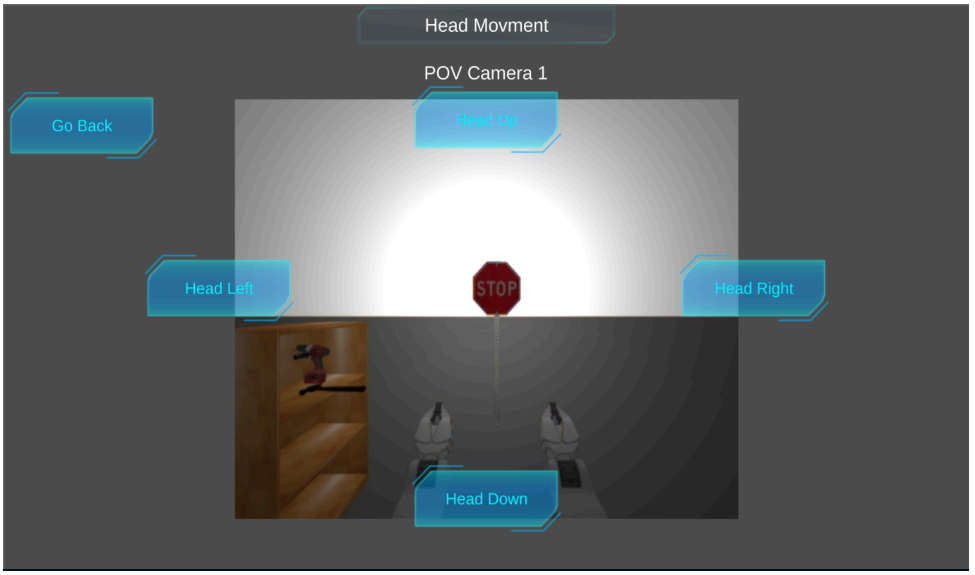

Figure 4.6: Head Movement Menu

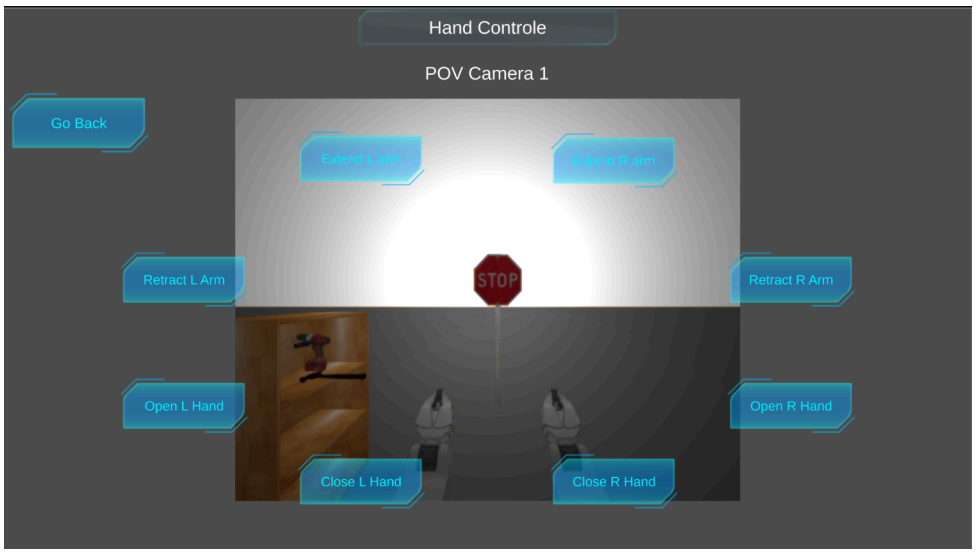

Figure 4.7: Hand Control Menu

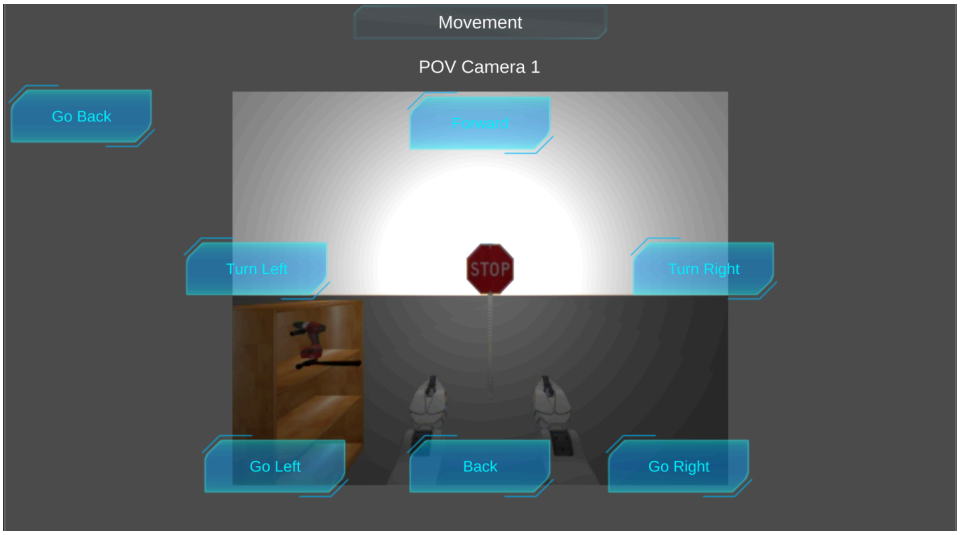

Figure 4.8: PR2 Movement Menu

4.2.3 Eye Tracker Controlled Selection

In the strategic development of the application, the Unity Game Engine emerged as the natural choice, driven primarily by its exclusive compatibility with the Tobii Eye Tracker 5 SDK. Unity, renowned for its versatility and an extensive toolkit tailored for GUI applications, provided an ideal platform for the seamless integration of eye-tracking functionalities. The Tobii SDK not only facilitates the access and processing of gaze data through a rich collection of functions but also augments Unity's capabilities with new objects and unique components. This symbiotic relationship between Unity and the Tobii SDK not only streamlines the development process but also opens up a realm of possibilities, empowering us to create a more immersive and user-centric experience.

