University of Minho
School of Engineering

Eduardo Alexandre Carvalho Fernandes

Service Robot remote control system (VR Headset + Pose Estimation)

march 2023

University of Minho
School of Engineering

Eduardo Alexandre Carvalho Fernandes

Service Robot remote control system (VR Headset + Pose Estimation)

Masters Dissertation
Mestrado Integrado em Engenharia Eletrónica
Industrial e Computadores

Dissertation supervised by
Professor Doutor Fernando Ribeiro

march 2023

Acknowledgements

I would like to express my heartfelt gratitude to the individuals who made this dissertation possible over the course of the past five years. My sincere thanks go to my loving parents, José Fernandes and Carmo Carvalho, who have always supported me and provided me with unconditional love and sacrifice. My gratitude also extends to my sisters, Cláudia and Beatriz Fernandes, for their constant affection and support. I am deeply grateful to my grandfather, Domingos Carvalho, for his knowledge and for being a constant presence in my life.

I am also grateful to Professor Fernando Ribeiro for providing me with the opportunity and support throughout my dissertation journey. I would like to thank the Laboratory of Automation and Robotics members for creating an excellent working environment and providing a positive team spirit. Special thanks to Syed Fawad Adil for his invaluable assistance with the robotic arm, and to Marco Luís and Tiago Ribeiro for the invaluable help and advice they provided throughout my dissertation.

I am forever grateful to my academic colleagues for their support and for all the incredible memories we have made together along the way. These memories will always be cherished and never forgotten.

Last but not least, I would like to express my gratitude to my childhood friends, particularly Diogo Ribeiro and Ângelo Abreu, for their constant support, friendship, and being there for me in my time of need.

Statement of Integrity

University of Minho, Guimarães, march 2023

Eduardo Alexandre Carvalho Fernandes

Abstract

Teleoperation of service robots has been a highly sought-after area of research in recent years. The ability to remotely operate a robot offers numerous benefits in terms of human safety, enabling the execution of dangerous tasks without the risk of significant harm.

This dissertation presents a comprehensive examination of the theoretical foundations of pose estimation algorithms. The Mediapipe framework was employed for human pose detection, in combination with an RGB-D camera for three-dimensional pose mapping. A virtual reality interface was developed to provide access to the robot's vision, allowing for an immersive experience for the operator.

In the first phase of experimentation, tests were conducted in a simulated environment using CoppeliaSim and the humanoid robot NAO to simulate the service robot. Communication between the control script and the simulator was established using the Robot Operating System (ROS) framework and a control node in Python. The goal of these tests was to evaluate the performance of the developed algorithm through tasks such as object handling.

Finally, the algorithm was evaluated in a real-world environment using the CHARMIE robot, with a focus on the manipulation of an object through the virtual reality interface.

Keywords Teleoperation, Service robot, Virtual reality, Pose estimation

Resumo

Teleoperação de robôs de serviços tem sido uma área muito procurada nos últimos anos. A capacidade de operar um robô remotamente traz muitas vantagens quando se trata da segurança dos seres humanos, permitindo que tarefas perigosas sejam realizadas sem o risco de perdas cruciais.

Esta dissertação apresenta as bases teóricas dos algoritmos de estimação de pose. A framework Mediapipe foi utilizada para deteção da pose humana, em conjunto com uma câmera RGB-D para mapeamento tridimensional da pose. Uma interface de realidade virtual foi desenvolvida para aceder à visão do robô, permitindo uma experiência imersiva por parte do operador.

Numa primeira fase, os testes foram realizados num ambiente de simulação, usando o CoppeliaSim e o robô humanóide NAO para simular o robô de serviço. A comunicação entre o *script* de controlo e o simulador foi configurada usando a framework ROS e um nó de controlo em Python. O objetivo dos testes é avaliar o desempenho do algoritmo desenvolvido, através de testes como a manipulação de objetos. Finalmente, o algoritmo foi testado em ambiente real no robô CHARMIE, onde o objetivo é a manipulação de um objeto usando a interface de realidade virtual.

Palavras-chave Teleoperação, Robô de serviços, Realidade virtual, Estimação de pose

Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Objectives . . . 2

1.3 Scientific Contributions . . . 3

1.4 Document Structure . . . 4

2 State of the Art **5**

2.1 Service robotics . . . 5

2.2 Teleoperation . . . 7

2.2.1 Applications . . . 8

2.3 Virtual Reality . . . 10

2.3.1 Head-Mounted Displays . . . 10

2.4 Pose Estimation . . . 13

2.5 Conclusions . . . 14

3 Algorithm Architecture: Pose Estimation Model **16**

3.1 Pose Estimation Framework: Mediapipe . . . 16

3.1.1 BlazePose Topology . . . 16

3.1.2 BlazePose Pose Tracking . . . 17

3.1.3 BlazePose Tracking Model . . . 18

3.1.4 Mediapipe FaceMesh . . . 19

3.1.5 Mediapipe Hands . . . 20

3.2 2D to 3D . . . 22

3.3 Body Pose Estimation by Image Processing . . . 24

3.3.1 Shoulder Pitch Calculation . . . 26

3.3.2 Shoulder Roll Calculation . . . 27

3.3.3 Elbow Roll Calculation 28
3.3.4 Elbow Yaw Calculation 28
3.3.5 Wrist Yaw 30
3.3.6 Hand Opening and Closing 31
3.3.7 Head Pitch and Head Yaw 32
3.4 Virtual reality 33
3.5 Conclusions 36

4 Simulation Framework 37
4.1 Simulation Environment: CoppeliaSim 37
4.1.1 NAO Robot 38
4.1.2 Head joints 39
4.1.3 Arm joints 40
4.1.4 Hand links 41
4.2 ROS Framework 41
4.3 Simulation Tests 45
4.3.1 Controlling the robot 45
4.3.2 Object manipulation 46
4.4 Robot vision through VR 46
4.4.1 Object manipulation using VR 47
4.5 Conclusions 47

5 Real World Tests 49
5.1 CHARMIE Antropomorphic Arm 49
5.2 CHARMIE Neck 52
5.3 Controlling the antropomorphic arm 53
5.4 Controlling the robot 55
5.5 Conclusions 57

6 Conclusions 58
6.1 Further Work 59

List of Figures

1.1 Reachy tests through VR [image from [2], with pemission from Michael Cussel]. 2

1.2 CHARMIE [4]. 4

2.1 Service robots. Units sold in 2018 and 2019, potential development 2020-2023 [9]. . . 6

2.2 Service robots. Sales value in 2018 and 2019, potential development 2020-2023 [9]. . . 6

2.3 Service Robots. Units sold in top 3 applications in 2018 and 2019, potential development 2020-2023 [9]. 7

2.4 Robonaut, a humanoid robot designed and developed by NASA [13]. 8

2.5 Communication delay in space [image from [15], with permission from Tamás Haidegger]. 9

2.6 DaVinci surgical system [17]. 10

2.7 Depiction of the horizontal and vertical field of view [21]. 11

2.8 Illustration of binocular disparity, where each eye captures a unique angle of the object [21]. 11

2.9 Oculus Quest 2. 12

2.10 Illustration of distortions types involved in lens-induced distortion correction, including ”pincushion” and ”barrel” distortion [21]. 12

2.11 6DoF - Rotation and translation axes [Adapted from [23], with permission from Smart VR Lab]. 13

2.12 Overview of the process of human pose estimation system [24]. 14

3.1 BlazePose 33 keypoint topology as COCO (green) superset [28]. 17

3.2 Human pose estimation pipeline overview [28] [29]. 17

3.3 Vitruvian man aligned via two virtual keypoints predicted by BlazePose vs face detection bounding box [28][29]. 18

3.4 Tracking network architecture: regression with heatmap supervision[28][29]. 19

3.5 Attention mesh inference pipeline and the model architecture overview [31]. 20

3.6 Hand landmarks [27]. 21
3.7 Hand detection ML pipeline [32]. 21
3.8 Intel® RealSense™ D455 (left) and Microsoft Kinect XBOX 360 (right). 22
3.9 Intel® RealSense™ D455 (left) and Microsoft Kinect XBOX 360 (right) depth maps. . . . 23
3.10 Microsoft Kinect XBOX 360 depth convertion from gray scale to distance. 24
3.11 Mediapipe keypoints detection. 25
3.12 Orthogonal projection of the 3D view. 26
3.13 Representation of Shoulder Pitch calculation. 27
3.14 Representation of Shoulder Roll calculation. 27
3.15 Representation of Elbow Roll calculation. 28
3.16 Representation of Elbow Yaw calculation. 29
3.17 Right arm (a) and left arm (b) elbow yaw configutations. 30
3.18 Wrist Yaw calculation. 31
3.19 Failing cases for Wrist Yaw calculation. 31
3.20 Face keypoints for perspective-n-point solution. 32
3.21 Face orientation prediction. 33
3.22 PX2.0 VR Headset operating mode. 33
3.23 Blue color contour with center point (left) and respective blue color mask (right). 34
3.24 Head pitch calculation using the new method. 35
3.25 Head yaw calculation using the new method. 35

4.1 Coppelia simulation environment. 38
4.2 NAO robot. 38
4.3 Head joints and respective ranges. 39
4.4 Arm joints and respective ranges. 41
4.5 Finger links and respective ranges. 41
4.6 ROS rqt_graph with the three ROS nodes and the five ROS topics. 42
4.7 Model and ROS initialization. 43
4.8 Right arm subscriber callback. 44
4.9 Robot vision publishing. 44
4.10 Robot control movement test. 45
4.11 Object manipulation test. 46
4.12 Robot vision on Coppelia (a) and robot vision streamed in real time (b). 47

5.1 CHARMIE antropomorphic arm. 50
5.2 CHARMIE tendon system to control the fingers. 51
5.3 CHARMIE's arm circuit diagram. 52
5.4 CHARMIE's neck degrees of freedom. 53
5.5 Communication between devices on antropomorphic arm control test. 53
5.6 Object manipulation test [images from [40]]. 54
5.7 Communication between devices on robot control test. 55
5.8 Object manipulation test through VR. 57

List of Tables

3.1 Intel® RealSense™ D455 and Microsoft Kinect XBOX 360 comparison. 23

4.1 HeadPitch motion range according to the HeadYaw value. 40

5.1 Range of each anthropomorphic arm joint. 50

5.2 Range of each anthropomorphic hand finger. 51

Chapter 1

Introduction

Service robots are becoming increasingly popular for a variety of applications, including assistance, entertainment, and education. These robots are typically designed to interact with humans and to perform tasks on their behalf. One key aspect of service robots is their ability to be controlled remotely, either through a user interface or through gestures and movements.

This dissertation is included in the CHARMIE project (*Collaborative Home/Healthcare Assistant Robot by Minho Industrial Electronics*), a project developed by the University of Minho's Laboratory of Automation and Robotics. The primary objective is to create an intuitive interface that allows human users to effectively command the robot through teleoperation. To accomplish this, all sensory information from cameras to sensors are transmitted and displayed to the user in virtual reality.

The robot is controlled by a precise pose estimation and computer vision system, which is able to recognize the positions of the human body's joints and convert them into robot actions. In doing so, the robot is capable of performing various tasks such as unloading a shopping cart and placing the items on the appropriate shelf.

1.1 Motivation

Robotic teleoperation is clearly crucial when it comes to increasing workers safety and reducing logistics costs, because it avoids the need for the presence of humans in tasks and environments that may present health risks, access difficulties or adverse climatic conditions.

Robotic teleoperation through virtual reality has revolutionized this area even more, giving the operator the sensation of telepresence, where the operator has the feeling of being where the robot is. An example of robotic teleoperation using virtual reality is Reachy [1], a humanoid robot produced by Pollen Robotics, capable of being controlled by teleoperation using virtual reality.

Reachy is designed to be a versatile and easy-to-use platform for researchers, developers, and educators

in the field of robotics. It is equipped with a range of sensors, including cameras, microphones, and tactile sensors, which allows it to perceive its environment and interact with it. It also has a variety of actuators, such as motors and servos, which allows it to move and manipulate objects.
Michael Cussel, a virtual reality and tech enthusiast, was invited by Pollen Robotics to participate in an experiment with Reachy. This experiment involved using a VR headset to teleoperate Reachy through the internet, testing the capabilities of its teleoperation feature by controlling the robot's movements and actions while also receiving real-time video and sensor data from Reachy.
In order to test Reachy's skills, a variety of tests were performed, such as tic-tac-toe, sorting blocks, drawing and making a sandwich, as demonstrated in Figure 1.1.

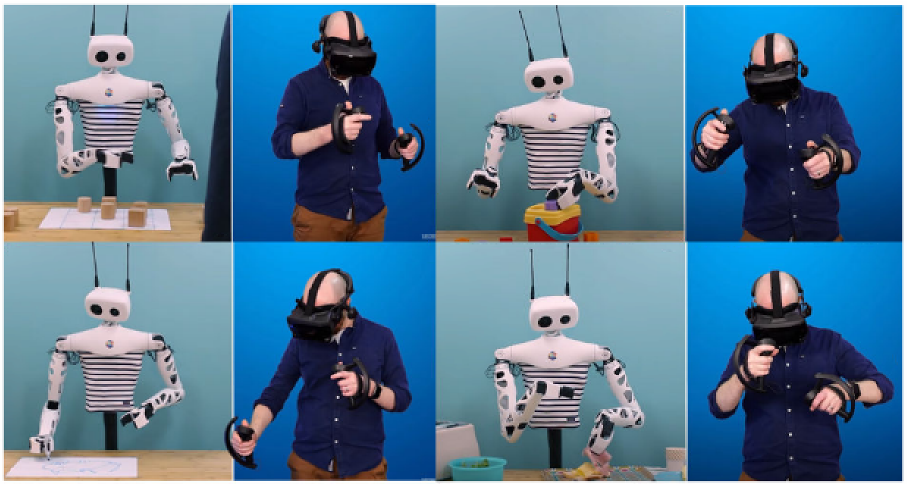


Figure 1.1: Reachy tests through VR [image from [2], with pemission from Michael Cussel].

The tests proved to be quite impressive, achieving a very acceptable latency for the distance at which the teleoperation was taking place, concluding that the development of virtual reality and augmented reality technologies enhance the teleoperation experience, providing operators with a more immersive and intuitive experience.

1.2 Objectives

The objectives of this dissertation consists of the following topics:

1. Introduction of concepts associated with the project

2. Research and development of algorithms for pose estimation

3. Creation of a virtual reality interface for human user interaction

4. Testing of the robot in simulated and real world controlled environments

5. Analysis of results and decisions of algorithms used

The research presented in this dissertation aims to provide a comprehensive understanding of the project by introducing the relevant theoretical concepts and developing advanced algorithms for human pose estimation. The first phase of the project involves researching and developing algorithms that accurately detect human movement and convert it into robot movement. The second phase focuses on designing and implementing a virtual reality interface that enables human users to access the robot's inputs and outputs, and view the robot's perspective in real-time. The robot is then tested in simulated environments to ensure proper functionality before being tested in real-world scenarios, performing simple tasks through teleoperation such as object manipulation. The goal of this phase is to gain a thorough understanding of the robot's behavior during teleoperated tasks and fine-tune the system to optimize its performance. The robot is then evaluated in real-world scenarios, where it performs simple operations through teleoperation. An in-depth analysis is conducted throughout the testing phase to evaluate the performance of the algorithms and identify areas for improvement.

1.3 Scientific Contributions

Laboratory of Automation and Robotics (LAR) is a Research and Development Laboratory of the University of Minho, focused on robotics projects of various types, from educational to competitive.
One of the projects currently under development is CHARMIE [3] (Figure 1.2), an assistive and service robot designed for domestic and healthcare settings, able to perform tasks in dynamic and non-standardized environments. Its ultimate objective is to provide aid in family homes, nursing homes, and healthcare facilities, and its scientific goal is to create a highly versatile robot that can perform a wide range of tasks by utilizing advanced machine learning techniques, allowing the robot to learn and improve its performance through observation and trial-and-error interactions with its surroundings.

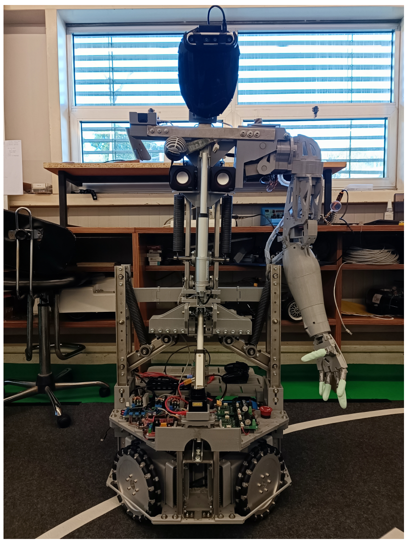


Figure 1.2: CHARMIE [4].

A key objective of the CHARMIE project is to participate in RoboCup@Home [5], a worldwide scientific challenge that aims to evaluate the capabilities of service robots in performing household tasks, as outlined in [6]. With its advanced design and capabilities, CHARMIE got qualified for the World Cup, scheduled to take place in July 2023. This is a significant milestone for the CHARMIE team, highlighting the tremendous progress that has been made in the development of this cutting-edge service robot.

1.4 Document Structure

This dissertation is structured into six chapters, starting with an Introduction that provides a brief overview and outlines the objectives and motivation for the research. The second chapter, State of the Art, reviews relevant literature on the topic and sets the context for the study. The third chapter, Algorithm Architecture: Pose Estimation Model, details the method used to address the problem at hand. The Simulation Framework chapter describes the simulation environment and the experiments conducted within it. The fifth chapter, Real World Tests, reports on the results of the experiments conducted on the real-world service robot CHARMIE. Finally, the Conclusions chapter summarizes the research findings, draws final conclusions and sugests future research.

Chapter 2

State of the Art

As previously discussed, the goal is the control of a service robot through teleoperation, using a pose estimation system complemented by a virtual reality system.
To understand the topic, an extensive literature review was developed, where four topics were approached:

1. Service robotics

2. Teleoperation and respective applications

3. Virtual Reality

4. Pose estimation algorithms

2.1 Service robotics

Due to the rapidly evolving landscape of the service robotics market, the term "service robot" does not yet have a strictly internationally accepted definition [7]. However, the International Federation of Robotics (IFR) defines service robots as semi or fully autonomous robots designed to provide useful services for human welfare, excluding manufacturing operations [8].
The range of applications for service robots is continuously expanding, from logistics to cleaning and even defense, such as disarming explosives. The IFR Statistics Department provides a set of reports each year containing data and graphs to track the dynamics of the service robotics market. Figures 2.1 and 2.2 show the number of units sold and the sales value for professional service robots, respectively, in 2018 and 2019, as well as potential developments for the period 2020 to 2023.

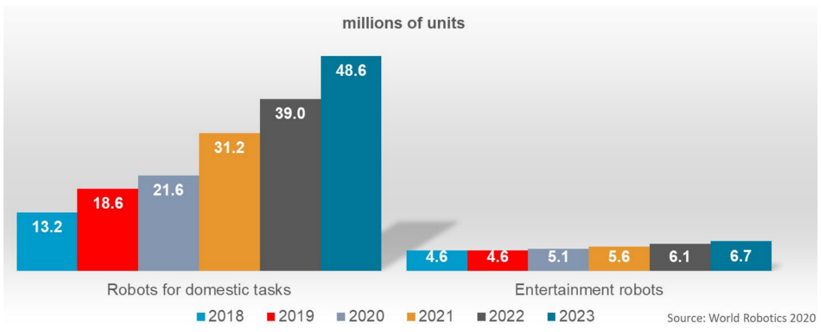


Figure 2.1: Service robots. Units sold in 2018 and 2019, potential development 2020-2023 [9].