Gaze Cursor Control:
The "Gaze_Cursor_Control_2" script serves as a fundamental component in enabling an eye-tracking-controlled cursor within the context of the Thesis development. Designed for integration with the Tobii Eye Tracker 5 and Unity SDK, this script continuously monitors the connection status of the eye tracker. Upon detection of a connected device, it fetches the current gaze point in Tobii screen coordinates using the TobiiAPI.GetGazePoint() function. This gaze point is then validated, converted to Unity screen coordinates, and used to update the cursor position. The Windows API function SetCursorPos facilitates this update, ensuring the cursor accurately follows the user's gaze. To enhance the precision of the cursor movement, the script employs a smoothing factor of 0.15, allowing for a seamless and responsive eye-tracking cursor control experience. By consistently updating the cursor position based on real-time gaze information, this script contributes to the creation of an intuitive and efficient eye-controlled interface, aligning with the overarching goals of the Thesis development.

GazeClick Script for Button Interaction:
The "GazeClick" script facilitates the interaction between the user's gaze and UI buttons. It begins by checking the availability of the Tobii Eye Tracker and retrieving the current gaze point in Tobii screen coordinates. A ray cast is then performed from the camera, simulating a ray projected from the user's gaze point onto the screen. If the ray hits a UI element, the script identifies the targeted button. To ensure accurate button detection, the script incorporates a timer mechanism, allowing the system to distinguish intentional button selection from momentary glances. If the user's gaze dwells on a button for a predetermined time, the script triggers a button click using the button.onClick.Invoke() method. This dwell time threshold ensures intentional clicks while avoiding accidental activations. This way, the script enables intuitive and precise interaction with UI buttons through gaze-based dwell times, aligning with the principles of eye-tracking technology for a user-friendly interface.

4.3 Personal Robot 2 (PR2)

4.3.1 Description

The PR2 (Personal Robot 2) [13] is a versatile and advanced humanoid robot designed by Willow Garage, renowned for its capability to perform various tasks in both research and practical applications. The robot stands approximately 165 cm tall and features a mobile base with two large wheels and a collection of sensors, making it adept at navigating its environment.

Figure 4.9: PR2 Robot

4.3.2 Hardware Overview

The PR2 is equipped with a robust set of sensors and actuators, including a Hokuyo laser rangefinder for mapping and navigation, a Kinect sensor for 3D perception, and a suite of touch and force sensors for interacting with the environment. Its two 7-DOF arms allow for precise manipulation of objects, and its torso can tilt to provide additional flexibility in its actions. The robot's design emphasizes modularity, enabling researchers and developers to swap out components for experimentation and customization easily.

4.3.3 Software Architecture

PR2 operates on the Robot Operating System (ROS), a flexible framework for writing robot software. This architecture allows for seamless communication between different hardware components and software modules. The PR2's software stack includes various ROS packages for perception, navigation, manipulation, and control. It enables the robot to perform a wide range of tasks, from recognizing objects and navigating environments to grasping and manipulating items.

4.3.4 Capabilities

The PR2 has been integral to advancing the field of robotics, serving as a platform for testing and refining new algorithms and approaches. Its dual-arm design, mobility, and extensive sensor suite make it suitable for a wide range of experiments, contributing to the development of more capable and intelligent robots. Overall, the PR2 stands as a testament

to the fusion of cutting-edge hardware and software, offering a robust platform for innovation and exploration in the field of robotics.

4.3.5 ROS Gazebo Simulation

In the comprehensive development and testing of the robotic system crucial to this thesis, the ROS Noetic Gazebo PR2 simulation [14] stands as a cornerstone. The selection of ROS Noetic as the underlying simulator, in conjunction with the Ubuntu 20.04 operating system, was a deliberate choice made to ensure compatibility with Unity (version 2021.3.3f1) and the communication tools employed in this research. This strategic alignment facilitates seamless integration between the simulated robot environment and the Unity-based graphical user interface, streamlining the development process and enhancing overall system coherence.

Throughout the deliberative process of selecting a suitable robot for this application, alternatives such as Nao [15] or Pepper [16] were considered. However, after meticulous evaluation, these options were deemed unsuitable for the telepresence robotic system envisioned in this thesis. The PR2 emerged as the optimal choice, driven by its inherent compatibility with ROS Noetic and the abundance of open-source materials and tools available for its implementation. The decision to leverage the PR2 not only aligns with the research goals but also underscores the commitment to an open and collaborative approach, essential for the successful realization of the objectives outlined in this thesis.

4.3.6 Simulation Set Up

To establish the foundation for simulating the PR2 robot in ROS Noetic, a series of key dependencies and libraries are meticulously installed. The process initiates with the installation of dependencies for SDformat, a simulation description format crucial for accurate representation. This encompasses acquiring development tools, libraries, and necessary packages essential for the subsequent build of SDformat. Following this, SDformat is cloned from its GitHub repository, specifically opting for version 'sdf6' to ensure compatibility with the intended simulation framework. A dedicated build directory is then created, configuring the build using CMake, and SDformat is compiled with a predefined installation prefix.

Next in the sequence is the installation of ROS Convex Decomposition, a pivotal library for decomposing convex shapes within the simulation. A fresh workspace is established for Convex Decomposition, wherein the GitHub repository is cloned, and the workspace is built using catkin_make. The setup script is thoughtfully added to the user's bashrc file, ensuring seamless sourcing for future use.

Simultaneously, the installation of ROS ivcon, or Inventor File Converter, is undertaken. This ROS package plays a crucial role in converting Inventor files. Similar to Convex

Decomposition, a dedicated workspace is created, the ivcon repository is cloned, and the workspace is built using catkin_make. As a final step, the setup script is appended to the bashrc file for streamlined future access.

The culmination of these preparatory stages involves the creation of a workspace specifically tailored for the PR2 simulator. This encompassing workspace hosts a selection of PR2-related repositories, including the simulator itself (pr2_simulator), the mechanism packages (pr2_mechanism), common packages (pr2_common), and mechanism messages (pr2_mechanism_msgs). With the framework in place, the workspace undergoes building using catkin_make, solidifying the essential components required for simulating the PR2 robot within the ROS Noetic environment. This meticulous setup, comprising SDformat dependencies, Convex Decomposition, ivcon installation, and the PR2 workspace creation, forms the robust underpinning for the ensuing simulations and experiments in this research endeavor.

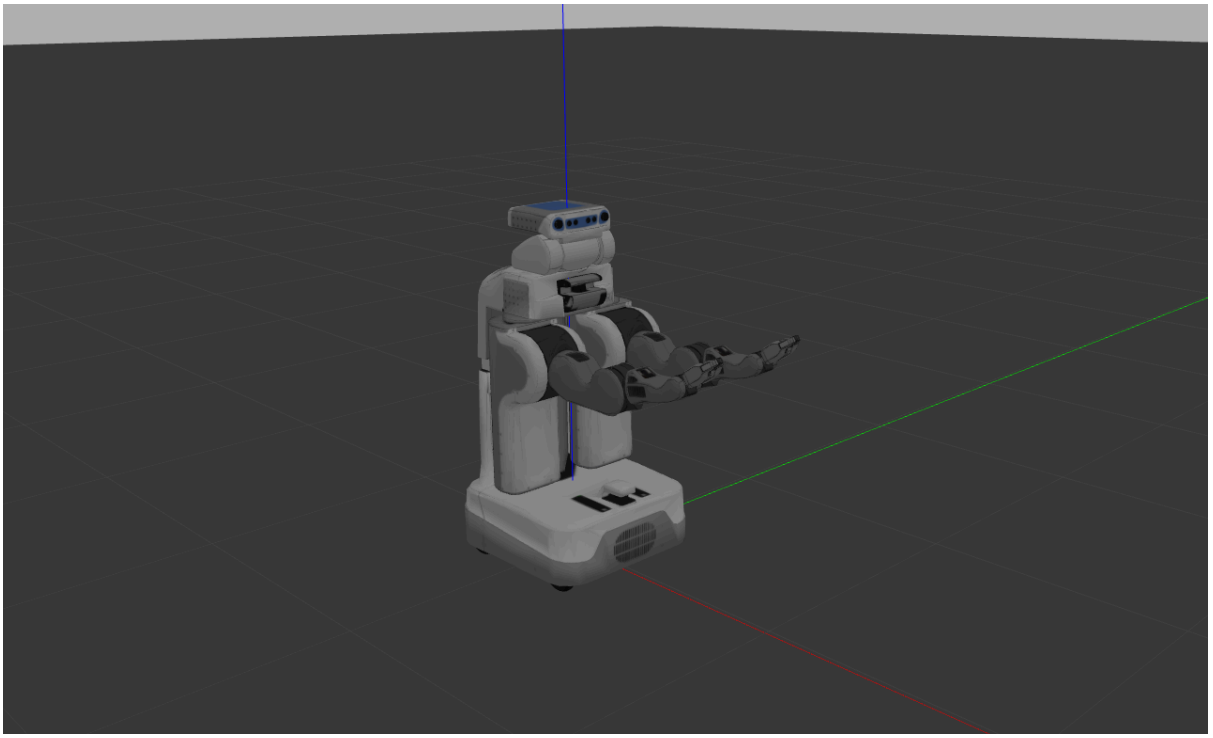

Figure 4.10: PR2 Empty World Simulation

4.4 Bridge between Unity and ROS

Establishing a seamless communication bridge between Unity and ROS serves as a crucial component for the successful operation of the application. This communication interface plays a pivotal role in facilitating the exchange of control commands from the Unity-based graphical user interface (GUI) to the robot, enabling users to interact with the system intuitively. Additionally, it serves as a conduit for relaying real-time feedback data from the robot back to the GUI, ensuring that users receive timely and accurate information about the robot's state and actions. This bidirectional communication framework not only

enhances the user experience by providing a responsive control mechanism but also enables a dynamic and interactive connection between the Unity environment and the ROS-controlled robot, forming the backbone of the application's functionality.

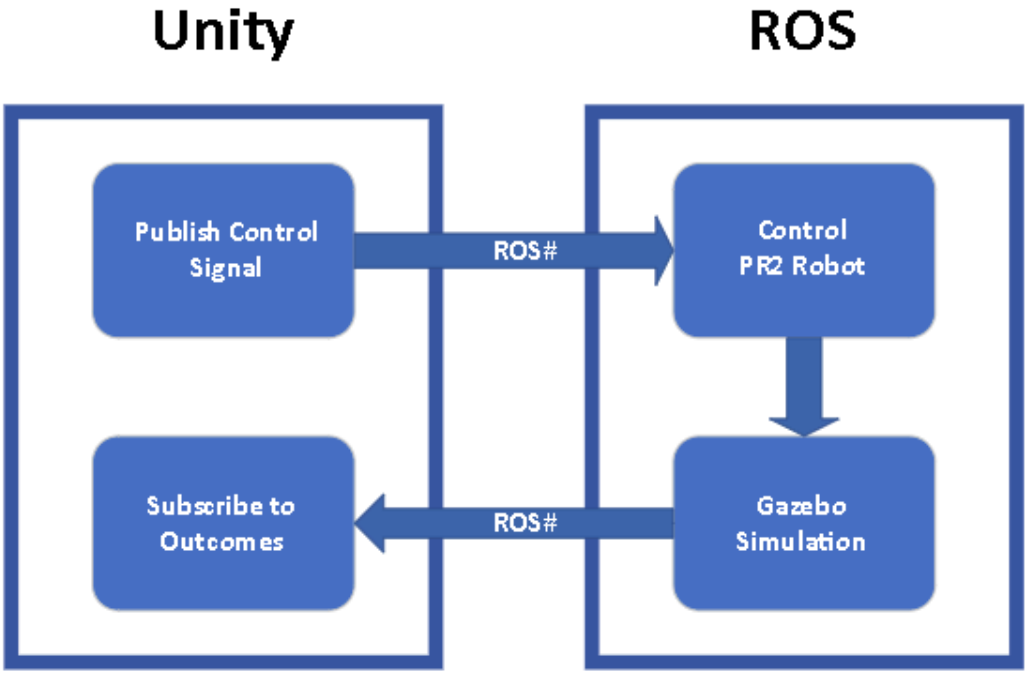

Figure 4.11: Bridge between Unity and ROS

4.4.1 ROSbridge package (ROS Side)

The ROSbridge package [17],[18] serves as a crucial middleware on the ROS side, acting as a communication bridge between the ROS environment and external systems, such as Unity. It facilitates the exchange of messages and commands in a format that is compatible with both ROS and non-ROS applications. During the simulation, a ROSbridge server is initiated, running concurrently with the simulation environment. This server becomes the communication hub, allowing external entities like the Unity-based GUI to interact with the ROS-controlled robot seamlessly.