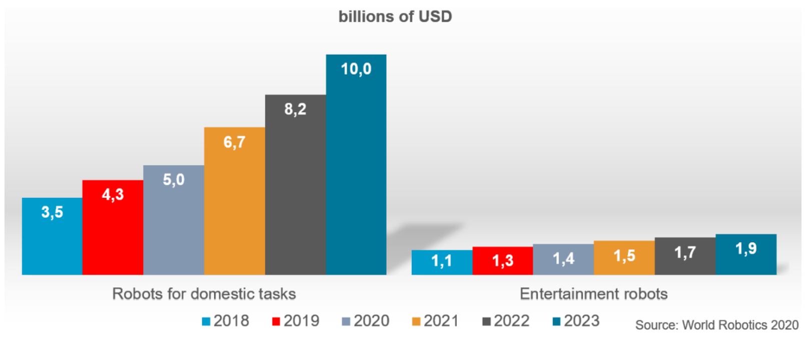


Figure 2.2: Service robots. Sales value in 2018 and 2019, potential development 2020-2023 [9].

Considering only the year 2020, it was possible to see a growth of 12% from a turnover of 6 to 6.7 billion dollars. Due to the pandemic situation, the most requested segment was professional cleaning service robots, having a 92% growth in terms of units sold and 51% in terms of turnover. In the same year, with 11% growth in turnover, were service robots built for cargo transport, accounting for one-third of sales [9]. Figure 2.3 shows the number of units sold in the 3 most requested sectors. In first place, logistics robots, responsible for automating the process of storage and transport of goods, and the main application of these robots is as AGV (Automated Guided Vehicles) in warehouses. Secondly, robots for public environments, usually used for public interactions in environments such as restaurants or hotels, and thirdly, robots applied to defense, such as disarming explosive material.

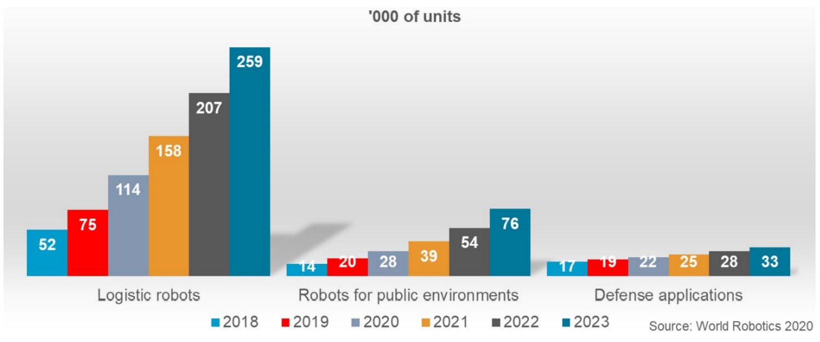


Figure 2.3: Service Robots. Units sold in top 3 applications in 2018 and 2019, potential development 2020-2023 [9].

2.2 Teleoperation

As mentioned earlier, some service robots perform increasingly complex and high-risk tasks. Therefore, there is a need for them not to perform autonomously, but controlled by humans through teleoperation.
The term teleoperation can be easily explained as "remote work", the terms "work" and "distance" being quite relative in this case, since it can be any kind of work and any distance [10]. Teleoperation can be understood as an extension of human perception and manipulation capability by coupling via communication means to artificial sensors and actuators. It commonly refers to a direct and continuous human control of the teleoperator [11].
A teleoperator is a machine that extends sensing and/or manipulation capabilities to a location distant from the person controlling it. It is typically equipped with environmental sensing sensors, artificial arms and hands, or other devices that allow mechanical work to be performed, and a communication system to and from the human operator [12].
According to the level of control exerced by the human, teleoperation can have three distinct classifications: direct, coordinated, and task-based [10].
In direct teleoperation, the teleoperator's actuators are analogously controlled by the operator and feedback is received by the operator in real-time. An example of direct teleoperation is a remote-controlled car. In coordinated teleoperation, the actuators are still controlled by the operator, but there is now a remote control loop on the teleoperator side that assists in closing the control loop, since the latter cannot be controlled by the operator due to existing delays. An example is a machine with a speed control set by the operator. In task-based teleoperation, most of the control lies with the teleoperator, who can perform tasks more or less autonomously and send information back to the operator through a Human-Machine Interface (HMI), where the operator only sends high-level commands. An example is a robot that can

perform a task autonomously, but the operator can monitor and make adjustments as needed.

2.2.1 Applications

The use of teleoperation to perform complex and high-risk tasks by service robots is a well-established practice. Teleoperation has been applied in several fields, including space exploration and medicine, which is later discussed in more depth. Additionally, teleoperation has also been used in other industries and applications.

Space Exploration

Having a physical human presence in space is difficult and expensive, making the application of teleoperation in this field crucial for its development. Teleoperation is already being used in various space-related activities, such as planetary exploration, satellite maintenance, and the assembly and maintenance of large space stations. One notable example of a teleoperated space robot is the Robonaut, a humanoid robot developed by NASA to assist humans in space exploration and tasks. The Robonaut uses a stereo display helmet (HMD - Head-Mounted Display), which provides live video from cameras embedded in the robot's head, allowing for intuitive operation and natural interaction with the work environment. This technology has been used in NASA's space exploration missions such as Robonaut 2 (R2) which was sent to International space station in 2011 to test its capabilities for space-related tasks [13]

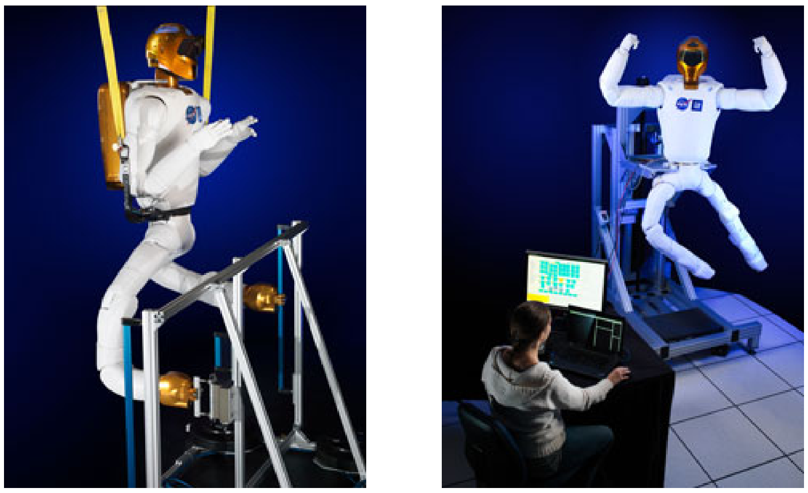


Figure 2.4: Robonaut, a humanoid robot designed and developed by NASA [13].

In the context of space exploration, the distance between the operator and the teleoperator is significant, as the teleoperator is located in space (Figure 2.6). Thus, tasks performed using teleoperation in this field require a high level of reliability, specifically a reliable communication connection with large enough bandwidth. This distance also results in communication delays, which can be addressed through the use of predictive models for task execution. [14].

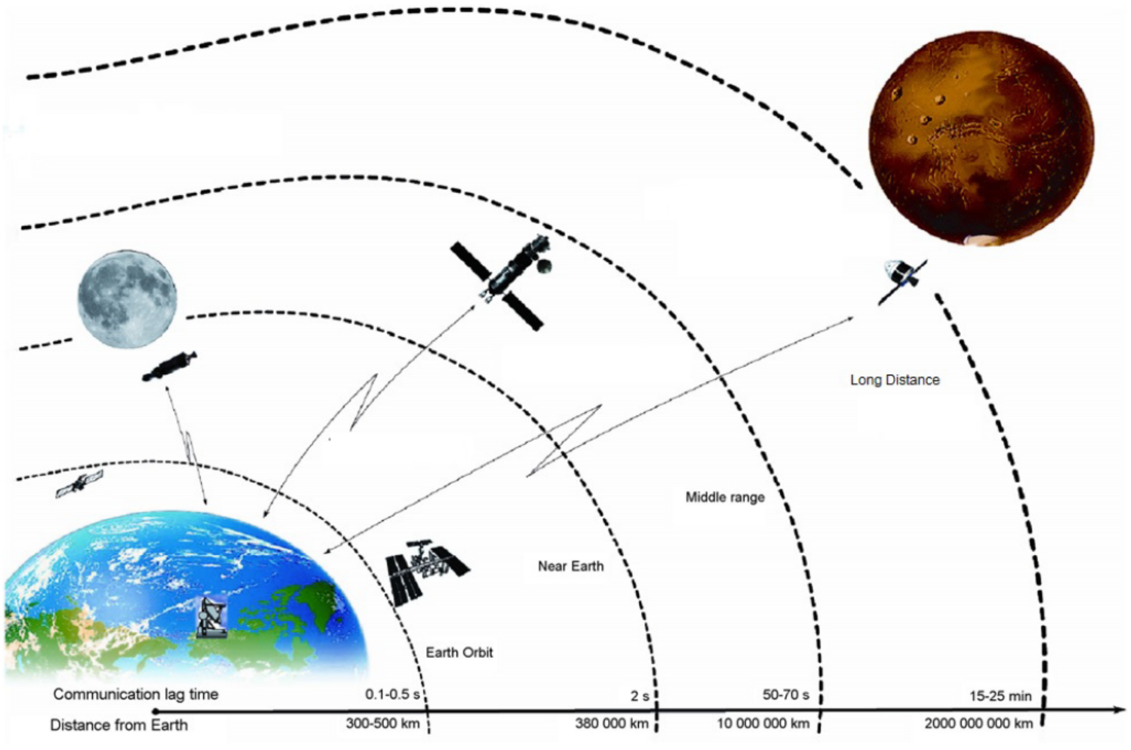


Figure 2.5: Communication delay in space [image from [15], with permission from Tamás Haidegger].

Medicine

Teleoperation is widely used in the medical field, from prostheses for patients with motor disabilities to Computer-Assisted Surgery, requiring the use of real-time 3D imaging technology to plan, execute and monitor surgical procedures. One of the most significant areas in which teleoperation has revolutionized is microsurgery. Microsurgery invlolves procedures that require a high level of precision, such as suturing blood vessels of 0.5mm in diameter. The precision required for such procedures is 50 to 100 μm, which is difficult to achieve with the human hand alone. However, with the help of teleoperation and robotic assistance, surgeons can achieve a precision of 10μm, which is necessary for surgical processes at the cellular level [16].

An example of a robot widely used in medicine in order to perform robotic telesurgery is the DaVinci robot, produced by the company Intuitive. This surgical system offers the surgeon an advanced set of instruments to use to perform surgery in a way that is as less invasive as possible. The surgeon's hand movements are translated in real time, and it also offers highly magnified 3D views of the surgical area [17].

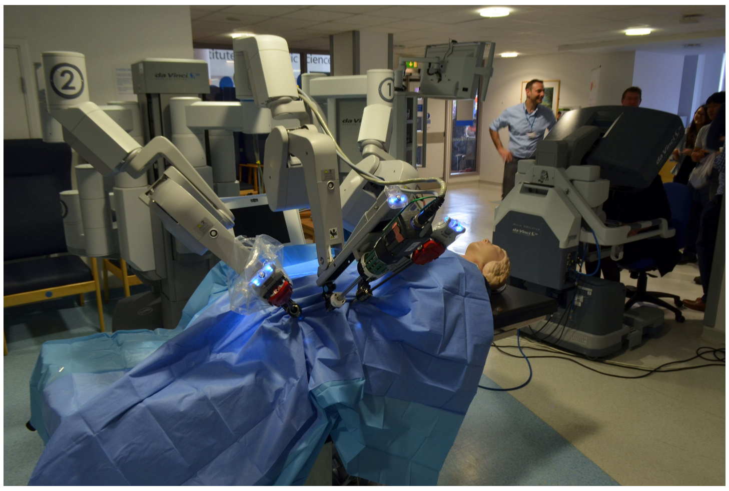


Figure 2.6: DaVinci surgical system [17].

2.3 Virtual Reality

One of the significant advancements in technology is the emergence of virtual reality, which has introduced a new way of interacting with users. Instead of interacting with a computer screen, users are immersed in the interface and their movements are captured to facilitate the human-machine interaction. [18]. Virtual reality consists of electronic simulations of experimental environments, allowing the user to interact in realistic three-dimensional situations [19]. This system typically includes a computer that can perform real-time animation, controlled by a set of wired gloves or other motion-sensing device, and a head-mounted display that provides the user with a stereoscopic visual output [20].

2.3.1 Head-Mounted Displays

Head-Mounted Displays (HMDs) are powerful pieces of hardware that enable users to immerse themselves in a virtual environment, simulating the user's vision with a slightly different image presented to each eye [21]. This creates a realistic field of view, as demonstrated in Figure 2.8, providing an enhanced user experience that cannot be achieved through traditional monitor displays.

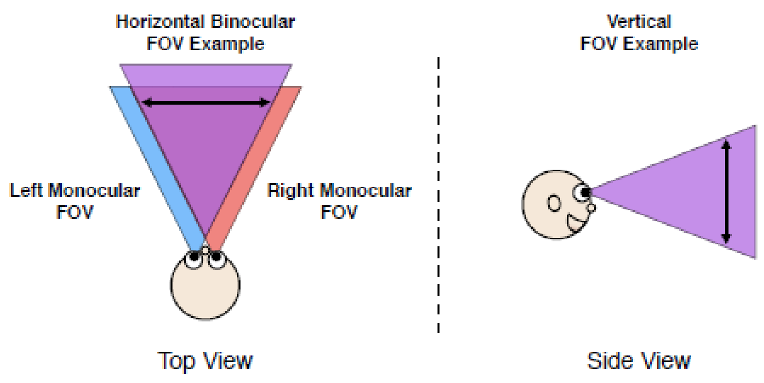


Figure 2.7: Depiction of the horizontal and vertical field of view [21].

As previously mentioned, the images presented in the virtual world should accurately reflect those in the real world for a more immersive experience. To achieve this, Head-Mounted Displays (HMDs) leverage the concept of stereoscopy, which is a form of binocular depth perception. This technique uses data from both eyes to create a more realistic sense of distance and depth, as the space between each eye provides a unique image to each [21].

An example of stereoscopy is the placement of a finger a short distance away from the face. When the finger is fixated upon with the left eye, while the right eye is kept closed, it can be seen in one position. However, when the reverse is done, fixating with the right eye and keeping the left eye closed, the position of the finger appears to change. To determine the distance at which the finger is located, triangulation is performed by the brain using the images provided by each eye.

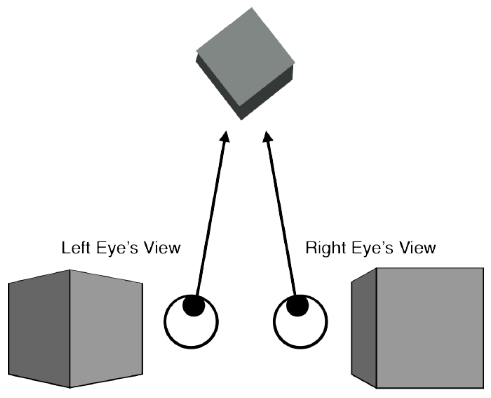


Figure 2.8: Illustration of binocular disparity, where each eye captures a unique angle of the object [21].

In Figure 2.9, it can be observed that each image captured by the eye has a different angle. If the correct simulation of the disparity between the eyes is achieved, stereoscopy can be performed almost naturally for the user of the HMD [21].

When it comes to the market for HMDs, the Oculus Quest 2 stands out as the most advanced technology currently available. It represents an upgrade from its predecessor, the Oculus Quest, and features an LCD (Liquid Crystal Display) panel with an impressive resolution of 1832x1920 pixels per eye, as well as a refresh rate of up to 120 Hz. [22]).

Figure 2.9: Oculus Quest 2.

The lenses of Oculus devices serve a dual purpose: to increase the field of view while eliminating any initial distortion. The lenses apply a ”pincushion” distortion, but then counteract it with a ”barrel” distortion to create the ideal image for the user. This process ensures that the magnification benefits are maintained while also neutralizing any lens distortion [21].

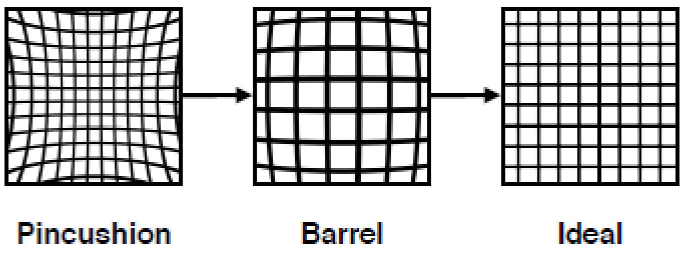


Figure 2.10: Illustration of distortions types involved in lens-induced distortion correction, including ”pincushion” and ”barrel” distortion [21].

The Oculus Quest 2 features a 6 DoF (Degrees of Freedom) system, tracking physical motion through the gyroscope and enabling virtual motion (Figure 2.12).

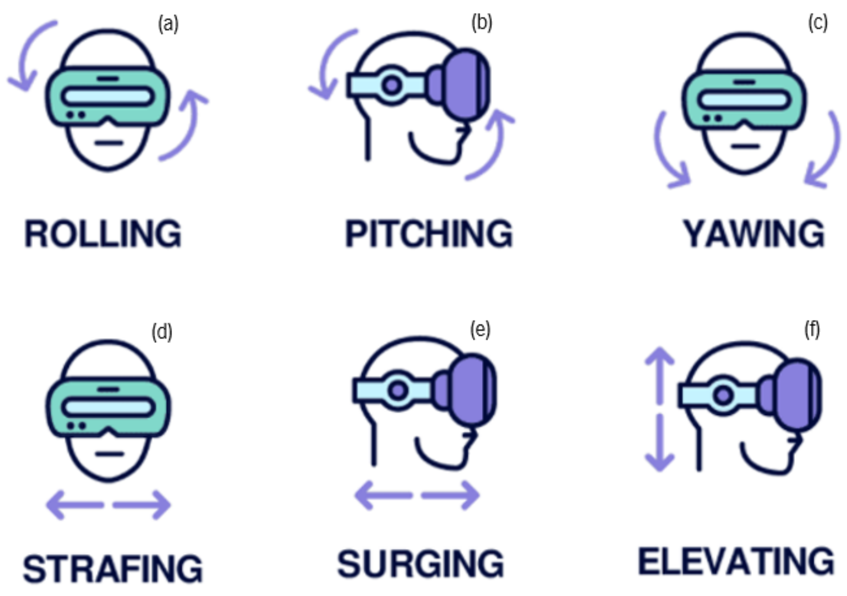


Figure 2.11: 6DoF - Rotation and translation axes [Adapted from [23], with permission from Smart VR Lab].

The movements are divided into two distinct axes: the rotation axis and the translation axis. The rotation axis, which is responsible for head movements, consists of three components: rolling on the x-axis (Figure 2.12 (a)), pitching on the y-axis (Figure 2.12 (b)), and yawing on the z-axis (Figure 2.12 (c)). The translation axis, on the other hand, is responsible for tracking the body's movement and consists of strafing on the x-axis (Figure 2.12 (d)), surging on the y-axis (Figure 2.12 (e)), and elevating on the z-axis (Figure 2.12 (f)).

2.4 Pose Estimation

The accurate estimation and tracking of human body in 3D is of crucial importance for a wide range of applications, particularly those that involve understanding human activity and movement. In the context of motion capture, it is common to use specialized clothing to more effectively capture the movement of the user. However, this approach can prove costly when the goal is to detect multiple individuals.