While the simulation runs, the ROSbridge server operates as an intermediary, handling communication between the ROS environment and external systems. This integration ensures that the simulation environment remains responsive to commands and data requests from Unity, creating a cohesive and interactive experience.

4.4.2 ROS# package (Unity Side)

The ROS# package [19] is designed for Unity and is the counterpart to the ROSbridge on the Unity side. It includes essential components like the ROSConnector object and RosConnector script, which function as the client connecting to the ROSbridge server. Through the utilization of sockets, the ROSConnector establishes a communication link

between the Unity environment and the ROSbridge server, enabling the GUI to send control commands and receive real-time feedback data from the ROS-controlled robot.

In the Unity environment, the ROS# package seamlessly integrates into the project. The ROSConnector object and associated script, RosConnector, play a pivotal role as they connect to the ROSbridge server via socket. This connection establishes a bidirectional communication channel, allowing the Unity-based GUI to efficiently send commands to the robot and receive timely updates on the robot's state. The robust integration of ROS# ensures effective communication between Unity and ROS, forming a cohesive framework for controlling and monitoring the robot during the simulation.

4.5 PR2 Control Methods

In controlling the PR2 robot through the ROS bridge, the focus is on enabling seamless communication between the Unity-based graphical user interface (GUI) and the simulated robot in ROS. The primary challenge lies in the disparity between supported message types in the ROS# package, where some critical types like

"pr2_controller_msgs/Pr2GripperCommand" for gripper movements and

"pr2_controllers_msgs/JointTrajectoryActionGoal" for joint movements

are not natively supported. To overcome this limitation, two distinct methods, namely “pr2_movement_pub” and “command_pub,” are employed for publishing command messages.

4.5.1 PR2 Movement Publisher

The `pr2_movement_pub` method efficiently handles GUI buttons associated with the movement of the robot's base. It serves as a critical component for controlling the movement of the PR2 robot within the simulated environment. This script is part of the RosSharp.RosBridgeClient namespace emphasizes its role in communicating with the ROS environment. The script inherits from the UnityPublisher class, which is specialized for publishing messages of type MessageTypes.Geometry.Twist. This message type is commonly used in ROS to represent linear and angular velocities, making it suitable for controlling the robot's movement.

The script is equipped with buttons in the Unity Inspector, each corresponding to a specific movement command: forward, backward, right turn, left turn, move right, and move left. Upon pressing any of these buttons, a corresponding method (e.g., `MoveForward`, `TurnRight`) is triggered. These methods set the linear and angular components of the `Twist` message according to the desired movement, and a predetermined number of frames (`movementFrames`) is assigned for the duration of the movement. The script

continuously publishes these `Twist` messages to the ROS environment, effectively controlling the robot's movement. This approach ensures a smooth and controlled motion of the PR2 in the simulated environment, offering a user-friendly way to interact with the robot.

4.5.2 PR2 Joint Control

On the other hand, for gripper and joint controls, a thoughtful bypass solution is implemented using the `command_pub` method. Here, instead of attempting to publish unsupported message types directly to the joint controller of the robot, the GUI sends straightforward "Int16" type messages to a dedicated handling node residing in the ROS PR2 simulation package named "pr2_limb_control." The node, specifically the "gripper_command_listener.py," subscribes to a Unity-created topic through the `command_pub` script. When an "Int16" message is received, the node interprets the unique values associated with each control button press, subsequently publishing the correct and supported message type for gripper or joint movements. This strategic by-pass approach successfully facilitates the GUI buttons in orchestrating predetermined movements of the PR2 robot, offering a streamlined and intuitive remote control experience.

This implementation showcases the versatility and adaptability achieved through the ROS bridge, allowing the GUI to exert precise control over the simulated robot's various functionalities, including base, gripper, and joint movements, through a simplified button interface.

Figure 4.12 gives an overview of the pr2 simulation in ros noetic graphically illustrated by the rqt_graph. This graph shows Nodes and the Topics that provide the communication between them.

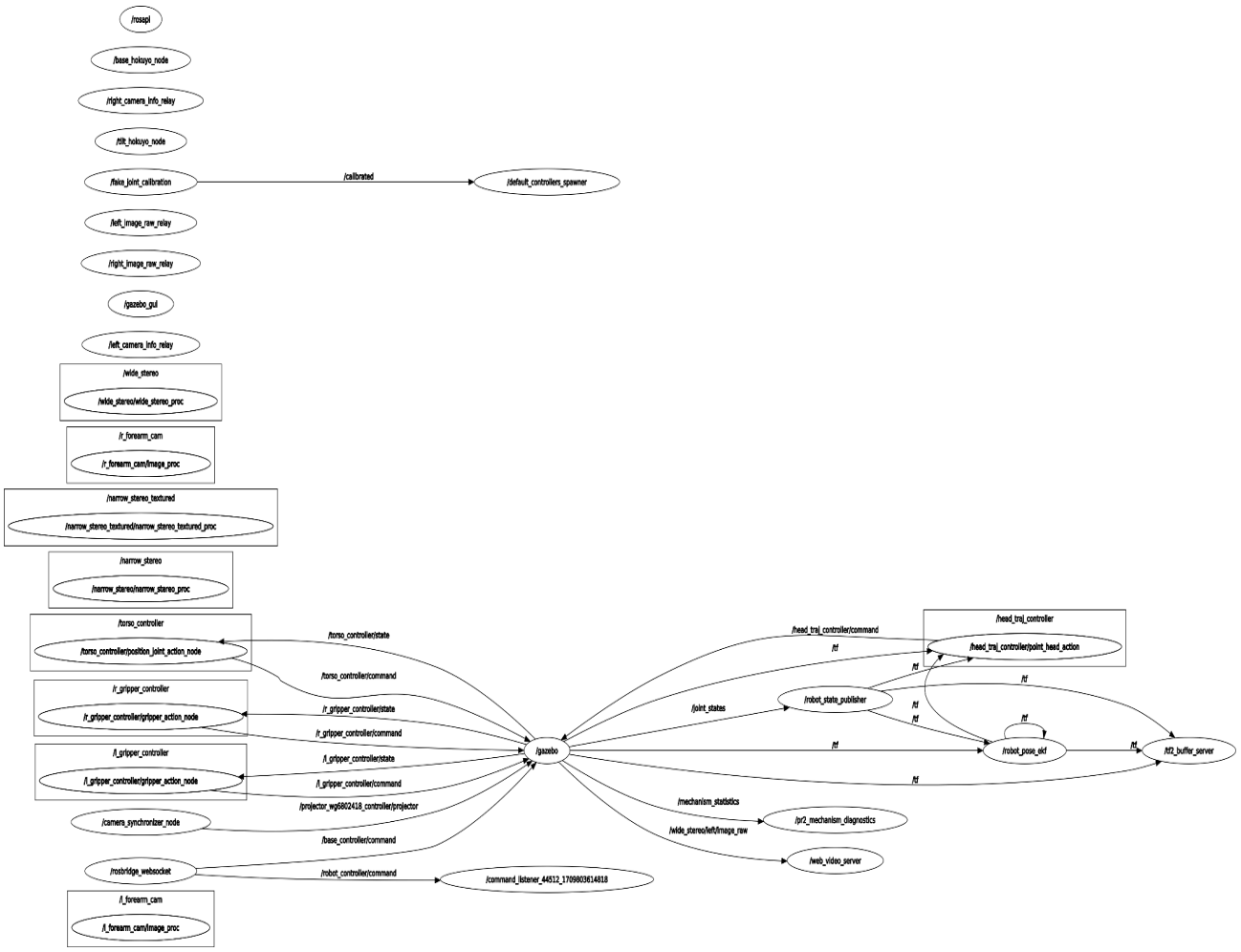

Figure 4.12: ROS Nodes Graph presenting Nodes Only.

4.6 PR2 POV Streaming

4.6.1 web_video_server (ROS)

In the pursuit of establishing a bidirectional communication link between the user and the PR2 robot, it becomes imperative to provide the user with real-time visual feedback of the robot's surroundings. This necessitates the streaming of video data from the onboard camera(s) of the PR2 back to the graphical user interface (GUI). To facilitate this, the integration of the `web_video_server` ROS package [20] comes into play. This ROS package serves as a bridge, enabling the streaming of video data from the robot's cameras to external applications, such as the Unity-based GUI.

4.6.2 WebStream (Unity)

The `WebStream` script in Unity acts as a crucial component in this video streaming setup. Its primary objective is to connect to the streaming feed provided by the `web_video_server` ROS package and project the video frames onto the GUI. The script

utilizes a series of functions and parameters to achieve this functionality. It defines a URL variable that points to the specific streaming topic and quality settings. The script leverages Unity's texture and rendering capabilities to display the video feed on a designated mesh renderer within the GUI.

The `GetVideo` function establishes a connection to the streaming server by creating an HTTP request to the provided URL. It then retrieves the response stream, initiating the continuous retrieval of video frames. The `GetFrame` coroutine manages the asynchronous retrieval of video frames, ensuring a seamless and continuous display of the streaming feed. By employing the `FindLength` function, the script accurately determines the length of each frame, allowing for proper decoding and rendering.

Notably, the `WebStream` script provides a practical solution for integrating live video streaming into the Unity GUI, fostering enhanced user monitoring and interaction with the PR2 robot. The seamless communication achieved through this script contributes to a more immersive and intuitive user experience, enabling effective remote control and observation of the robot's activities.

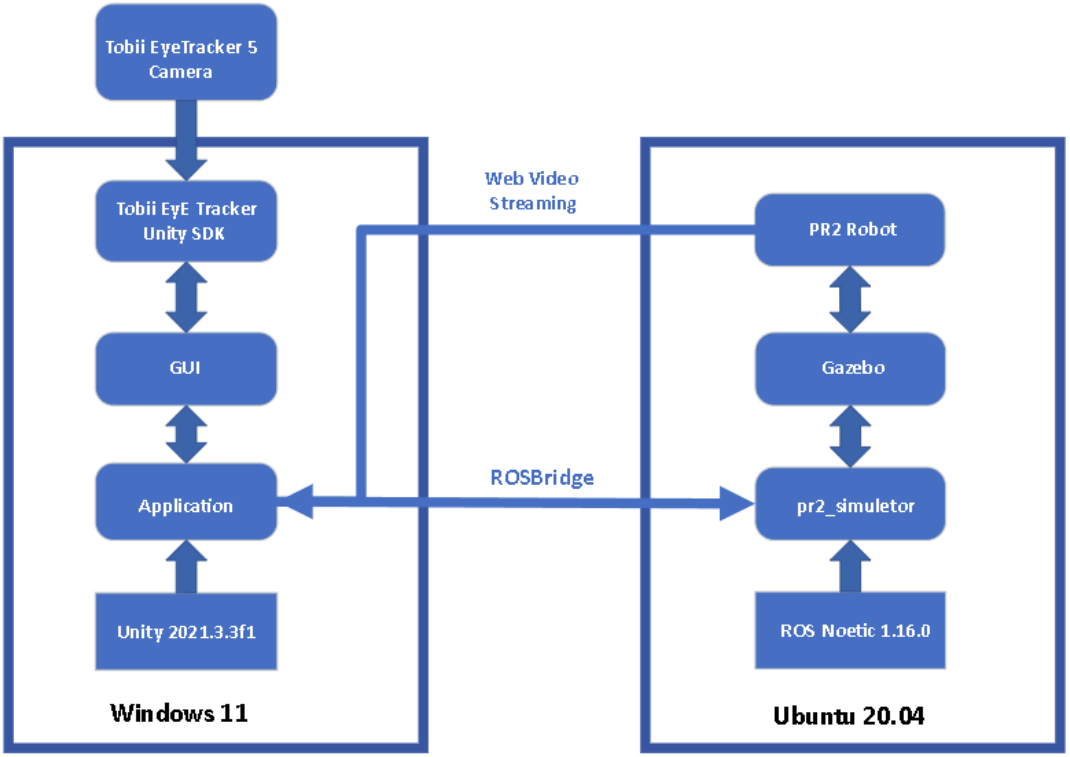

Figure 4.13: Project Overview

The system was built to cater to the needs of bedridden people with tetraplegia or other mobility impairments. To provide a surrogate body and help with accessibility to that surrounding environment and telepresence capabilities through live video streaming from the robot.