The pose estimation system makes it possible to determine the position of various joints on the human body. However, this detection can become challenging, as the estimation of the three-dimensional position and orientation is subject to factors such as lighting conditions or the clothing worn by the user. For example, difficulties may arise in detecting when the arms are resting against the body.

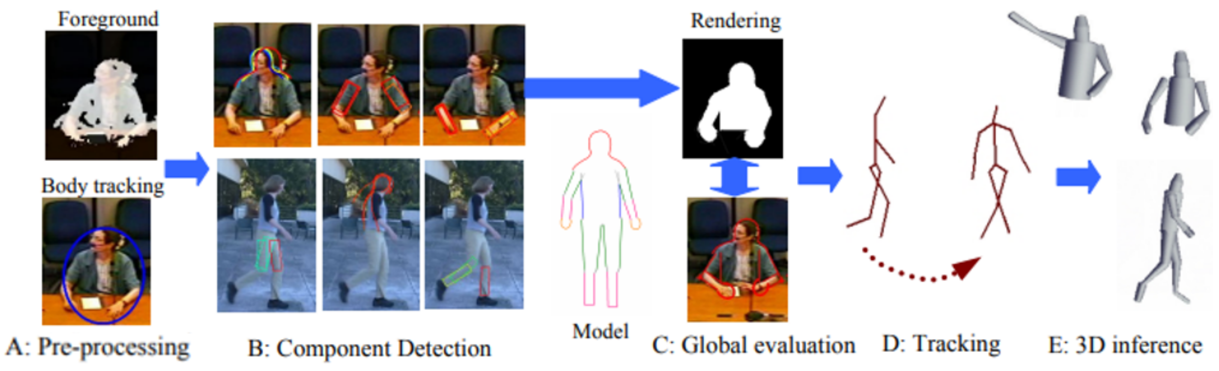


Figure 2.12: Overview of the process of human pose estimation system [24].

Lee and Nevatia [24] propose a hierarchical approach to estimate a set of features corresponding to the body outline. This comprises of a blob separation of large objects and objects that present a great contrast with the background. These blobs are tracked using ellipses to provide rough estimates of the position, size and orientation of the body (Figure 2.13 (A)). Subsequently, individual body features are detected in a hierarchical approach, with several options for each component, followed by an optimization process to eliminate the least likely options (Figure 2.13 (B)). Combining the different components then results in pose hypotheses (Figure 2.13 (C)) which are evaluated on a joint probability distribution from predictions of the body edges. This explains the effect of self-occlusion, which is the detection of hidden body parts, such as when arms are crossed. The optimal pose trajectory of the sequence is extracted using dynamic programming (Figure 2.13 (D)) and physical constraints of a 3D articulated human model are introduced to estimate the pose in three dimensions (Figure 2.13 (E)).

2.5 Conclusions

This chapter presents an overview of the current state of service robotics, teleoperation and virtual reality and how these concepts are relevant to the proposed pose estimation algorithm for controlling an anthropomorphic robot through mimicry and virtual reality. The literature review covers the various applications, technologies, and challenges associated with service robotics, teleoperation and virtual reality, providing a foundation for understanding the basic concepts addressed in this dissertation. The chapter also highlights the importance of the service robotics market and its growth, as well as the different types of teleoperation and their characteristics and the role of virtual reality in the system. Overall, the chapter provides a comprehensive overview of the research field and sets the stage for the development and implementation of the proposed algorithm.

In the following chapter, the development process of the pose estimation algorithm is thoroughly explained, including the specific framework chosen for its implementation. Detailed analysis of the calculation of the various angles of the robot joints is provided, as well as a comprehensive explanation of the method used for the development of the virtual reality interface for accessing the robot's vision.

Chapter 3

Algorithm Architecture: Pose Estimation Model

The scientific world has recently paid a lot of attention to the problem of estimating the human pose in three dimensions from an image or video due to the widening number of new applications, such as human-robot interaction, video games, and sports performance monitoring, being developed all the time. This has led to a multitude of studies in the area of pose estimation, however, there is still room for improvement.
Human pose estimation is a method that identifies how a person appears in an image. This estimation consists determining the location of each body part, such as facial points, hands and body joints, also known as key points. This chapter presents an elaborate algorithm for pose estimation and a description of the hardware/software used for its implementation.
This chapter presents the theoretical foundations and implementation of the pose estimation algorithm, as well as a comparison between two RGB-D cameras to determine the best choice. Furthermore, a detailed explanation of the algorithm is given, describing the method used to calculate the angles responsible for controlling the robot. Finally, the conclusions, difficulties encountered and solutions are documented.

3.1 Pose Estimation Framework: Mediapipe

As referred before, pose estimation is a computer vision task that involves the detection, connection and tracking of several keypoints. The proposed model for pose estimation makes use of Google's open-source software Mediapipe, which has the capability to detect hands, face and body pose simultaneously. In order to perform this detection, Mediapipe uses a holistic model pipeline, integrating separate models for pose, face and hands.

3.1.1 BlazePose Topology

COCO topologyis the current standard for human body pose. It consists of 17 keypoints located in the middle of the torso arms and legs. However, these keypoints do not include hands and feet, thus lacking scale and orientation information for these body parts. With BlazePose, these keypoints are extended to a total of 33 for a single person, using both heat maps and topology regression to acquire keypoint coordinates, predicting a landmark model. For face and hands tracking, a separate model is used. BlazeFace [25] [26] is a face detector specialized in detecting and tracking facial features such as eyes, nose and mouth. This model uses a combination of landmark and feature extraction techniques to detect the facial region and then track it across frames. Additionally, for hand tracking, BlazePalm [27] is used that combines heat maps, point-based and box-based techniques to accurately detect and track the hands.

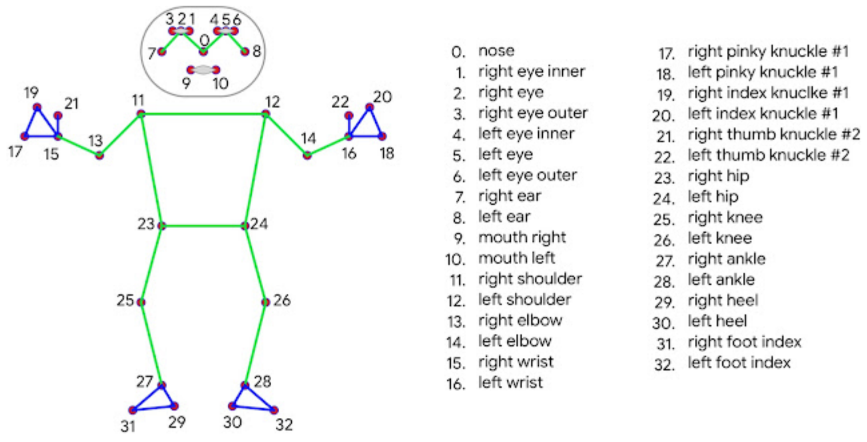


Figure 3.1: BlazePose 33 keypoint topology as COCO (green) superset [28].

3.1.2 BlazePose Pose Tracking

A two-step detector-tracker ML pipeline is used to estimate the pose, firstly identifying the pose region-of-interest (ROI) in the frame using a detector [29]. The tracker then identifies all the 33 landmarks from this ROI, using BlazePose's pose detector, designed to be extremely fast, with a processing time of only a few milliseconds per frame. In video use cases the detector only runs on the first frame, deriving ROI from the previous frame's pose keypoints [28].

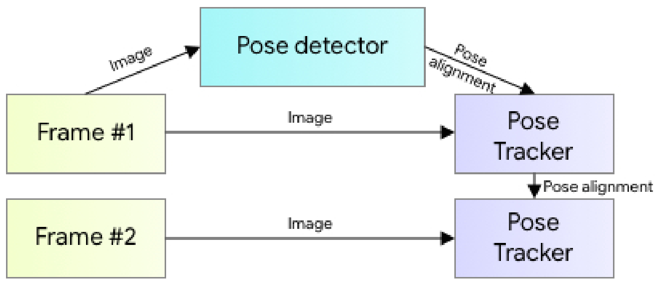


Figure 3.2: Human pose estimation pipeline overview [28] [29].

To achieve this remarkable speed the detector uses BlazeFace [25] [26], a face detection model, as a proxy for pose estimation. BlazeFace is used to detect the location of the person and to predict two additional virtual keypoints, which are the midpoints of the person's hips and shoulders. These two points are used to describe the human body's center, rotation, and scale, based on Leonardo da Vinci's Vitruvian Man concept [29].

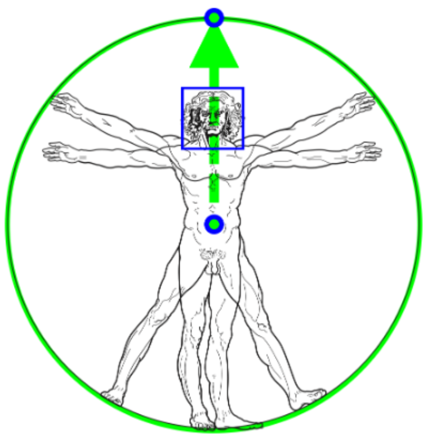


Figure 3.3: Vitruvian man aligned via two virtual keypoints predicted by BlazePose vs face detection bounding box [28][29].

3.1.3 BlazePose Tracking Model

The Mediapipe BlazePose is a system that can recognize and track people in videos and images. It uses a special dataset of pictures that have been labeled by humans, with either the whole person visible or just the hips and shoulders visible. The system was trained with this dataset and a technique called augmentation, a process of artificially increasing the amount of data by generating new data points from existing data. This technique helps in cases with heavy occlusions, using a neural network, which is a type of computer system modeled after the human brain, to analyze the data and keep track of the people.
The network is designed to locate 33 different body parts (Figure 3.4 right tree) and it uses a combination of different techniques like heatmap, offset and regression. The heatmap and offset loss are only used during the training stage (Figure 3.4 middle and left tree) and removed before the system is used in real-life (Figure 3.4 left tree), using only the regression encoder. The network also uses skip-connections and gradient-stopping connections to achieve balance between high-level and low-level features, improving the accuracy of the predictions.

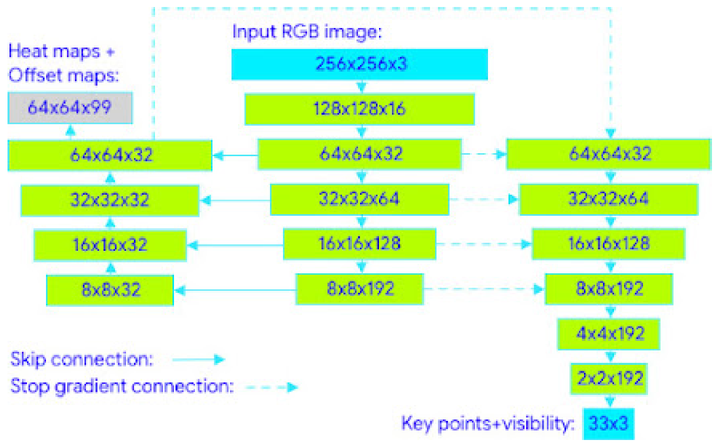


Figure 3.4: Tracking network architecture: regression with heatmap supervision[28][29].

3.1.4 Mediapipe FaceMesh

MediaPipe Face Mesh is a machine learning pipeline developed by Google [30] that uses a neural network to estimate the 3D facial geometry of a person in a video or image. The pipeline includes modules for detecting and tracking faces, estimating facial landmarks and generating a 3D mesh of the face.
Mediapipe FaceMesh is a machine learning (ML) pipeline that uses two deep neural network models to detect faces in an image and then predict the approximate 3D surface of the face. This pipeline can accurately crop the face from the image, reducing the need for data augmentation, allowing the ML model to focus more on predicting the 3D surface accurately. It also uses the face landmarks identified in the previous frame to generate the crops, and only if the model can no longer detect the face, the face detector is used to locate it again.
It uses different modules for face detection, these being:

1. Face detector: used to find faces in a video, using Mediapipe Face Detection [25][26].

2. Face landmark model: used to find specific points on the face, like the position of the eyes and lips. This model was trained using a special technique called transfer learning, which means it was trained on both synthetic (computer-generated) data and real-world data. The model can predict both the 3D position of the landmarks and the 2D position of the landmarks on the face. This model takes in a video frame as input and gives the positions of the landmarks as output, as well as the probability of a face being present and aligned.

3. Attention mesh module: similar to the face landmark model but applies attention to semantically meaningful face regions, thus predicting landmarks more accurately around lips, eyes and irises [31].

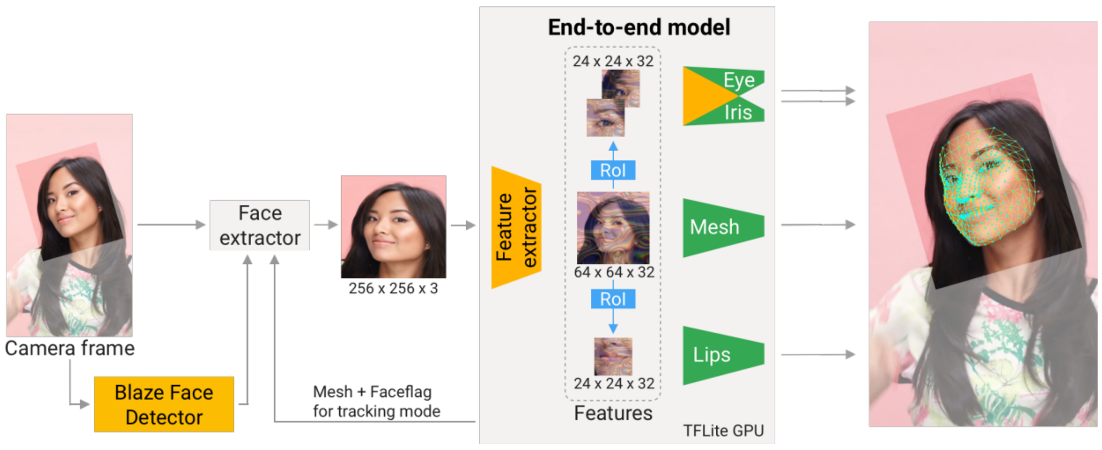


Figure 3.5: Attention mesh inference pipeline and the model architecture overview [31].

3.1.5 Mediapipe Hands

MediaPipe Hands [27][32] is a tool for detecting and understanding hand gestures in video. It uses computer vision and machine learning techniques to analyze the video and identify hand regions, as well as the position and movement of the fingers.

The tool can detect multiple hands in the same video and track their movement over time. It also provides information about the hand's landmarks, including the positions of the fingers and the palm, as well as the hand's orientation and scale.

MediaPipe Hands uses a combination of models to perform this task. First, it uses a hand detector model to find and locate hands in the video. The hand detector model is trained to recognize hand regions in different positions, orientations, and scales, based on a dataset of annotated images [32].

Once the hand regions are detected, MediaPipe Hands then uses a hand landmarks model to find the positions of the fingers and the palm within the hand region. This model is trained to predict the positions of the landmarks based on the hand's appearance and texture, generating a total of 21 landmarks, as depicted in Figure 3.6.

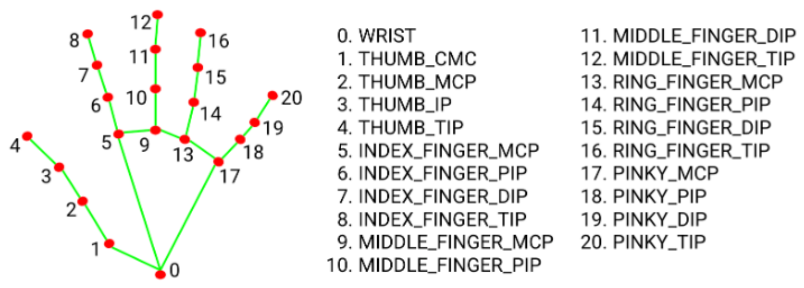


Figure 3.6: Hand landmarks [27].

It also uses a palm detector model that can detect the position, orientation, and 3D size of the hand's palm. This information can be used to understand the hand's gesture, such as the hand's position, movement and grasping.

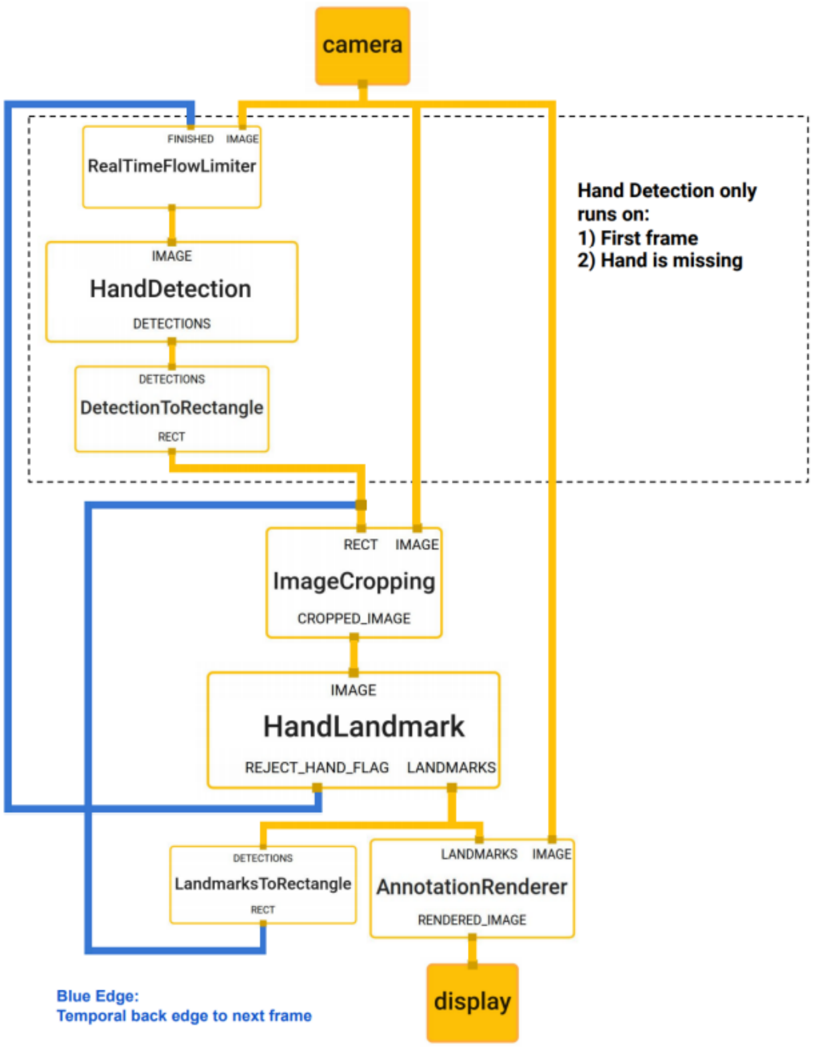


Figure 3.7: Hand detection ML pipeline [32].

3.2 2D to 3D

As referred in 5.1, Mediapipe predicts a landmark model which consists of a total of 33 keypoints, each one with three degrees of freedom (x, y and visibility). However, this model only allows a 2-dimensional prediction, which is not enough to measure all the operator's movements with great accuracy. To upgrade this model, it is necessary to add a new degree of freedom to the keypoints in order to provide a more precise prediction of the operator's pose.
To solve this issue, an RGB-D camera is needed, giving the keypoints a new degree of freedom: z or depth. A comparative study was carried out between two RGB-D cameras, being the Intel® RealSense™ D455 [33] and Microsoft Kinect XBOX 360 [34], two cameras already acquired by the LAR lab, in order to compare the depth mapping in each of them. Specifically, the study assessed the accuracy and precision of the depth mapping in each camera, as well as the resolution of the data produced.