Chapter 5 Results

5.1 User Study

A user study was conducted to playtest the application (Figure 5.1) and unearth run-time problems through the impartial lens of the users. The system was given to four healthy participants aged twenty-three (23) to fifty-three (53). For this test, two computers were used, arranged side to side. One is equipped with the eye-tracking camera running the application on Windows 11 (Figure 5.2), and the other hosts the server, loaded with the PR2 simulation on Ubuntu 20.04 (Figure 5.3). A video demonstrating the application demo is available on YouTube [21]. On the top left side of the video the GUI of the application can be seen, while on the bottom right there is a view of the PR2 Simulation environment. The participant can be seen both in the upper right and bottom left corner, using the application hands-free, only by Eye-Gaze interaction.

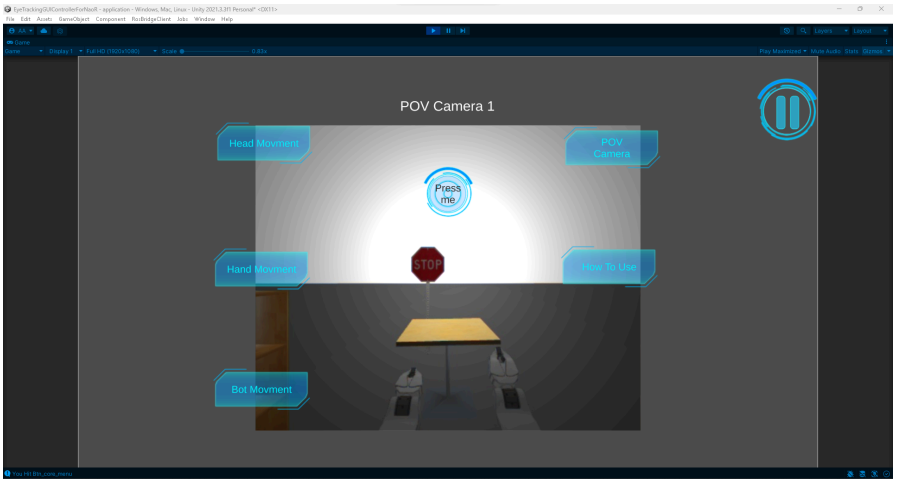

Figure 5.1: Application running on Unity on Windows 11 machine.

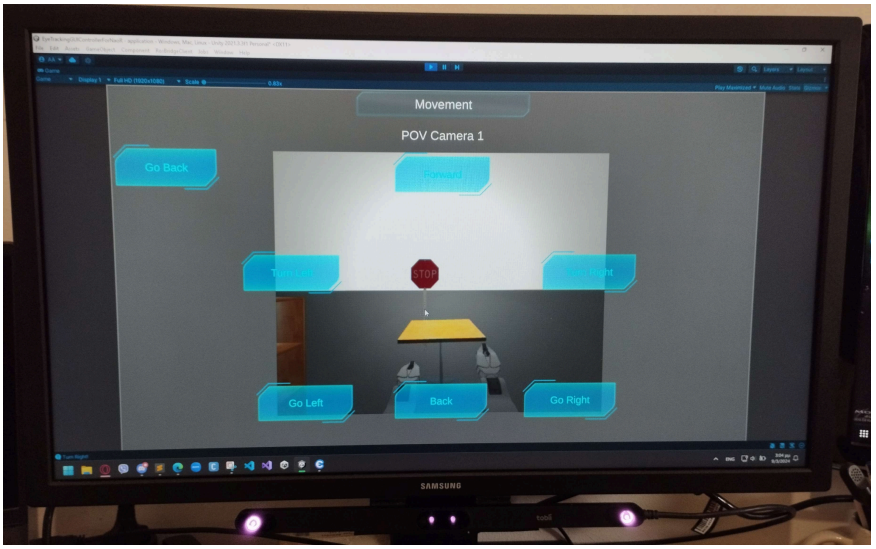

Figure 5.2: GUI and Tobii Eye Tracker 5 camera placement.

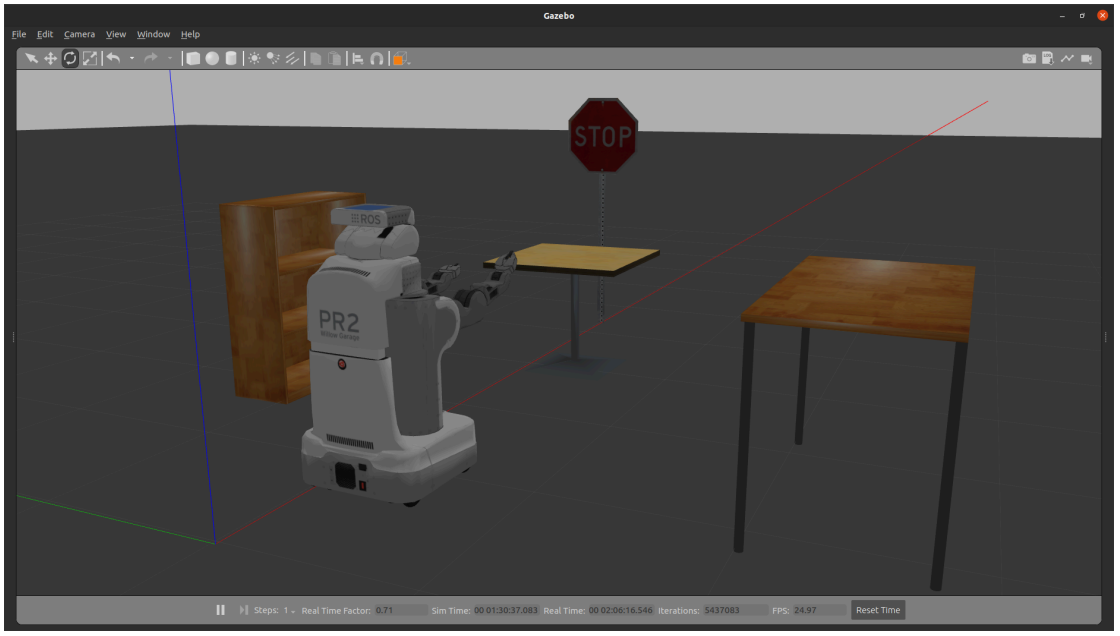

Figure 5.3: PR2 ROS Noetic Simulation on Ubuntu 20.04

5.1.1 User Study Procedure

Before running each participant's playtest, a mandatory eye-tracker calibration was done. Creating a personal movement profile from the Tobii Eye-Tracker SDK is recommended to increase the quality of the outcome. After the calibration, the participants were given a quick introduction to the scope of this study and were instructed to limit their interaction with the application to gaze interaction. They were encouraged to do a full walk-through of the options of the application, after which were left to manipulate it as they pleased.

The study was conducted in two rounds. The first round purpose was to collect first thoughts on the design and collect any complaints and potential bugs on the application's runtime. The second round was for the users to assess the quality of some implemented recalibrations. In both rounds, the participants were given a questionnaire at the end in order to express their thoughts in a compact form. The questionnaire had five simple questions:

1. Rate the intuitiveness of the application on a scale from one to ten, where one is not intuitive at all and ten is extremely intuitive.
2. Rate the usability of the application on a scale from one to ten, where one is not easy to use, and ten is extremely easy to use.
3. Rate the application’s user environment on aesthetics from one to ten, where one is chaotic-disorganized, and ten is clean and user-friendly.
4. Rate the response time of the robot after a successful button Click from one to ten, where one is unresponsive, and ten is immediate response.

5. What problems did you face during the trial?

5.1.2 User Study First Trial

Questions **Participants**	Q1:	Q2:	Q3:	Q4:	Q5:
Participant 1 (age 23)	9/10	7/10	10/10	10/10	Cursor drifting
Participant 2 (age 25)	9/10	6/10	10/10	10/10	Cursor drifting
Participant 3 (age 28)	9/10	7/10	10/10	10/10	Cursor drifting
Participant 4 (age 53)	7/10	5/10	10/10	10/10	Cursor drifting

Table 5.1.2 User Study First Trial Results

The results reflect that the application succeeds in delivering an intuitive and user-friendly application. The response time of the simulation seemed to be within desirable parameters proving that the ROSbridge implementation is working under lean parameters without overwhelming the network. Only one aspect of this iteration had a significant lack in its intended functionality.

After the end of the first trial it was apparent the need to increase the efficiency of the eye-tracking controlled cursor. The problem is noted here as “Cursor drifting”, because it describes the behavior of the cursor when it approaches the frame of the screen. This makes the core of the screen the more efficient area and the borders of the screen the more erratic area. To combat this issue two recalibrations were adopted. The first was to implement a smoothing algorithm. This function smoothens the cursor movement by applying a smoothing factor to the calculated gaze points over time. The second modification was made on the peripheral buttons of the GUI, expanding the box colliders towards the erratic aria as shown in Figure 5.1.2. This, in effect, compensates for the Cursor Drift and focuses on the location that the user is looking at and beehives within the desirable parameters.

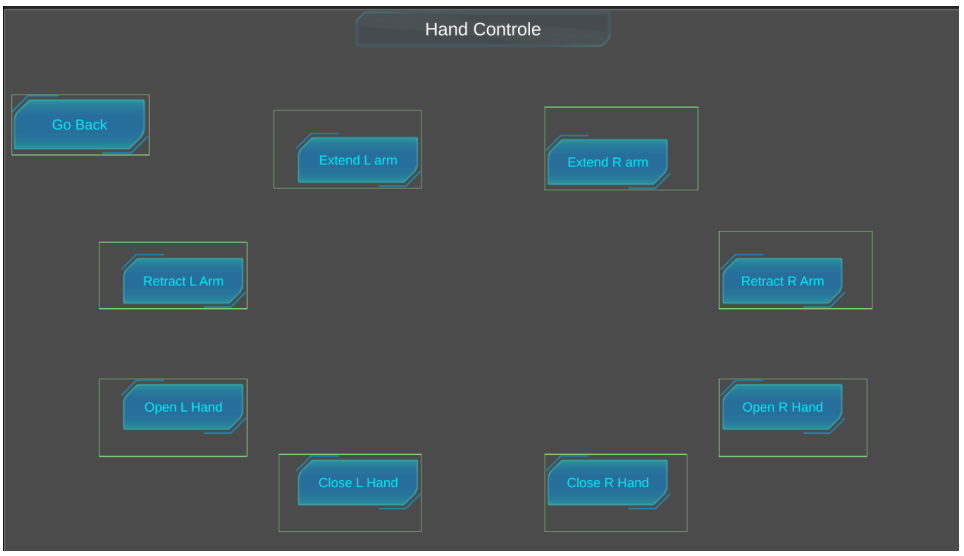

Figure 5.4: Expansion of the battens' box colliders

5.1.3 User Study Second Trial

Questions **Participants**	Q1:	Q2:	Q3:	Q4:	Q5:
Participant 1 (age 23)	10/10	10/10	10/10	10/10	_
Participant 2 (age 25)	10/10	9/10	10/10	10/10	_
Participant 3 (age 28)	10/10	10/10	10/10	10/10	_
Participant 4 (age 53)	9/10	8/10	10/10	10/10	_

Table 5.1.3 User Study Second Trial Results

With these modifications the next iteration of the application had significant improvement in the participant's experience. The Second trial showed that the recalibrations managed to bridge the gap between the previous iteration and the target capabilities. The user experience assessment reveals that the developed system delivers an intuitive, streamlined, and user-friendly interface. It provides a unique solution in situations where a standard mouse is impractical. Finally, in the last trial, the participants were asked their preference between a monitor mount eye-tracker, a VR, and an IEG headpiece as an alternative prolonged GUI control methodology. They chose the non-invasive and less cumbersome approach, the monitor approach.