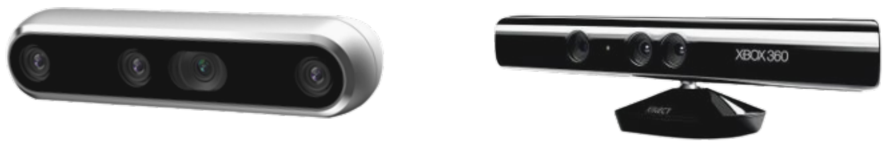


Figure 3.8: Intel® RealSense™ D455 (left) and Microsoft Kinect XBOX 360 (right).

A wide number of applications have benefited from the usage of the third dimension. Among the various applications already existing, pose estimation is one of them. The ability to know where an object is in a 3-dimensional space can improve the existing 2-dimensional algorithms because it allows to upgrade from frame to space object position.
Intel® RealSense™ D455 is a stereo vision depth camera with a vision processor, RGB sensor and stereo depth module, designed to be compact for easy and portable setup. When performing tasks that require great accuracy, such as object manipulation, 3D data is more reliable once it is less susceptible to light change conditions. As Intel® RealSense™ D455, Microsoft Kinect Xbox 360 also has this motion sensing technology, allowing users to interact with the console using only body motion. This technology creates an immersive experience, once there is no need for a controller. A comparison between these two cameras technologies is shown in Table 3.1.

Table 3.1: Intel® RealSense™ D455 and Microsoft Kinect XBOX 360 comparison.

	Intel® RealSense™ D455	**Microsoft Kinect XBOX 360**
Dimensions	124 x 26 x 29 mm	249 x 66 x 67 mm
Recommended Range	40 cm to 6 m	1.2 m to 3.5m
Minimum depth	40 cm	50 cm
Maximum depth	20 m	5 m
Accuracy range	Up to 1%	Up to 4%
Depth resolution	Up to 1280x800	640x480
Depth frame rate	Up to 90 fps	30 fps
Depth FOV	86° x 57° (± 3)	43° x 57° (± 27)
RGB resolution	Up to 1280x800	640x480
RGB frame rate	Up to 90 fps	30 fps
RGB FOV	86° x 57° (± 3)	43° x 57° (± 27)

The Intel® RealSense™ D455 and the Kinect XBOX 360 are both RGB-D cameras, but they use different technologies to calculate depth. The Intel® RealSense™ D455 uses a stereo-camera system, consisting of two cameras that use triangulation to measure depth. This process involves calculating the angle and distance between the two cameras in order to determine object's depth. The D455 also has infrared emitter and receiver technology to measure depth, enhancing precision in scenes with low-light conditions. On the other hand, Kinect XBOX 360 uses an infrared emitter and receiver to calculate depth. The device emits an infrared light and captures its reflexion with a CMOS (complementary metal oxide semiconductor) sensor, using this data to calculate the distance to an objects in the scene.
A comparison between both depth maps can be viewed in Figure 3.9, where each pixel on Intel depth map is color-coded to denote the corresponding depth, while Kinect depth map uses a grayscale to indicate the relative proximity to the camera, with darker colors indicating greater distances.

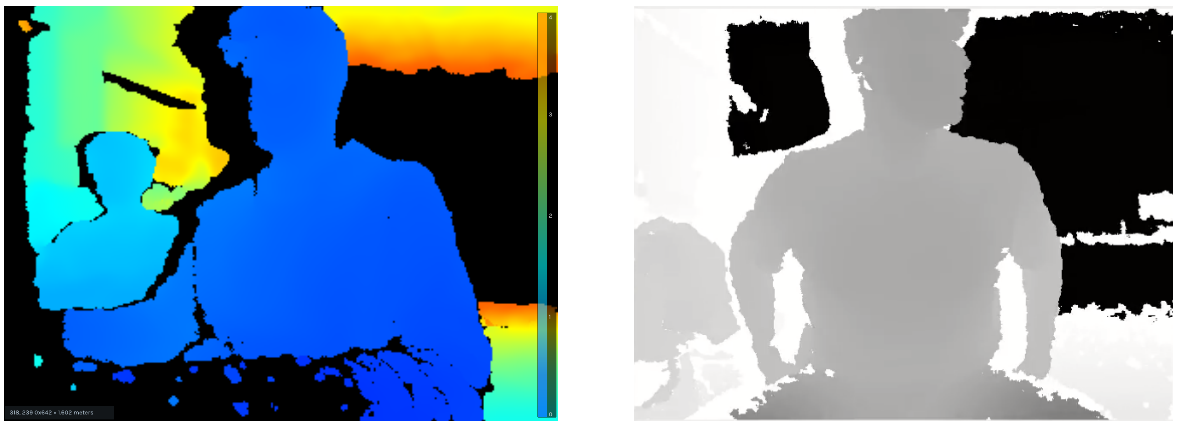


Figure 3.9: Intel® RealSense™ D455 (left) and Microsoft Kinect XBOX 360 (right) depth maps.

The Intel depth technology provides a function called *get_distance()*, which provides an accurate reading of distance in meters. This is a far more efficient solution than the process required for Kinect, which entails a conversion of gray values to depth, by measuring the distance at various points in order to create a graph and its associated function, as depicted in Figure 3.10.

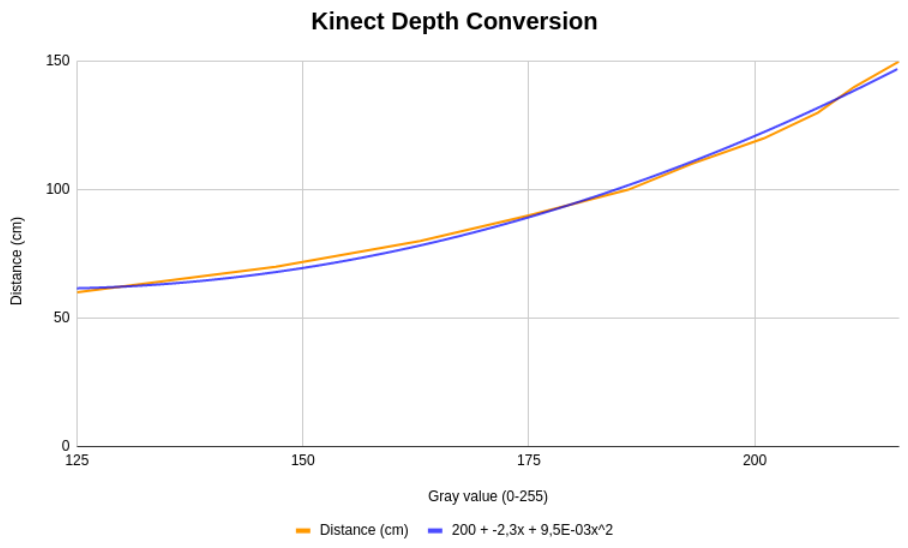


Figure 3.10: Microsoft Kinect XBOX 360 depth convertion from gray scale to distance.

After comparing both cameras, the Intel® RealSense™ D455 has superior specifications, since it offers better accuray range, which is a very important aspect to measure depth. Additionally, Intel's camera provides superior image and depth resoluton, making this camera a more desirable option over Microsoft Kinect Xbox 360.

3.3 Body Pose Estimation by Image Processing

In order to accurately calculate all the joints movements and determine the values for all degrees of freedom, a Python algorithm was developed. First, it is necessary to ensure that the depth camera is properly calibrated. A clipping distance was set to make sure that any depth values beyond the clipping value are ignored, ensuring that the depth map only captures the operator. Next, the color and depth frames must be aligned in order to achieve accurate pose estimation. This is a critical step, as misalignments can lead to miscalculations, making pose estimation unreliable. Once the camera is calibrated, Mediapipe can be used to perform pose estimation on the color frame. As outlined in the previous chapter, Mediapipe identifies several keypoints of the person's body and returns their x and y positions on the frame.

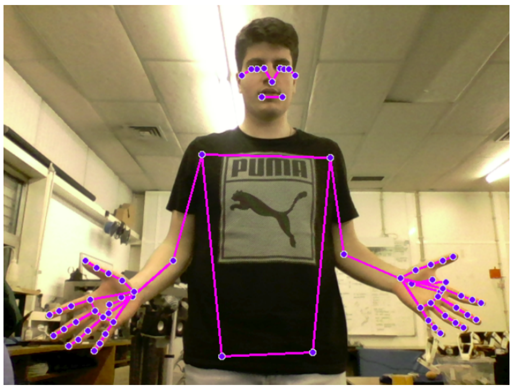


Figure 3.11: Mediapipe keypoints detection.

The third dimension is then added to locate the keypoints in the 3D space. To accomplish this, the function provided by the Realsense Python library, *get_distance()*, is used, taking a frame position as input and returning the depth at that point, resulting in a 3D model approximation containing all the body keypoints, since it's impossible to have a full 3D map using only one camera.
Using this 3D model, it is possible to calculate all the joints angles with higher accuracy. The chosen method to calculate them was the scalar product, denoted by the following formula:

$$\theta = \cos^{-1}\left[\frac{A \cdot B}{||A||\;||B||}\right] \tag{3.1}$$

A and B are two vectors that share a common point from which the angle θ between them is measured. The scalar product of the two vectors is represented by $A \cdot B$ and the norm of each one is denoted by $||A||$ and $||B||$, respectively.
Incorporating depth information into pose estimation calculations can greatly improve the accuracy and precision of the results. However, this added complexity can also introduce a problem known as joint overlap, which can lead to inaccurate estimates. To address this issue, a rotatable array system can be used to store joint values. By rotating the array and averaging the stored values, any overlap is accounted for and the joints depth can be accurately estimated.
While the inclusion of a third dimension can greatly enhance the precision and accuracy of calculating joint angles, certain situations may call for a two-dimensional approach. One such example is the calculation of Shoulder Pitch. By using the 3D coordinates of the points, it is possible to decompose them into distinct planes for analysis - namely, the XY plane (frontal view), the YZ plane (lateral view), and the XZ plane (top view), making an orthogonal projection of the 3D view, as represented in Figure 3.12. By focusing on only the necessary planes for analysis, the calculation process can become more efficient and streamlined.

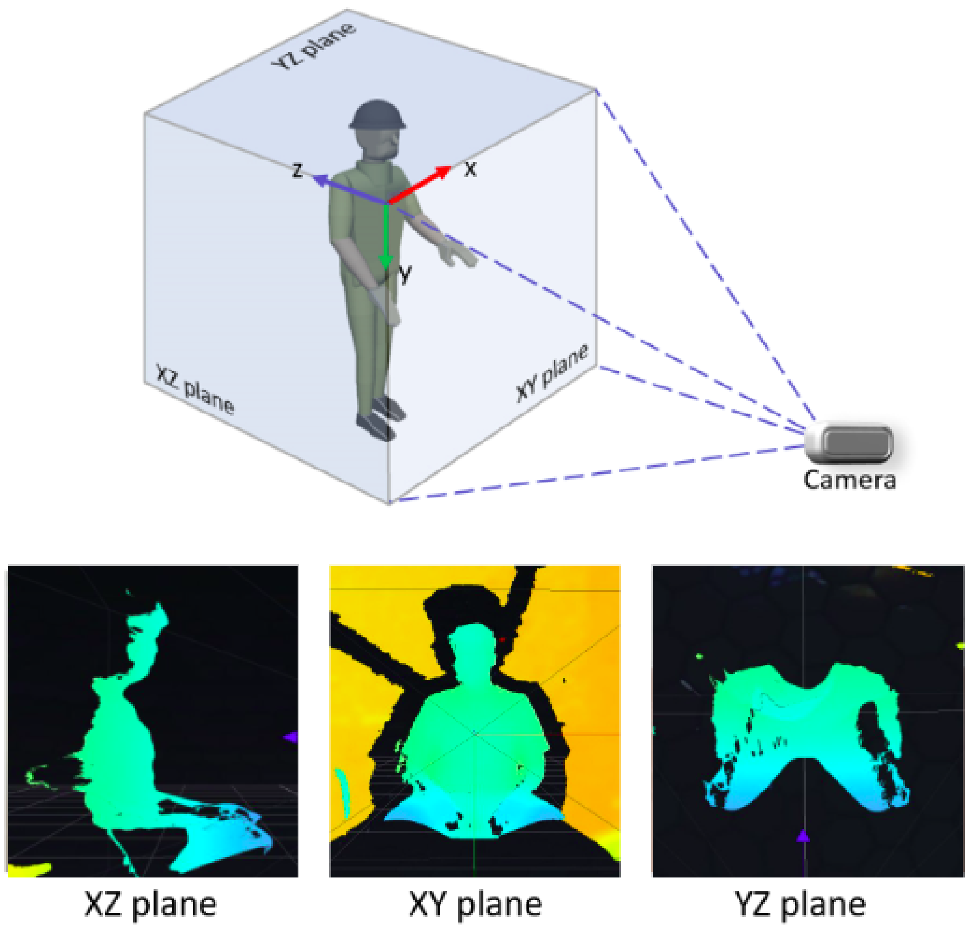


Figure 3.12: Orthogonal projection of the 3D view.

3.3.1 Shoulder Pitch Calculation

Shoulder Pitch is the shoulder movement along the Y-axis, responsible for the forward-backward motion. In order to calculate this degree of freedom, the three reference points for the scalar product application are the hip, the shoulder and the elbow. Since it is a foward-backward movement, only the x and z coordinates of the points are used, corresponding to the lateral plane, as shown in Figure 3.13.

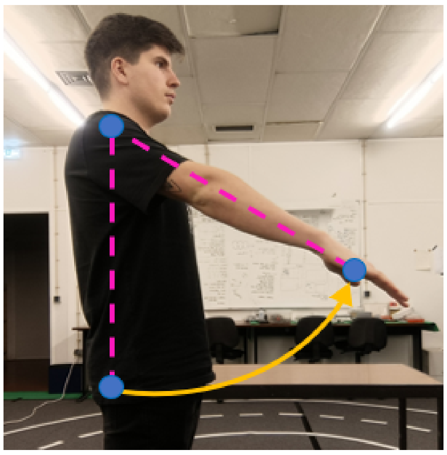


Figure 3.13: Representation of Shoulder Pitch calculation.

3.3.2 Shoulder Roll Calculation

Like Shoulder Pitch, Shoulder Roll is calculated using the same three points: hip, shoulder and elbow. Since Shoulder Pitch is a lateral movement, only the x and y coordinates of the points are used, corresponding to the frontal plane, as represented in Figure 3.14.

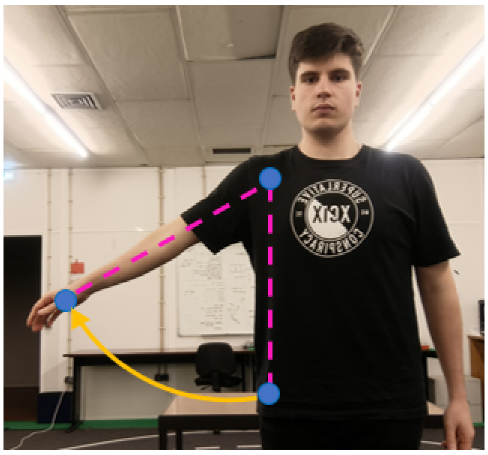


Figure 3.14: Representation of Shoulder Roll calculation.

One of the limitations in the Shoulder Roll calculation is when the elbow stands in front of the shoulder. In this situation, the angle is miscalculated, resulting in an erroneous value for ShoulderRoll. In order to accurately measure Shoulder Roll in this case, a new method was devised. This method takes into account the difference between elbow and shoulder depths, using this value to modify the Shoulder Roll measurement..

$$ShoulderRoll = ShoulderRoll * \left(1 - \frac{1}{20} * (ShoulderDepth - ElbowDepth)\right) \qquad (3.2)$$

This is achieved by multiplying the Shoulder Roll value by the function in parentheses, which varies between 0 and 1 depending on the difference between the depths. When the difference between the depths is maximum (20cm), the ShoulderRoll value is multiplied by 0, while when it is minimum (0cm) the ShoulderRoll value is multiplied by 1.

3.3.3 Elbow Roll Calculation

Elbow Roll is the degree of freedom responsible for the open-closed movement of the elbow. As with the previous degrees of freedom, the scalar product was used to calculate the angle using the shoulder, elbow, and wrist points, as shown in Figure 3.15.

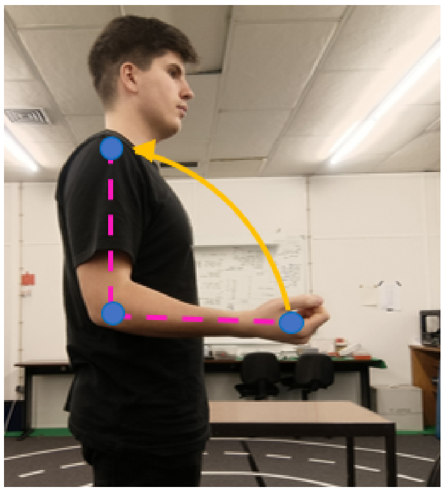


Figure 3.15: Representation of Elbow Roll calculation.

3.3.4 Elbow Yaw Calculation

Elbow Yaw is the degree of freedom responsible for the rotational motion of the shoulder joint. Calculating these movements can be a challenge, as precise calculation can be difficult to obtain if relying solely on image processing. To calculate Elbow Yaw, three points are used in the scalar product: the wrist, the elbow and an imaginary point containing the x-coordinate of the elbow and the z-coordinate of the wrist, as illustrated in Figure 3.16.

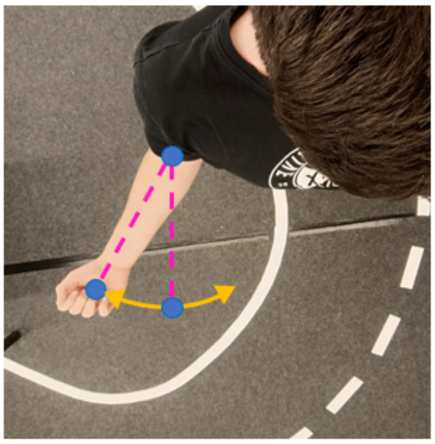
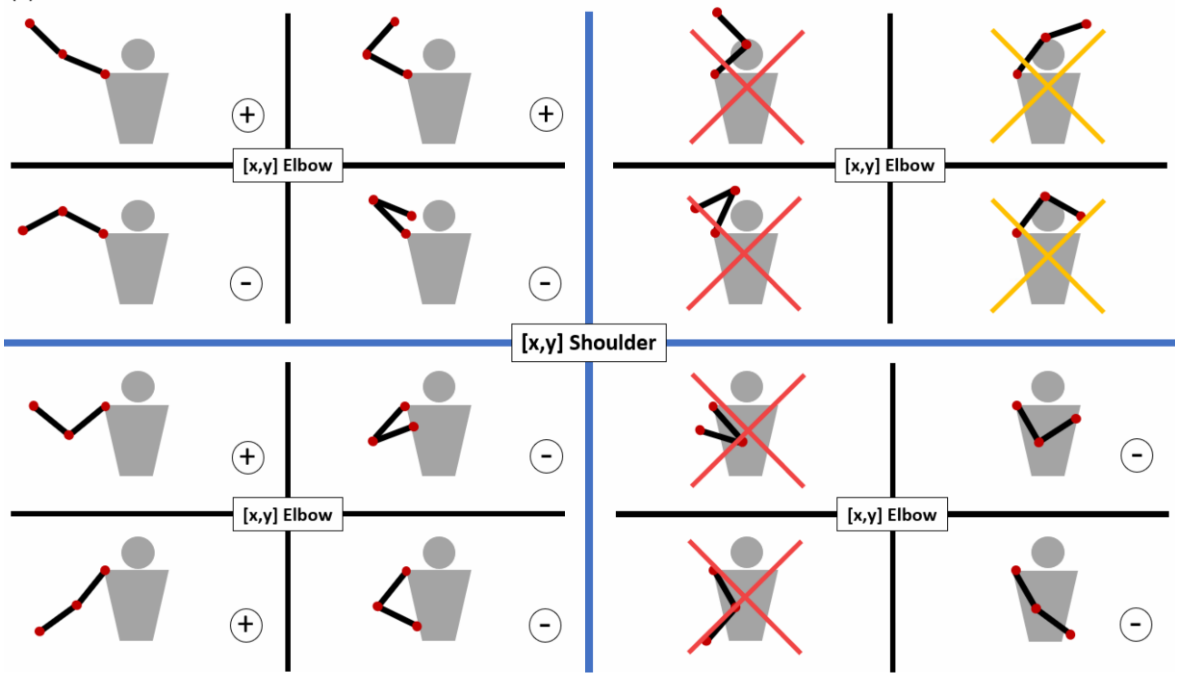


Figure 3.16: Representation of Elbow Yaw calculation.

A comprehensive study was conducted to analyze all possible cases of torsion, where each one corresponds to either outward (positive) or inward (negative) torsion. X and y coordinates of the shoulder were taken into consideration and 4 possible cases for locating the x and y coordinates of the elbow were identified. Additionally, within each of these cases, 4 possible cases for locating the x and y coordinates of the wrist in relation to the elbow were established, totaling 16 cases.

Figure 3.17 illustrates all ElbowYaw configurations, with (a) representing the right arm and (b) representing the left arm. Red crosses identify physically impossible positions, while yellow crosses indicate positions that are physically possible, but not achievable by the robot due to limited joint range of motion.

(a)



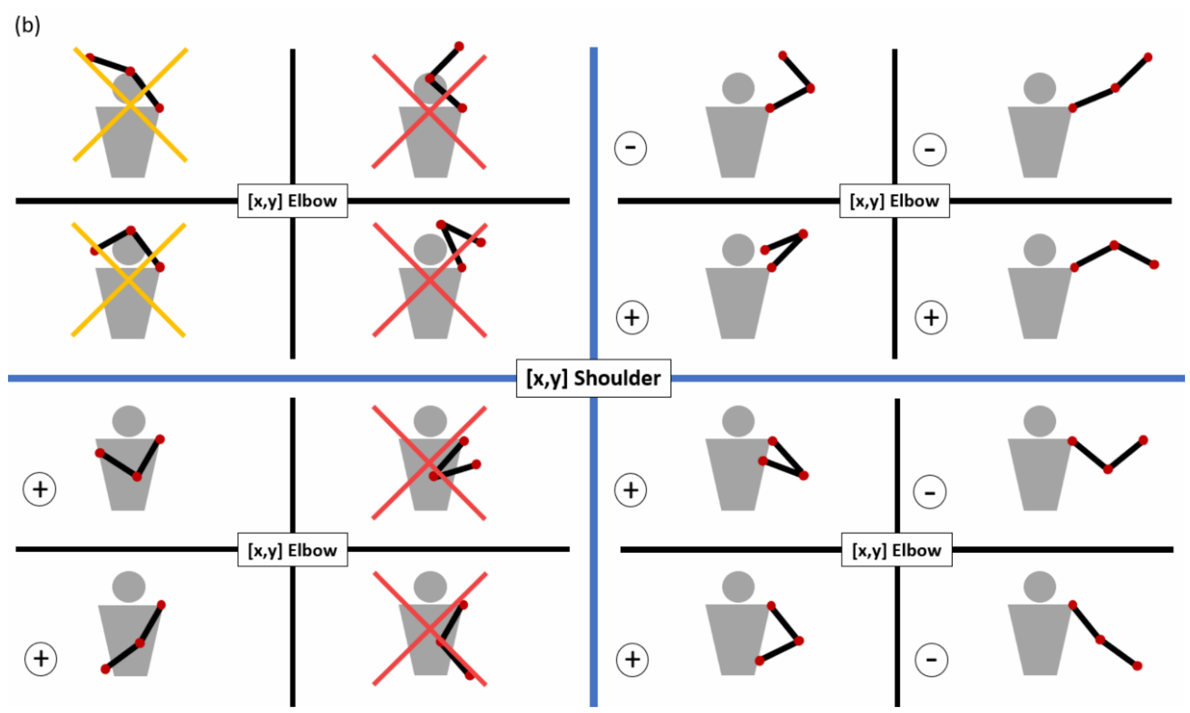


Figure 3.17: Right arm (a) and left arm (b) elbow yaw configutations.

3.3.5 Wrist Yaw

Wrist yaw is the degree of freedom that allows wrist rotation. Calculating rotations using only image processing can be difficult, so a method was employed that determines the slope of two lines, both of which have the wrist as a common point. The first line has the metacarpophalangeal joint (MCP) of the index finger as its second point, while the second line has the MCP of the pinky as its second point. First the slope of each of the lines is calculated using the following mathematical expression.

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{3.3}$$

Where *m* is the slope of the line formed by the points (x_1, y_1) and (x_2, y_2).
After calculating both lines slopes, the angle made between them can be determined using the inverse tangent of the slopes.

$$\theta = tan^{-1}\left[\frac{m_1 - m_2}{1 + m_1.m_2}\right] \tag{3.4}$$

Where θ is the angle between the two slopes m_1 and m_2.

Using both of the previously presented equations, it is possible to determine the angle that the wrist makes, as illustrated in Figure 3.18.

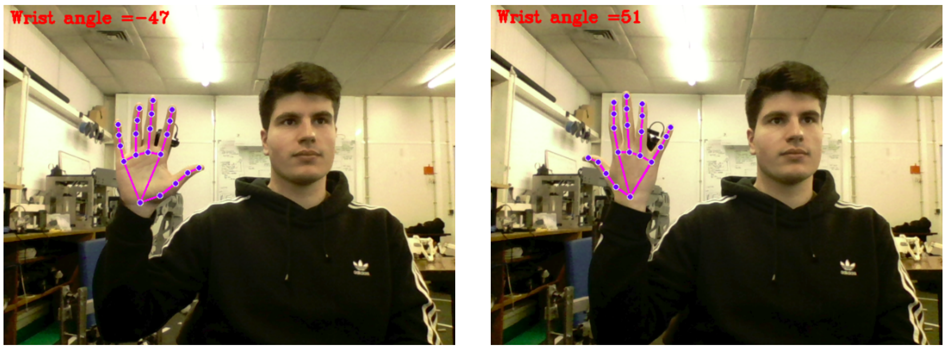


Figure 3.18: Wrist Yaw calculation.

Although this method has been found to be effective in some cases, it is not always reliable. When the arm points in the camera direction, i.e. the palm is not visible, it is not possible to accurately calculate wrist rotation. Figure 3.19 shows 3 cases where the arm is in three different positions. In all three cases it can be seen that there was no rotation of the wrist. However, the values given for the slope in each of the cases are different, meaning that this method is not suitable for wrist rotation calculation.

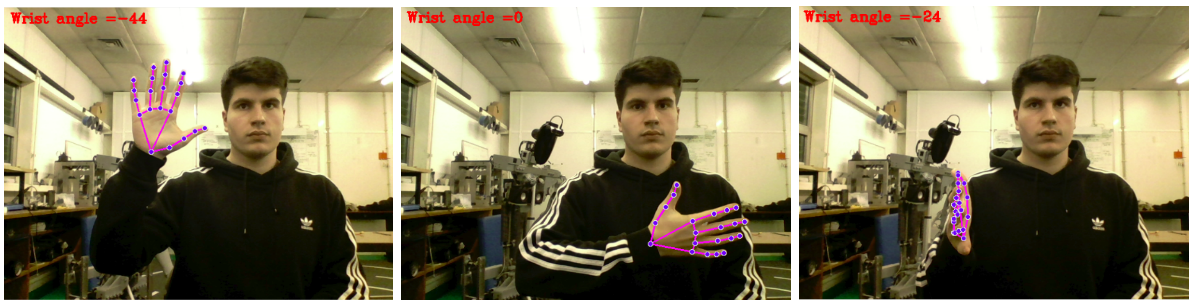


Figure 3.19: Failing cases for Wrist Yaw calculation.

3.3.6 Hand Opening and Closing

One of the key advantages of teleoperation is the ability to handle objects. This control has two distinct modes, open an close, while individual finger control is not possible. To detect the opening and closing of the hand, the wrist and middle tip finger points are used. Using a 3D vector-based approach using the Euclidean distance formula, the distance between these two points can be measured.

$$dist = \sqrt{(Wrist_x - Middle\_tip_x)^2 + (Wrist_y - Middle\_tip_y)^2 + (Wrist_z - Middle\_tip_z)^2} \tag{3.5}$$

3.3.7 Head Pitch and Head Yaw

The joints responsible for head movement, known as head pitch and head yaw, are calculated differently from the other joints in the body. Head pitch allows for a twisting motion, while head yaw allows for a forward-backward motion. To perform this calculation, the Mediapipe library and OpenCV are used. The MediaPipe library is used for face detection and facial landmark estimation, while OpenCV is used for the solution of the perspective-n-point [35] problem and the decomposition of a rotational matrix. The perspective-n-point solution is based on keypoints of the face, which are illustrated in Figure 3.20.

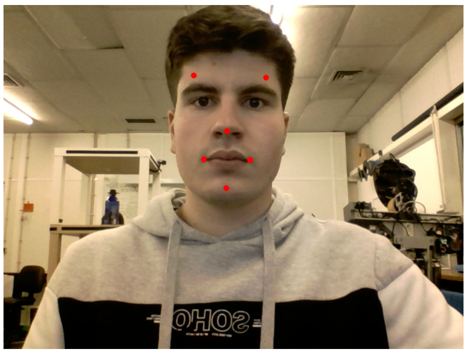


Figure 3.20: Face keypoints for perspective-n-point solution.

These keypoints are detected and estimated using MediaPipe and the face orientation in the image is determined using the PnP (Perspective-n-Point) method with OpenCV. The PnP method is used to determine the head orientation from the face 3D model and the 2D coordinates of the keypoints by estimating the transformation of the 3D scene to the 2D image. Using the camera matrix, 2D and 3D point matrices and the assumed zero distortion matrix, a system of equations can be solved to calculate the rotation matrix via singular value decomposition (SVD).

As depicted in Figure 3.21, a blue line is drawn connecting nose point and the predicted point, allowing to get the pitch and yaw angles.

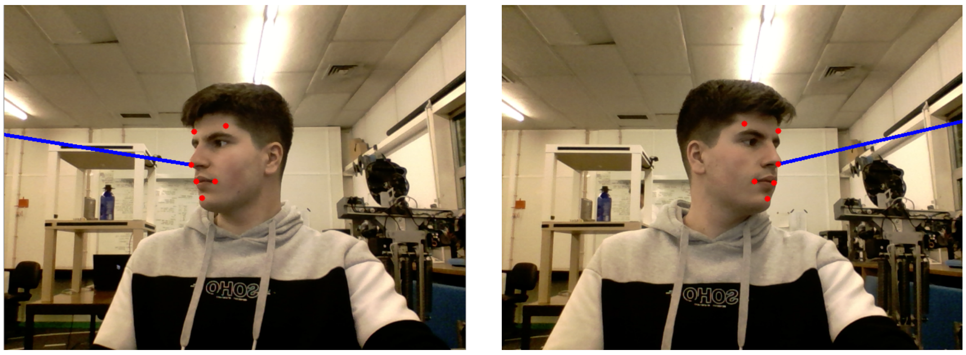


Figure 3.21: Face orientation prediction.

3.4 Virtual reality

By leveraging the power of telepresence technology, people can now control and communicate with a robotic device situated in a remote location. This technology offers an incredibly immersive experience, thanks to virtual reality (VR) equipment, allowing users to feel like if they were truly present in the location they are interacting with. Additionally, this allows for real-time collaboration between individuals, no matter the physical distance, offering unprecedented levels of convenience and connectivity.

The PX2.0 virtual reality headset offers a very good virtual reality experience at an affordable price. With its adjustable lenses, it allows users to transform two distinct images displayed on a phone into an immersive virtual reality experience, simply by inserting the phone into the headset's housing. Figure 3.22 shows the functioning of the virtual reality headset, with a phone displaying two distinct images, one for each eye. These images are processed in OpenCV and then sent to a web framework, so they can be accessed through the phone.

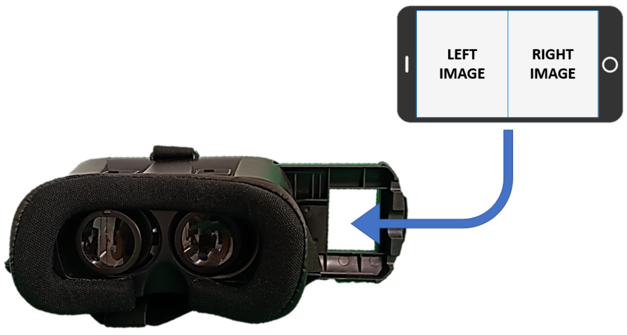


Figure 3.22: PX2.0 VR Headset operating mode.

One problem arose when using the VR headset. When these were placed on the operator's head, the face could not be detected, making the solution for head pose estimation unfeasible. To solve this problem, a new method for calculating head joints was developed. This method consisted of placing a surface with a detectable color in front of the headset. Using a filter with an HSV mask, it was possible to detect only the desired color through a mask. This mask was then processed in order to find the contours and calculate the center, this point being approximately the nose. Figure 3.23 represents the mask extraction process and the mask contour, with the center of the contour represented by a white dot.

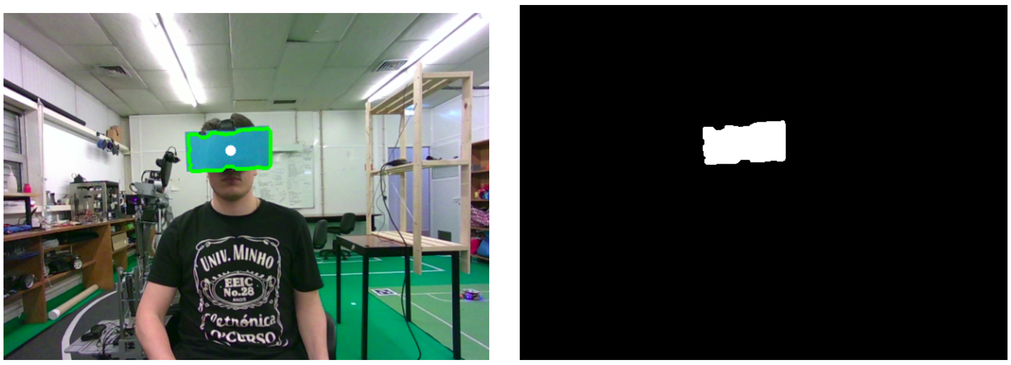


Figure 3.23: Blue color contour with center point (left) and respective blue color mask (right).

Once this point was detected, the depth was calculated to obtain a three-dimensional coordinate. To calculate head pitch, two additional points were created - one at the center of the shoulders and the other at a distance of half the length of the shoulders and with the same depth, perpendicular to the first point. These three points - the center of the mask and the two created points - were used to calculate the head pitch. Figure 3.24 illustrates the position of the three points previously mentioned, as well as a clear representation of how the angle is calculated.

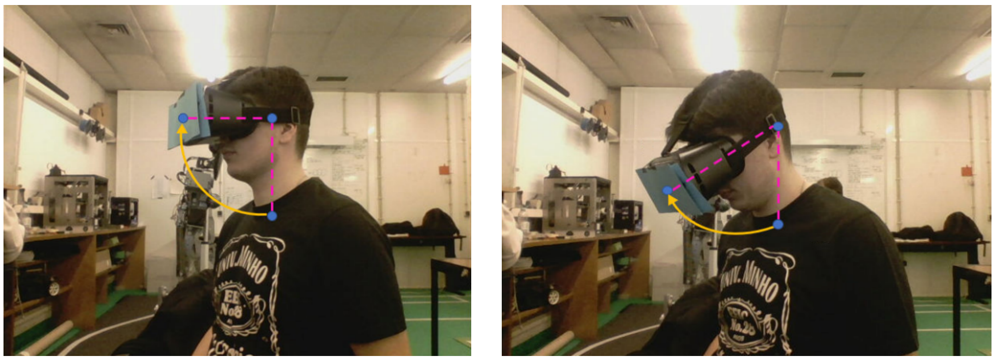


Figure 3.24: Head pitch calculation using the new method.

For the calculation of the Head yaw angle, the method used was similar to that of the Elbow yaw. A point perpendicular to the shoulder line (described previously) was used as the first point. A second point is then created, with the same x and y coordinates, but with a lower depth. The angle between these two points, in conjunction with the mask center point, was then calculated to determine the Head yaw angle, using only the top view (XZ plane). The position of the three points and the angle calculation are illustrated in Figure 3.25.

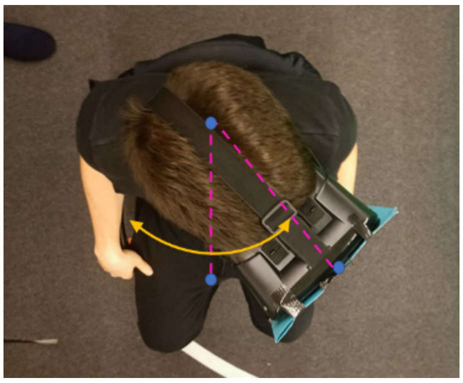


Figure 3.25: Head yaw calculation using the new method.

3.5 Conclusions

This chapter explores deeper into the technical details of the pose estimation algorithm, offering a comprehensive guide on its development. It highlights the key considerations and challenges encountered during the process, and presents a comparison between two RGB-D cameras for three-dimensional pose estimation: the Intel Realsense D455 and the Microsoft Kinect XBOX 360. Additionally, an in-depth analysis of the Mediapipe framework used for algorithm development is provided, focusing on the pose, hand, and face detection frameworks. Furthermore, the chapter demonstrates the development of a virtual reality interface using a web framework, which can be accessed through a VR headset.

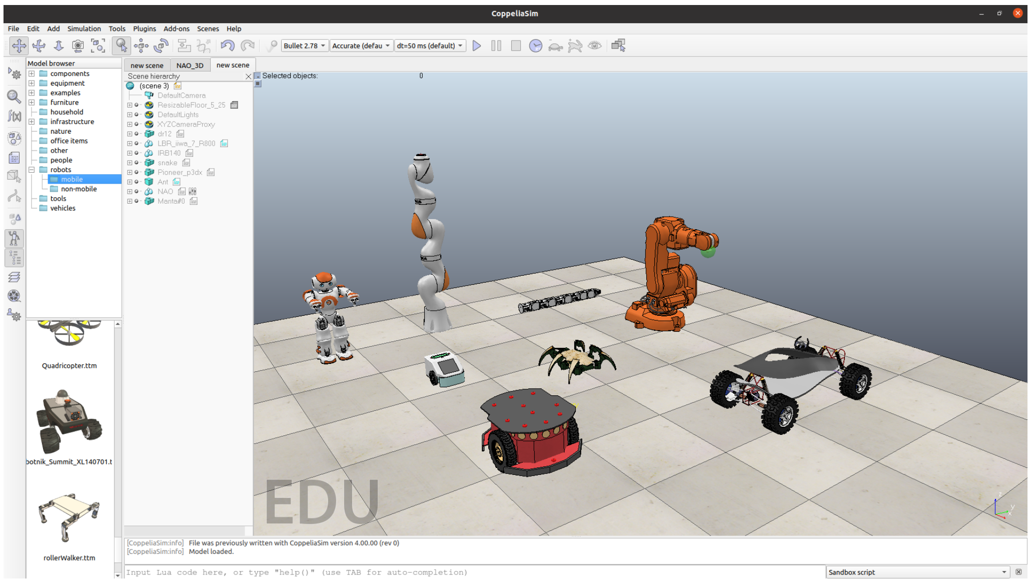
Chapter 4

Simulation Framework

In order to ensure the successful deployment of the developed algorithm, it is essential to perform simulations prior to real-world tests. This ensures that the algorithm is functioning properly and that the service robot is able to complete its daily tasks effectively. By running simulations, any potential issues can be identified and addressed before implementation in the real environment.
This chapter is divided into two parts. The first part focuses on describing the simulation environment used to perform the tests, as well as introducing the service robot used. A detailed explanation of how the developed scripts and the Robot Operating System (ROS) were used to communicate with the simulation environment is also provided. The second part of the chapter focuses on performing the tests with the robot. It explains the methods used to perform the tasks and provides an analysis of the results obtained from the tests.