Chapter 6 Conclusions

6.1 Conclusion

This culmination of research and development marks the completion of a project aimed at revolutionizing assistive technologies for individuals facing complete mobility impairment. Through the integration of robotic systems and eye-tracking technology, this endeavor sought to empower users with innovative means of interaction, navigating the challenges posed by severe mobility limitations. The project's methodologies encompassed a comprehensive exploration of gaze-based controls, leveraging the capabilities of eye-tracking technology and integrating them with a robotic system, exemplified by the PR2 simulation in ROS. This section provides a concise overview of the project's implementation strategies, highlighting the innovative blend of technologies employed to create a more accessible and user-friendly solution for individuals with mobility challenges.

6.2 Discussion

The project's multifaceted exploration traversed through the domains of gaze-based interaction, robotic control, and seamless integration of these technologies within the context of assistive applications. The results obtained reflect a commendable stride forward in providing individuals with complete mobility impairment an avenue for enhanced independence and interaction. The gaze-based cursor control, though not a perfect replacement for traditional mouse interfaces, presents a unique opportunity for scenarios where standard control methods prove challenging.

A significant focus was dedicated to refining the conversion algorithms mapping gaze coordinates to the system cursor, ensuring a responsive and intuitive control experience. The challenges faced in achieving precision were met with iterative testing and adjustments, resulting in a system with a manageable margin of error, particularly in the context of GUI button interactions.

Video streaming from the PR2 simulation's onboard cameras added a crucial dimension to the user experience. However, challenges in maintaining optimal video quality persisted, necessitating occasional compression to mitigate latency. The robustness of communication between Unity and ROS was showcased, although the compatibility issues with certain message types highlighted potential challenges in implementing non-predetermined movements directly controlled by Unity.

6.3 Future Work

In the realm of future work, several promising directions unfold, driven by the insights and challenges encountered in this pioneering project. While initially contemplated, the integration of blinking tracking functionality confronted limitations owing to the absence of support from the Tobii Eye Tracker 5 SDK. Exploring alternative blinking detection methods, such as leveraging external webcams or adopting more advanced eye-tracking technologies, holds the potential to introduce nuanced and natural user interactions, further refining the user experience.

Expanding the platform's capabilities could involve the implementation of reverse kinematics on the PR2 robot, enhancing the versatility of its arm movements and ultimately increasing accessibility. This addition would empower users to manipulate the robot's arms with greater precision, opening up new possibilities for interactive tasks.

Moreover, the incorporation of sound transfer capability between the robot and the application can elevate the telepresence aspect of the system. Enabling users to both hear the robot's environment and express themselves through the robot enhances the overall user experience.

Moreover, advancing the robot's movement capabilities by integrating autonomous obstacle avoidance using onboard sensors emerges as a critical aspect of future development. This enhancement would contribute to the system's autonomy, allowing the robot to navigate its environment more effectively and further reducing the reliance on user input for basic navigation tasks. These forward-thinking ideas underscore the potential for continual innovation and improvement in the realm of assistive technologies for individuals with mobility impairments.

6.4 Lessons Learned

The journey of conceptualizing, developing, and refining this assistive technology solution imparted invaluable lessons. The iterative nature of design and testing underscored the importance of user feedback and adaptability in crafting a system that genuinely addresses the needs of its users. The challenges faced in communication between Unity and ROS emphasize the significance of a flexible and comprehensive communication interface for future developments in similar interdisciplinary projects. Overall, this project serves as a stepping stone, laying the groundwork for continued innovation in assistive technologies for individuals facing complete mobility impairment.

Bibliography

[1] Eye-TrackingSystem.
URL: https://www.sciencedirect.com/topics/psychology/eye-tracking-system

[2] Unity. URL: https://unity.com

[3] ROS URL: https://www.ros.org

[4] Gazebo URL: https://gazebosim.org/home

[5] Rupp R, Biering-Sørensen F, Burns SP, Graves DE, Guest J, Jones L, et al. (2021-03-01). "International Standards for Neurological Classification of Spinal Cord Injury: Revised 2019". Topics in Spinal Cord Injury Rehabilitation.

[6] The Encyclopedia of World Problems and Human Potential, UIA, Tetraplegia.

[7] Cowan RE, Fregly BJ, Boninger ML, Chan L, Rodgers MM, Reinkensmeyer DJ. Recent trends in assistive technology for mobility. J Neuroeng Rehabil. 2012;9(1):1–8.

[8] Sunny, M.S.H., Zarif, M.I.I., Rulik, I. *et al.* Eye-gaze control of a wheelchair mounted 6DOF assistive robot for activities of daily living. *J NeuroEngineering Rehabil* 18, 173 (2021). https://doi.org/10.1186/s12984-021-00969-2

[9] Zhang, Guangtao, Hansen, John, Minakata, Katsumi, Alapetite, Alexandre, Wang, Zhongyu, 2019/03/08, Eye-Gaze-Controlled Telepresence Robots for People with Motor Disabilities 10.1109/HRI.2019.8673093

[10] Tobii Eye tracker 5, URL: https://gaming.tobii.com/product/eye-tracker-5/

[11] Tobii Experience, URL: https://gaming.tobii.com/getstarted/

[12] Tobii Eye tracking 5 Unity SDK,
URL: https://developer.tobii.com/pc-gaming/unity-sdk/

[13] PR2 Robot, URL: https://robotsguide.com/robots/pr2

[14] PR2 ROS Simulation. URL: http://wiki.ros.org/pr2_simulator

[15] Nao Robot. URL: https://www.aldebaran.com/en/nao

[16] Pepper Robot, URL: https://www.aldebaran.com/en/pepper

[17] ROS Bridge Package, URL: http://wiki.ros.org/rosbridge_suite

[18] ROS Bridge Documentation,
URL: https://carla.readthedocs.io/projects/ros-bridge/en/latest/

[19] ROS# Package, URL: https://assetstore.unity.com/packages/tools/physics/ros-107085

[20] Web Video Server ROS Package: URL: http://wiki.ros.org/web_video_server

[21] Demo video of playtesting: URL: https://youtu.be/9SFLfWfPE7M