4.1 Simulation Environment: CoppeliaSim

The software used for the simulations was CoppeliaSim [36], a cutting-edge simulation platform that offers powerful visualization tools and robust physics engines. It provides users with realistic simulations, using different physics engines, the ability to interact with the world during the simulation and mesh manipulation. Due to its ability to create simulations for robotics, virtual reality, medical and industrial applications, it makes this software an ideal choice.



```
[CoppeliaSim:info]  File was previously written with CoppeliaSim version 4.00.00 (rev 0)
[CoppeliaSim:info]  Model loaded.
Input Lua code here, or type "help()" (use TAB for auto-completion)
```

Figure 4.1: Coppelia simulation environment.

All simulations performed were developed on an Asus X550LB-XO085H laptop. The Central Process Unit (CPU) is an Intel Core i5 4200U, the Graphics Processing Unit (GPU) is an NVIDIA GeForce GT 740M, and 4GB DDR3 of Random-Access Memory (RAM). As storage drives, the laptop has a 240 GB Solid-State Drive (SSD). The Operating System (OS) used is Ubuntu 20.04.5 LTS.

4.1.1 NAO Robot

One of Coppelia's features is its extensive library of pre-built models, including mobile robots. The model used to simulate the service robot was the NAO robot, a realistic and sophisticated representation of a humanoid robot.

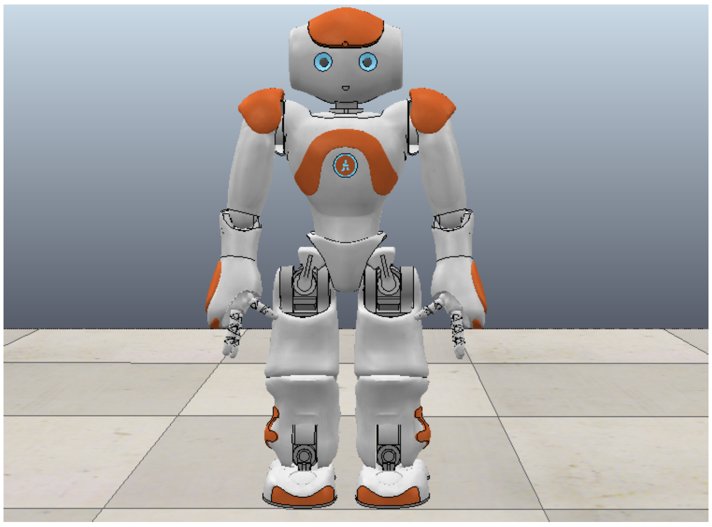


Figure 4.2: NAO robot.

The Coppelia NAO robot is an advanced humanoid robot capable of performing a variety of tasks. It has a total of 28 degrees of freedom (DoF) distributed across its body. The various DoF enable the robot to move its head, arms, and legs with great precision and accuracy. For the simulation, only the joints that are above the waist, namely the neck, shoulder, elbow, wrist, and fingers, were used. The standard definitions provided by Coppelia were also used in order to obtain a realistic implementation compared to a real service robot. Extensive research was conducted on the various degrees of freedom of each of the joints used in the robot, focusing on the joints range. The documentation provided by the manufacturer of the NAO robot, Aldebaran, was consulted to obtain detailed information about its specifications [37].

4.1.2 Head joints

The head joints, referred to as HeadYaw and HeadPitch, are a set of two joints responsible for the twisting (Z-axis) and up-down (Y-Axis) movement of the neck, respectively. Regarding range, HeadYaw has a minimum of -119.5° and a maximum of 119.5°, while Headpitch has a minimum of -38.5° and a maximum of 29.5°. As illustrated in Figure 4.3, each of the head joints is depicted.

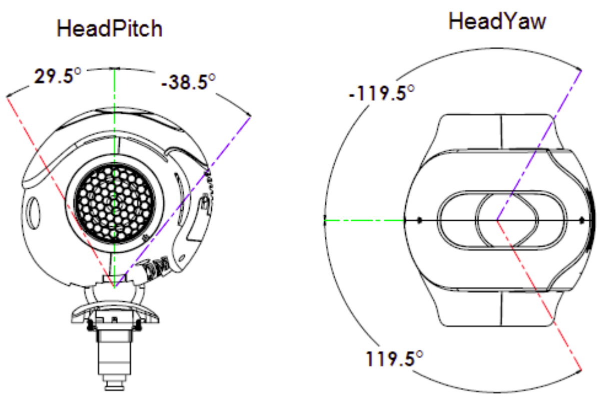


Figure 4.3: Head joints and respective ranges.

To reduce the risk of collision, the developers of the NAO robot have implemented a safety feature that automatically restricts the HeadPitch value based on the current HeadYaw value. NAO's documentation provides a comprehensive table containing all the necessary limitations.

Table 4.1: HeadPitch motion range according to the HeadYaw value.

HeadYaw	**HeadPitch Min**	**HeadPitch Max**
(degrees)		
-119.52	-25.73	18.91
-87.49	-18.91	11.46
-62.45	-24.64	17.19
-51.74	-27.50	18.91
-43.32	-31.40	21.20
-27.85	-38.50	24.18
0.0	-38.50	29.51
27.85	-38.50	24.18
43.32	-31.40	21.20
51.74	-27.50	18.91
62.45	-24.64	17.19
87.49	-18.91	11.46
119.52	-25.73	18.91

4.1.3 Arm joints

The arm joints are a collection of several joints, which can be divided into three different groups: shoulder, elbow and wrist. The shoulder group has two joints, called ShoulderPitch and ShoulderRoll, responsible for the forward-backward (Y-axis) and right-left (Z-axis) movements, respectively. The elbow group, on the other hand, is a set of two joints, ElbowYaw and ElbowRoll, where the first one is responsible for the shoulder twisting movement (X axis). Although this degree of freedom is in the elbow group, the goal is to simulate shoulder torsion, while the second joint is responsible for the open-close movement of the elbow (Z axis). Finally, the wrist group has a single joint, called WristYaw, which has the wrist torsion function (X axis).

Regarding range, ShoulderRoll has a minimum range of -18° and a maximum range of 76°. As for ShoulderPitch and ElbowYaw, both have the same range, between -119.5° and a maximum of 119.5°, while ElbowRoll is limited to a range between -88.5° and 88.5°. Finally, the only joint in the pulse group, WristYaw, has a minimum range of -104.5° and a maximum of 104.5°. Figure 4.4 shows the visual representation of all joints and their ranges.

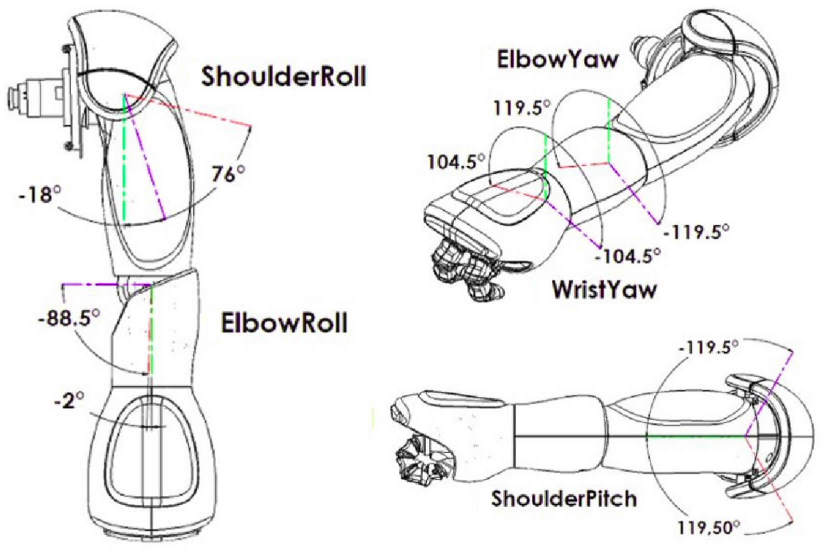


Figure 4.4: Arm joints and respective ranges.

4.1.4 Hand links

NAO's hand is composed of three fingers, with one being the thumb. Each finger has a set of three links, while the thumb has only two. During the simulation, hand's control is limited to two states: open or closed. Figure 4.5 is a representation of NAO hands, illustrating each finger's links.

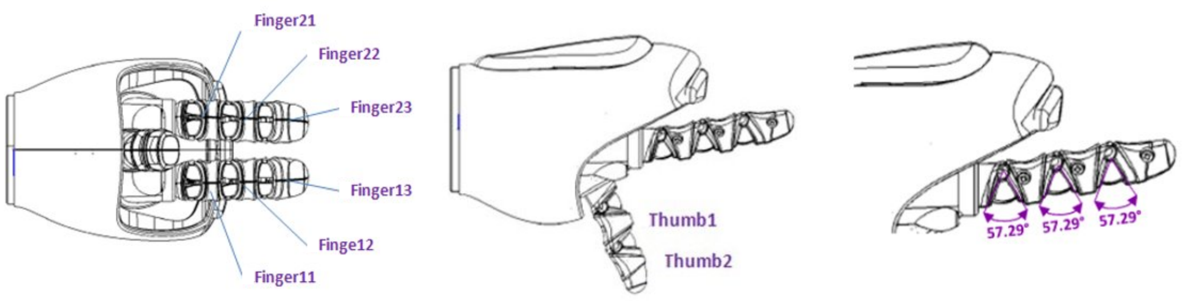


Figure 4.5: Finger links and respective ranges.

4.2 ROS Framework

One of the tools provided by Coppelia for controlling robots is through a ROS node. Robotic Operating System (ROS) is a pseudo-operating system designed to facilitate the development of robot applications. The aim of ROS is to create a unified standard for robot application development, allowing software to be written independently of the underlying hardware. ROS provides a wide range of tools and libraries that enable efficient communication between multiple computers on the same network.

The choice of ROS is based on its robustness, scalability and flexibility, as well as its applicability and the vast number of compatible libraries available. It is also the most widely used platform when it comes to developing robotic applications, providing an extensive range of features and capabilities.
The NAO robot control system consists of three ROS nodes, namely */sim_ros_interface* for the simulation, */NAO_Control* for the pose estimation script, and */Robot_vision* for the robot vision. Figure 4.6 depicts the interaction between the three ROS nodes, as illustrated by the ROS tool, rqt_graph, via the 5 ROS topics.
In order to further improve the communication between the three ROS nodes, 5 separate topics were created to efficiently control the data transmission between them.

- */RightArm_Values* (std_msgs/Float32MultiArray) - An array with six float elements, containing the values of ShoulderRoll, ShoulderPitch, ElbowRoll, ElbowYaw, WristYaw and hand position (0 - open, 1 - closed) for the right arm.

- */LeftArm_Values* (std_msgs/Float32MultiArray) - An array with six float elements, containing the values of ShoulderRoll, ShoulderPitch, ElbowRoll, ElbowYaw, WristYaw and hand position (0 - open, 1 - closed) for the left arm.

- */Head_Values* (std_msgs/Float32MultiArray) - An array with two float elements, containing the values of HeadYaw and HeadPitch.

- */RightEye_Image* (sensor_msgs/Image) - A message containing an uncompressed image, this image being the view from the robot's right eye.

- */LeftEye_Image* (sensor_msgs/Image) - A message containing an uncompressed image, this image being the view from the robot's left eye.

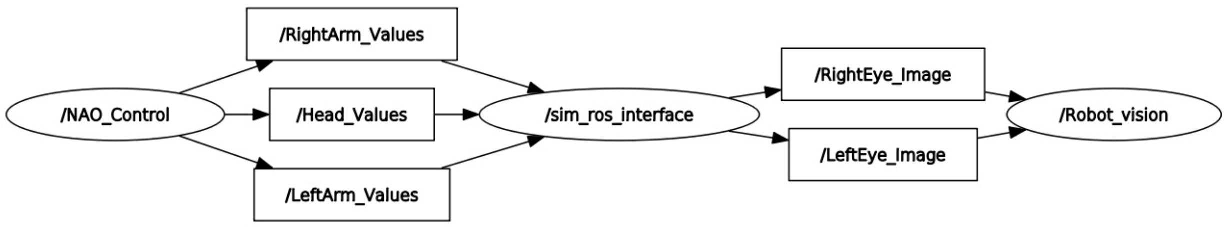
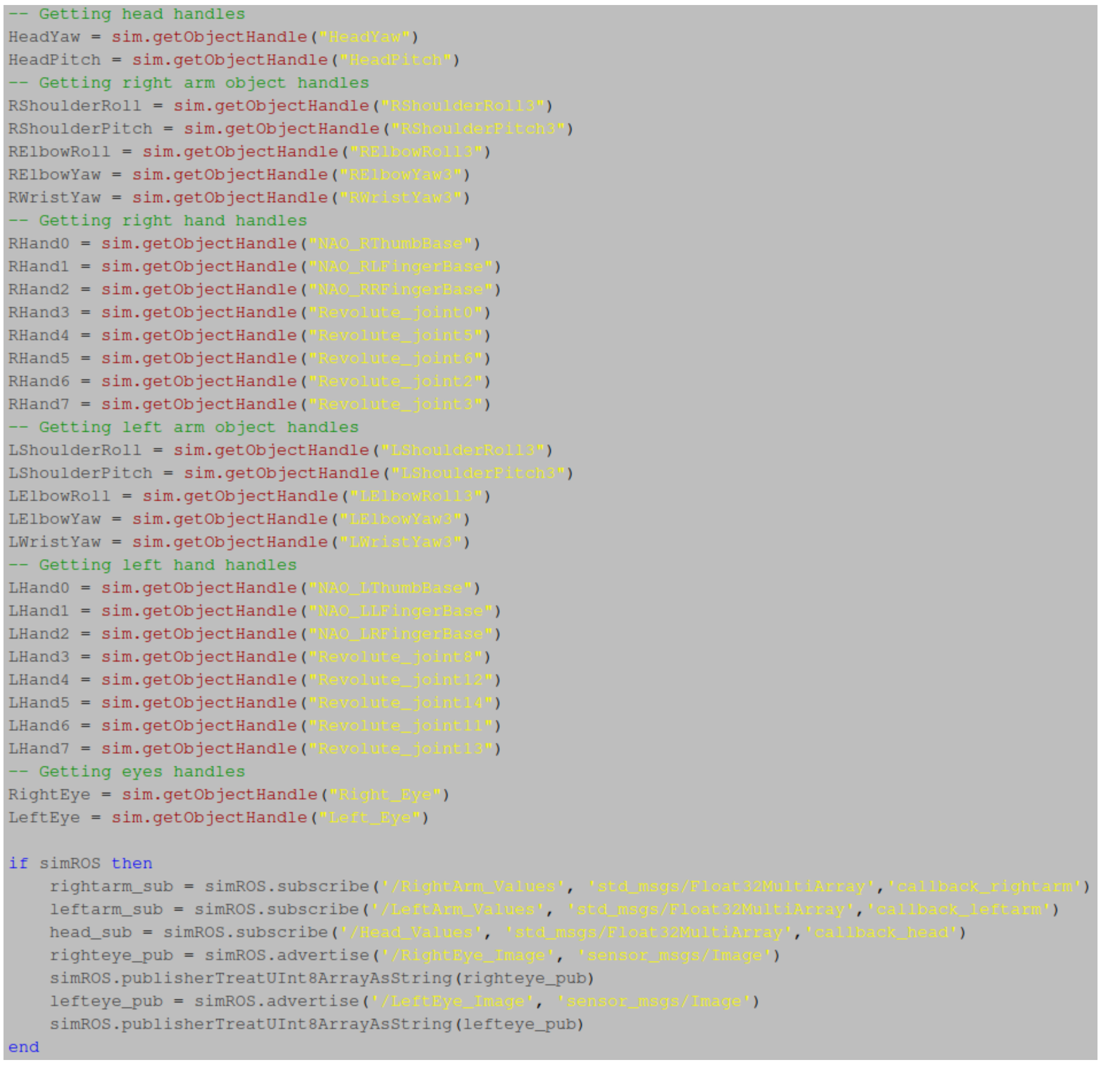


Figure 4.6: ROS rqt_graph with the three ROS nodes and the five ROS topics.

Coppelia provides embedded scripts for controlling various elements, such as scenes and objects, written in the versatile and powerful scripting language Lua. These scripts are compiled into an internal machine

language for efficient and speedy execution, granting users greater flexibility than what the ROS framework alone can offer. Through the use of these scripts, users can customize the simulator to their specific needs, unlocking a wide range of capabilities and possibilities.
An embedded script has been developed for precise control of the NAO robot. The script is segmented into several parts, each with a distinct purpose. As illustrated in Figure 4.7, the first part of the script is responsible for initializing the model and ROS, so that the robot can effectively subscribe and publish to the necessary topics.

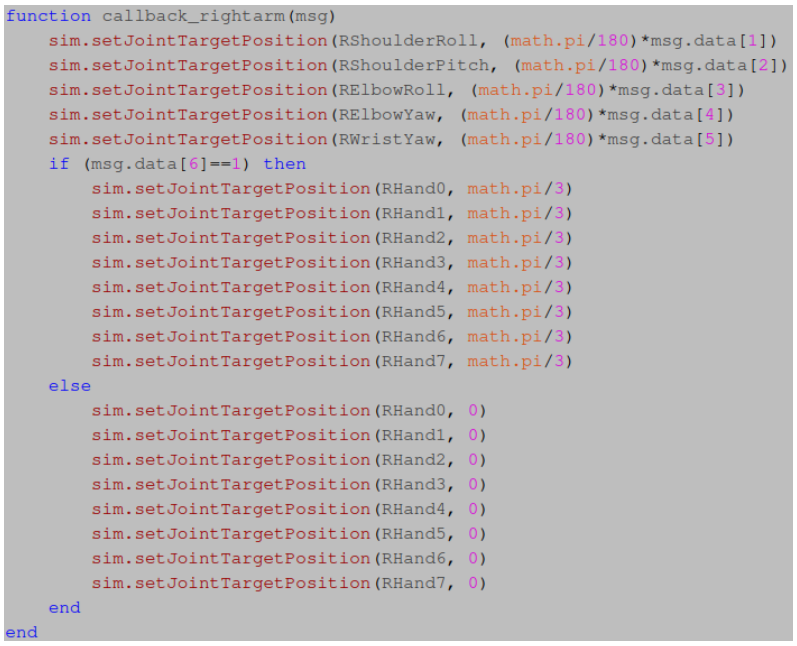


```
-- Getting head handles
HeadYaw = sim.getObjectHandle("HeadYaw")
HeadPitch = sim.getObjectHandle("HeadPitch")
-- Getting right arm object handles
RShoulderRoll = sim.getObjectHandle("RShoulderRoll3")
RShoulderPitch = sim.getObjectHandle("RShoulderPitch3")
RElbowRoll = sim.getObjectHandle("RElbowRoll3")
RElbowYaw = sim.getObjectHandle("RElbowYaw3")
RWristYaw = sim.getObjectHandle("RWristYaw3")
-- Getting right hand handles
RHand0 = sim.getObjectHandle("NAO_RThumbBase")
RHand1 = sim.getObjectHandle("NAO_RLFingerBase")
RHand2 = sim.getObjectHandle("NAO_RRFingerBase")
RHand3 = sim.getObjectHandle("Revolute_joint0")
RHand4 = sim.getObjectHandle("Revolute_joint5")
RHand5 = sim.getObjectHandle("Revolute_joint6")
RHand6 = sim.getObjectHandle("Revolute_joint2")
RHand7 = sim.getObjectHandle("Revolute_joint3")
-- Getting left arm object handles
LShoulderRoll = sim.getObjectHandle("LShoulderRoll3")
LShoulderPitch = sim.getObjectHandle("LShoulderPitch3")
LElbowRoll = sim.getObjectHandle("LElbowRoll3")
LElbowYaw = sim.getObjectHandle("LElbowYaw3")
LWristYaw = sim.getObjectHandle("LWristYaw3")
-- Getting left hand handles
LHand0 = sim.getObjectHandle("NAO_LThumbBase")
LHand1 = sim.getObjectHandle("NAO_LLFingerBase")
LHand2 = sim.getObjectHandle("NAO_LRFingerBase")
LHand3 = sim.getObjectHandle("Revolute_joint8")
LHand4 = sim.getObjectHandle("Revolute_joint12")
LHand5 = sim.getObjectHandle("Revolute_joint14")
LHand6 = sim.getObjectHandle("Revolute_joint11")
LHand7 = sim.getObjectHandle("Revolute_joint13")
-- Getting eyes handles
RightEye = sim.getObjectHandle("Right_Eye")
LeftEye = sim.getObjectHandle("Left_Eye")

if simROS then
    rightarm_sub = simROS.subscribe('/RightArm_Values', 'std_msgs/Float32MultiArray','callback_rightarm')
    leftarm_sub = simROS.subscribe('/LeftArm_Values', 'std_msgs/Float32MultiArray','callback_leftarm')
    head_sub = simROS.subscribe('/Head_Values', 'std_msgs/Float32MultiArray','callback_head')
    righteye_pub = simROS.advertise('/RightEye_Image', 'sensor_msgs/Image')
    simROS.publisherTreatUInt8ArrayAsString(righteye_pub)
    lefteye_pub = simROS.advertise('/LeftEye_Image', 'sensor_msgs/Image')
    simROS.publisherTreatUInt8ArrayAsString(lefteye_pub)
end
```

Figure 4.7: Model and ROS initialization.

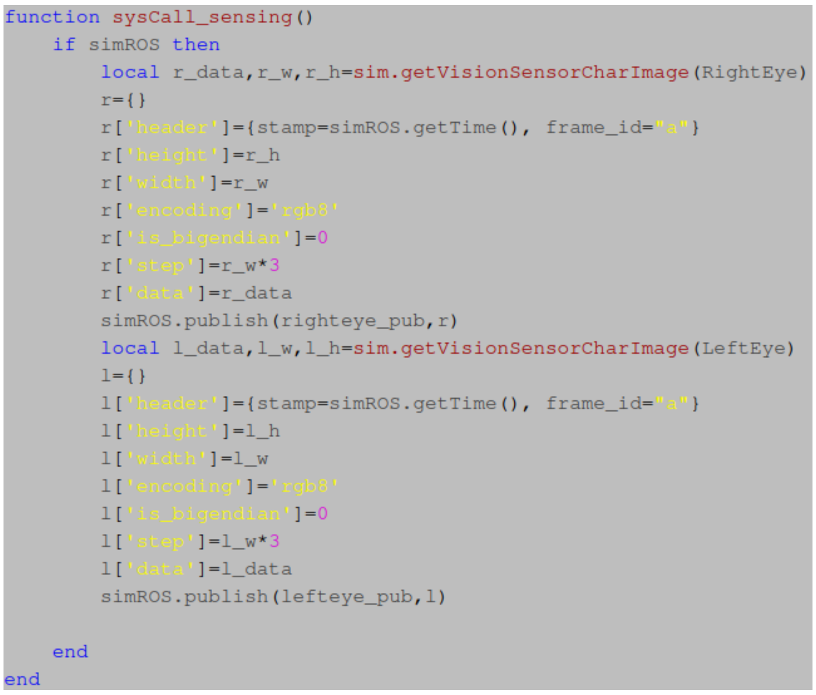
The other part of the script is related to receiving and sending data. Figure 4.8 illustrates the callback for data processing sent through the Python script, relative to the right arm. Each subscribed topic has its own callback, which receives the data values from each joint and assigns those values to the corresponding robot joints.



```
function callback_rightarm(msg)
    sim.setJointTargetPosition(RShoulderRoll, (math.pi/180)*msg.data[1])
    sim.setJointTargetPosition(RShoulderPitch, (math.pi/180)*msg.data[2])
    sim.setJointTargetPosition(RElbowRoll, (math.pi/180)*msg.data[3])
    sim.setJointTargetPosition(RElbowYaw, (math.pi/180)*msg.data[4])
    sim.setJointTargetPosition(RWristYaw, (math.pi/180)*msg.data[5])
    if (msg.data[6]==1) then
        sim.setJointTargetPosition(RHand0, math.pi/3)
        sim.setJointTargetPosition(RHand1, math.pi/3)
        sim.setJointTargetPosition(RHand2, math.pi/3)
        sim.setJointTargetPosition(RHand3, math.pi/3)
        sim.setJointTargetPosition(RHand4, math.pi/3)
        sim.setJointTargetPosition(RHand5, math.pi/3)
        sim.setJointTargetPosition(RHand6, math.pi/3)
        sim.setJointTargetPosition(RHand7, math.pi/3)
    else
        sim.setJointTargetPosition(RHand0, 0)
        sim.setJointTargetPosition(RHand1, 0)
        sim.setJointTargetPosition(RHand2, 0)
        sim.setJointTargetPosition(RHand3, 0)
        sim.setJointTargetPosition(RHand4, 0)
        sim.setJointTargetPosition(RHand5, 0)
        sim.setJointTargetPosition(RHand6, 0)
        sim.setJointTargetPosition(RHand7, 0)
    end
end
```

Figure 4.8: Right arm subscriber callback.

Finally, the part of the script shown in Figure 4.9 is responsible for sending images corresponding to the robot's view to be further processed in an external script.



```
function sysCall_sensing()
    if simROS then
        local r_data,r_w,r_h=sim.getVisionSensorCharImage(RightEye)
        r={}
        r['header']={stamp=simROS.getTime(), frame_id="a"}
        r['height']=r_h
        r['width']=r_w
        r['encoding']='rgb8'
        r['is_bigendian']=0
        r['step']=r_w*3
        r['data']=r_data
        simROS.publish(righteye_pub,r)
        local l_data,l_w,l_h=sim.getVisionSensorCharImage(LeftEye)
        l={}
        l['header']={stamp=simROS.getTime(), frame_id="a"}
        l['height']=l_h
        l['width']=l_w
        l['encoding']='rgb8'
        l['is_bigendian']=0
        l['step']=l_w*3
        l['data']=l_data
        simROS.publish(lefteye_pub,l)

    end
end
```

Figure 4.9: Robot vision publishing.

The last part of the script is responsible for the environment's clean up, shutting down the publishers and subscribers. This is done to ensure that all resources are released and all messages sent and received are processed.

4.3 Simulation Tests

Simulation tests are a valuable tool for predicting the performance of a system under a variety of conditions. These tests use computer models to simulate real-world scenarios, allowing engineers and researchers to evaluate a system's behavior without the need for physical prototypes. In addition to helping optimize a system's performance, simulation testing is also a form of software verification, safeguarding the integrity of the hardware. It can save time and resources by identifying potential issues early in the design process, and it can be used to explore different design configurations.

4.3.1 Controlling the robot

The first test conducted was to control the robot through basic movements. The purpose of this test was to evaluate the Robot Operating System (ROS) framework efficiency and to assess whether the robot was accurately replicating the movements commanded by the teleoperator. Figure 4.10 displays a few examples of positions made by the teleoperator and copied by the robot.

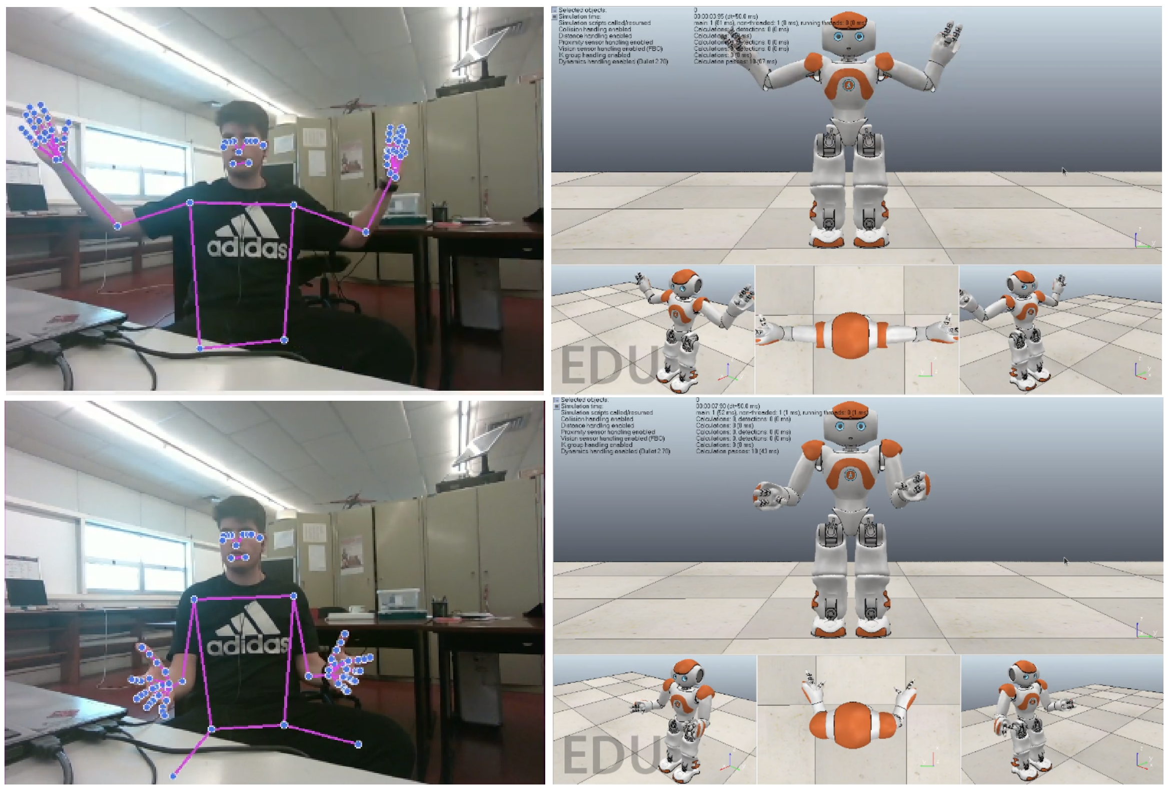


Figure 4.10: Robot control movement test.

4.3.2 Object manipulation

The second test consisted of object manipulation. This test aim was to handle an object with precision and accuracy. To achieve this goal, strong physical dexterity and control were needed to manipulate the object safely and precisely. Figure 4.11 shows the results of this test, where the robot successfully picks up the object.

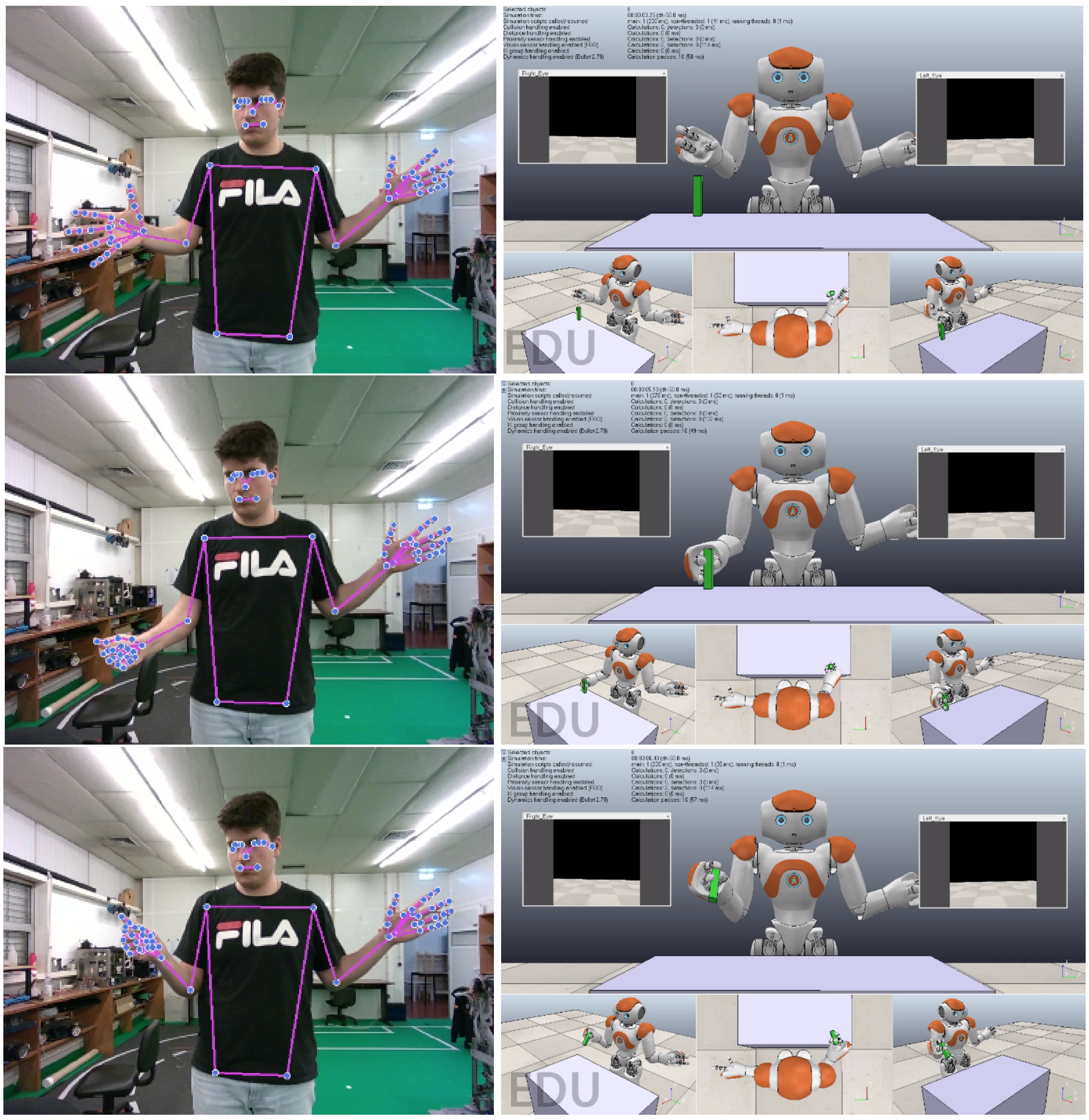


Figure 4.11: Object manipulation test.

4.4 Robot vision through VR

In order to create an even more immersive experience through telepresence, it was necessary for the operator to have access to the robot's vision. To achieve this, the simulator sends images corresponding to the robot's vision through the ROS framework to a web application powered by Flask. Flask was chosen as the web framework for this application due to its ease of implementation. An illustration of this can be seen in Figure 4.12 (a), which shows two windows next to the robot, each one being the view of each of the robot's eyes. In Figure 4.12 (b), the same images are streamed in real time in the web application, providing the operator with an enhanced level of immersion.

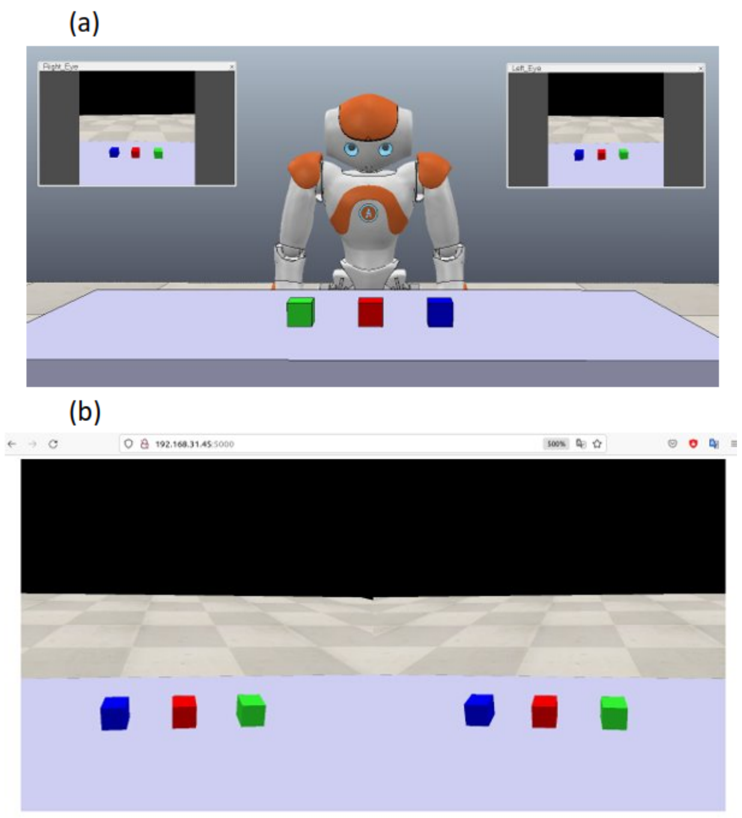


Figure 4.12: Robot vision on Coppelia (a) and robot vision streamed in real time (b).

4.4.1 Object manipulation using VR

The objective of the final test is to integrate the techniques demonstrated in previous tests, using the robot's visual capabilities to manipulate an object. However, executing this test proved to be a challenge due to a low frame rate, which resulted in an uncomfortable experience when using the headset.

4.5 Conclusions

In this chapter, the simulator used for conducting initial experiments is introduced. By conducting simulations, it is possible to identify potential issues without compromising the integrity of the hardware.
The simulation environment employed in this study was CoppeliaSim, and the NAO robot was used to

simulate the service robot. The Robot Operating System (ROS) framework was used for communication between the simulation environment and the control script.

The first experiment focused on controlling the robot through basic movements, with the aim of testing communication through ROS and the conversion of human movements into robot movements. The second experiment involved manipulating an object through the control of the robot, testing the robustness of the developed algorithm. The third and final experiment, though unable to be carried out due to a low frame rate, aimed to manipulate an object using virtual reality as the operator's perspective.

All experiments were successfully executed, providing a thorough evaluation of the robot's control capabilities and the effectiveness of the algorithm developed for converting human movement into robot movement. The virtual reality interface provided an immersive experience, allowing for a heightened sense of presence as if being in the robot's position, despite the challenge of integrating it with the task of object manipulation.

The subsequent chapter, Chapter 5, delves into similar experiments on the actual robot within a controlled environment.

Chapter 5

Real World Tests

Teleoperating a robot with an anthropomorphic arm presents a unique set of challenges and opportunities. It allows tasks that require a high degree of dexterity and precision to be performed by the robot, but also requires a reliable and responsive communication system to ensure that the operator's commands are accurately transmitted.

This chapter documents the real-world tests conducted to evaluate the performance of the teleoperated robot. Descriptions of the test setup, tasks, results, any issues encountered and how they were addressed are also provided.

5.1 CHARMIE Antropomorphic Arm

As mentioned earlier, CHARMIE is an anthropomorphic service robot, capable of performing household chores and helping elderly people. One of its standout features is its robotic arm, which was designed and developed by Hafiz Syed Fawad Adil [38]. This arm, based on the Inmoov model, was created to replicate the movements and actions of a human arm. It was made using a range of motors, joints, and connections to provide high-precision capabilities and facilitate a variety of tasks. Inmoov [39], the model upon which CHARMIE's arm is based, is an innovative, open-source, life-size robot created by the famous French sculptor and inventor, Gael Langevin.

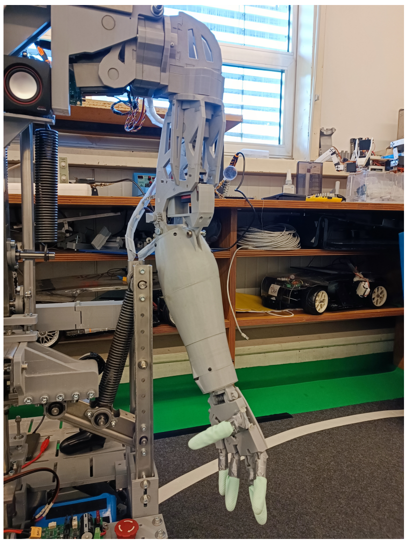


Figure 5.1: CHARMIE antropomorphic arm.

According to Fawad's thesis [38], the anthropomorphic arm has five degrees of freedom: shoulder roll, shoulder pitch, elbow yaw, elbow roll, and wrist. Each of these degrees of freedom has a minimum and maximum range due to limitations imposed during the construction of the arm. A summary of these ranges can be found in Table 5.1.

Table 5.1: Range of each anthropomorphic arm joint.

Joint	**Minimum range**	**Maximum range**
Shoulder Roll	6°	65°
Shoulder Pitch	30°	160°
Elbow Yaw	70°	120°
Elbow Roll	45°	135°
Wrist	30°	150°

The anthropomorphic arm also features an anthropomorphic hand, unlike the NAO robot used in simulation. The hand is composed of individual servos, each controlling the finger's flexion and extension through fishing line-based tendons. Figure 5.2 illustrates the forearm's structure, which makes up the finger control system. Like the rest of the arm, each finger has a range of motion, which can be observed in Table 5.2.

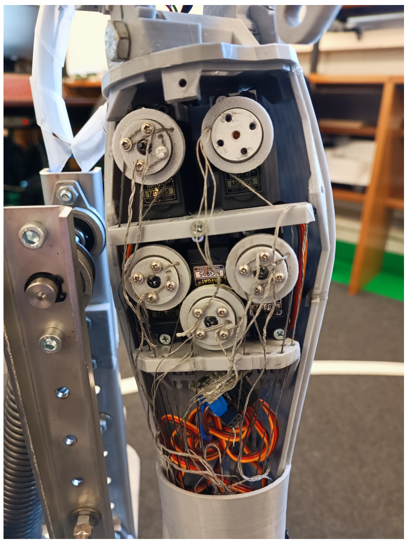


Figure 5.2: CHARMIE tendon system to control the fingers.

Table 5.2: Range of each anthropomorphic hand finger.

Finger	**Minimum range**	**Maximum range**
Thumb	70°	160°
Index finger	15°	125°
Middle finger	0°	100°
Ring finger	20°	120°
Pinky	20°	110°

The entire control system of the servos is done through an Arduino Mega, as depicted in Figure 5.3.

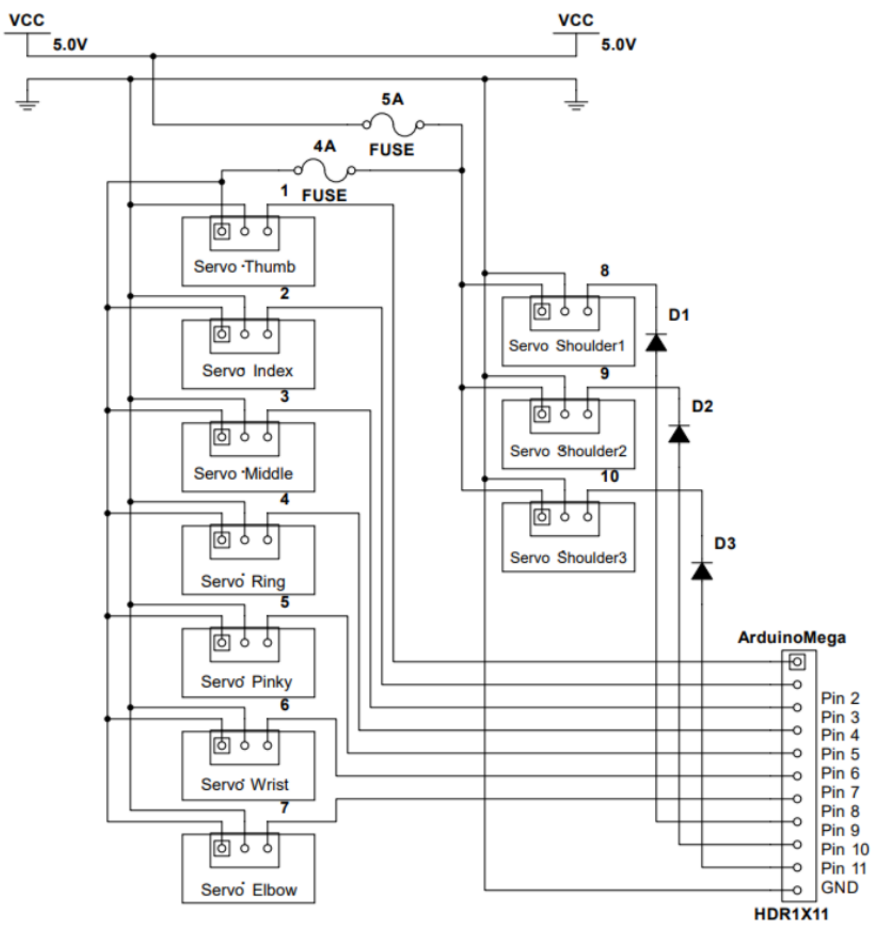


Figure 5.3: CHARMIE's arm circuit diagram.

5.2 CHARMIE Neck

CHARMIE's neck is composed of two powerful servos, each providing one degree of freedom. The first servo controls the neck twisting movement, corresponding to Head Yaw, while the second servo controls the up-down movement, corresponding to Head Pitch. This powerful combination of servos allows CHARMIE to move its head with greater precision and accuracy, enabling it to better interact with its environment.

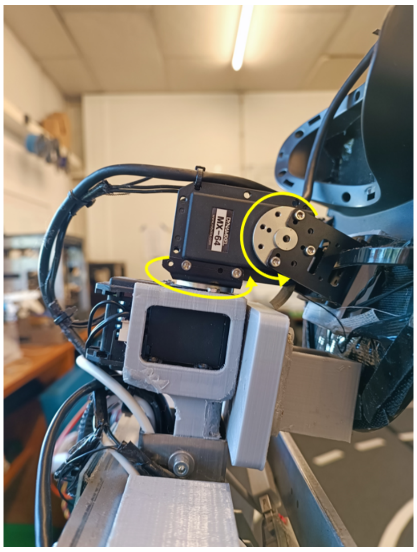


Figure 5.4: CHARMIE's neck degrees of freedom.

5.3 Controlling the antropomorphic arm

The initial test conducted was to evaluate the functionality of the robotic arm. To perform this test, modifications to the code were made, specifically to the communication with the Arduino Mega. As shown in Figure 5.5, this test involved establishing communication between the various devices.

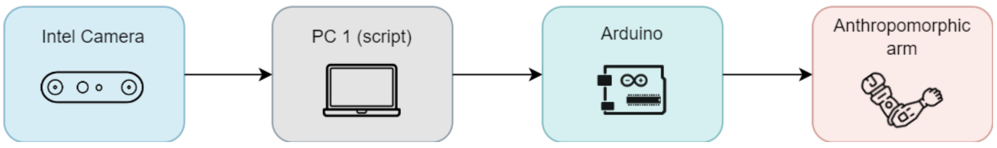


Figure 5.5: Communication between devices on antropomorphic arm control test.

As the values being sent to the servos are constantly changing, it is not safe to apply them directly. Therefore, a filter is applied to smooth these values. The same method used for measuring depth is used, which involves a cyclic array that updates the values at each iteration and applying an averaging filter to these arrays, in order to smooth the angle value sent to the servos.
Another important consideration when sending values to the servos is their range of motion. The calculated values may exceed the servo's range, so an interpolation function is needed to convert the calculated angle into a similar value that can be applied to the servo. This function takes 5 parameters: the calculated angle, the minimum and maximum values that the calculated angle can have and the minimum and maximum values of the servo. This prevents the servos from going beyond their range.

$$Servo\ value = max \begin{cases} min \begin{cases} out_{max} \\ \frac{(x-in_{min}).(out_{max}-out_{min})}{in_{max}-in_{min}} + out_{min} \end{cases} \\ out_{min} \end{cases} \quad (5.1)$$

Where *Servo value* is the converted servo value, *x* is the calculated value, in_{min} and in_{max} are the minimum and maximum values for the calculated value, and out_{min} and out_{max} are the minimum and maximum values for the servo range.

The objective of the test was to evaluate the effectiveness of CHARMIE's object handling capabilities using Intel Realsense technology. The Intel Realsense camera was placed on top of CHARMIE's head and operated with the operator in front of it. The results of this initial test can be seen in Figure 5.6, which show that CHARMIE was able to successfully handle the object with precision and accuracy.

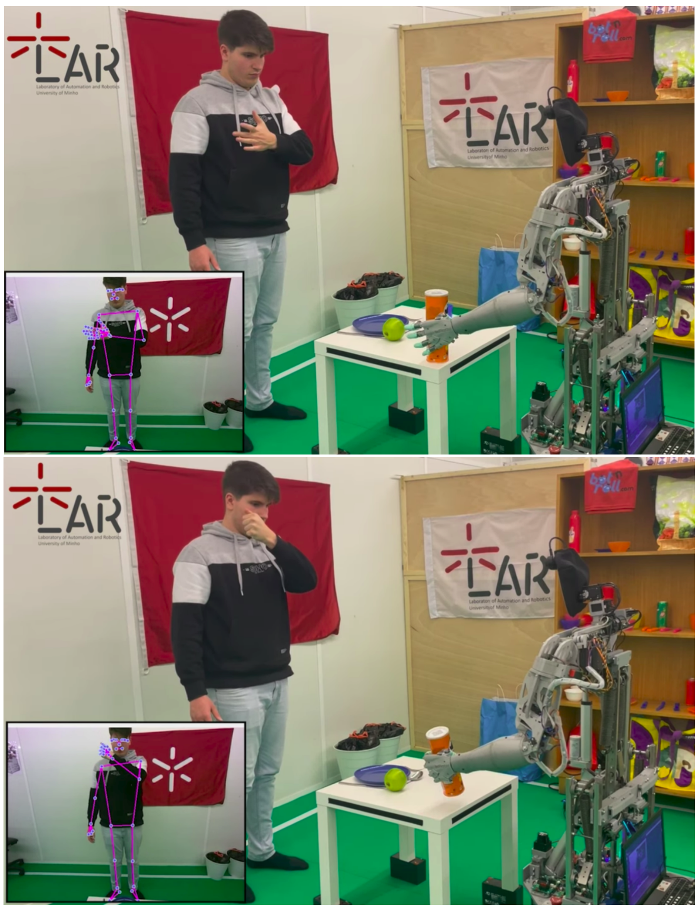


Figure 5.6: Object manipulation test [images from [40]].

5.4 Controlling the robot

The final test is the ultimate challenge, as it requires the operator to manipulate an object in virtual reality while controlling CHARMIE's robotic arm. Additionally, the operator has access to the robot's vision, which gives an even more immersive experience. By tilting the head and controlling CHARMIE's neck, the operator can observe the environment in 3D and make precise movements. Furthermore, the robot arm is accurately controlled by the operator's movements, allowing for precise and delicate movements. All of these elements make the test a true test of the operator's skills and provide an engaging experience.

The diagram in Figure 5.7 illustrates the intercommunication of the various devices required to execute this test. The operator is situated in front of the Intel camera, responsible for transmitting the images to PC 1 for processing. The calculated values of all joints are then sent to PC 2 via sockets, which subsequently sends the values to CHARMIE's arm and neck. Concurrently, PC 2 receives images from the camera that is mounted on the robot and performs stereoscopic rendering on the image to be transmitted to the web. These images can then be accessed via the VR headset, which is placed on the operator's head, creating an immersive experience.

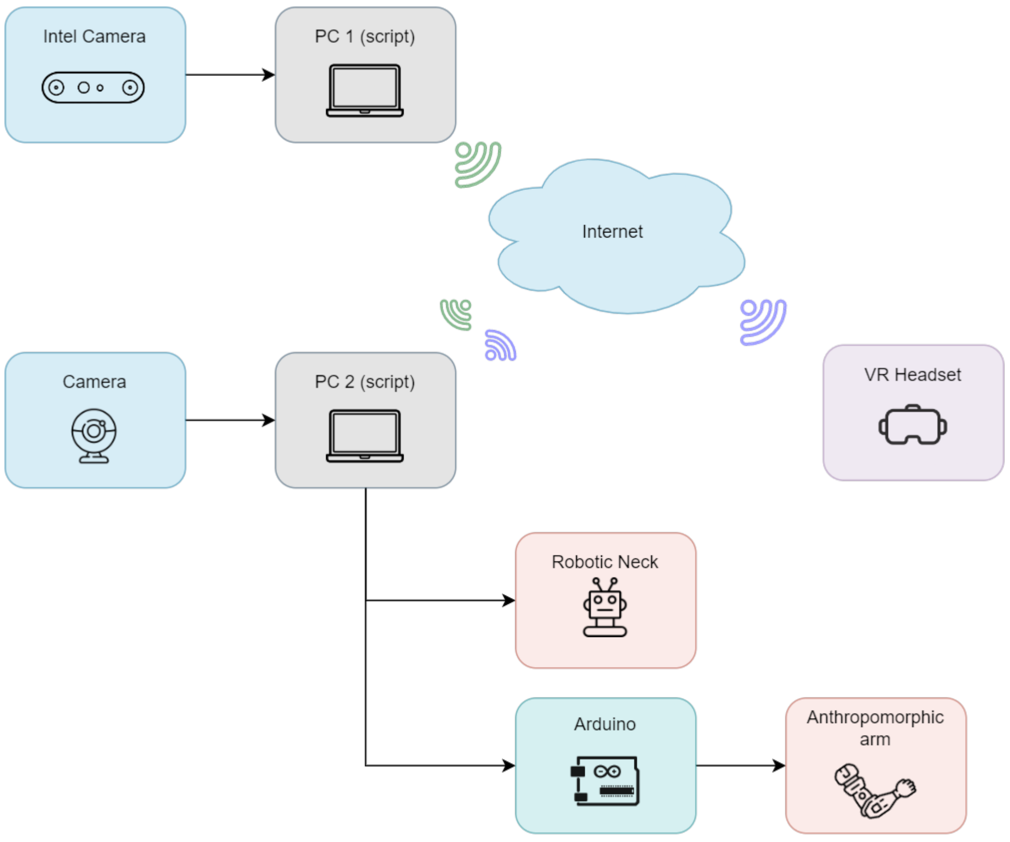
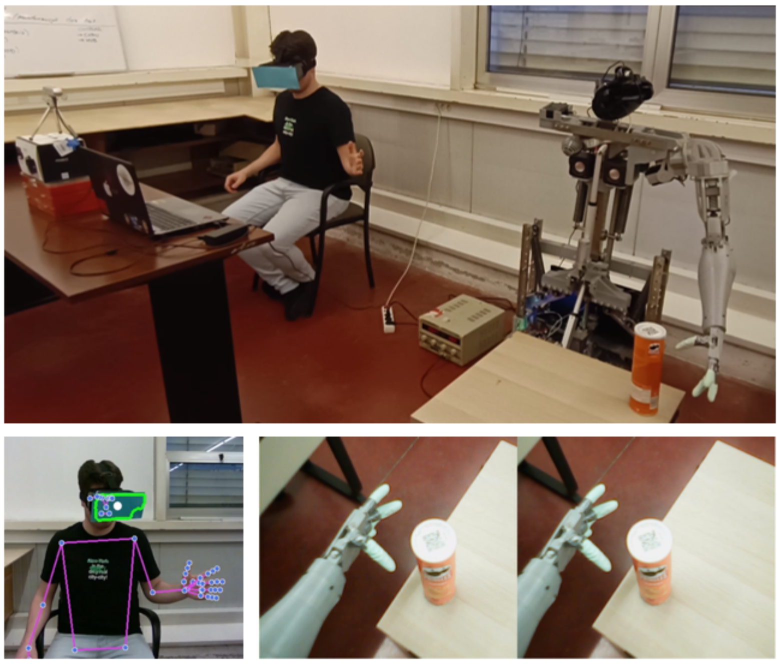


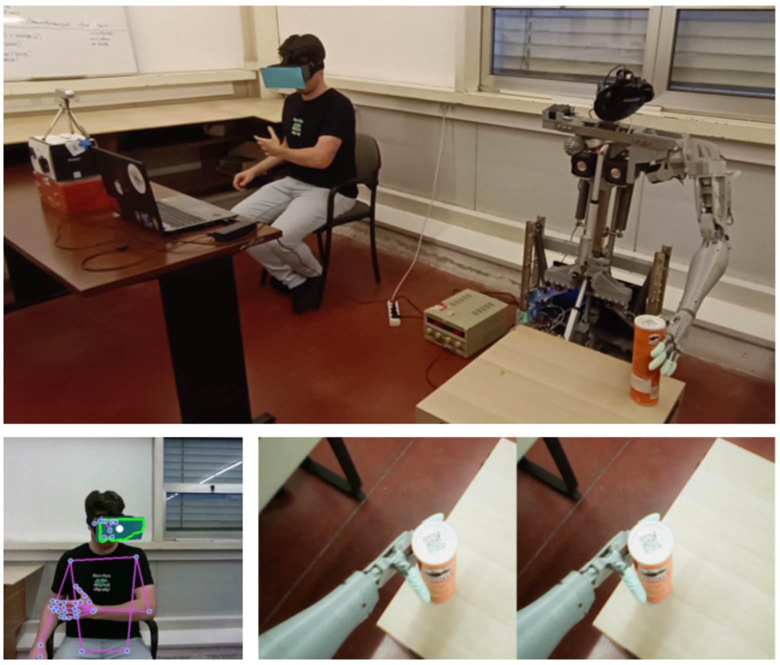
Figure 5.7: Communication between devices on robot control test.

The results of the test are illustrated in Figure 5.8, which illustrates three distinct action frames. In each of these frames, it is possible to observe the robot's perspective at that moment, as well as the pose detection. The first frame shows the arm in proximity to the can (Figure 5.8 (a)), the second frame depicts the arm in contact with the can (Figure 5.8 (b)) and the third and final frame demonstrates the successful grasping of the can (Figure 5.8 (c)), thus completing the test.

(a)



(b)



(c)

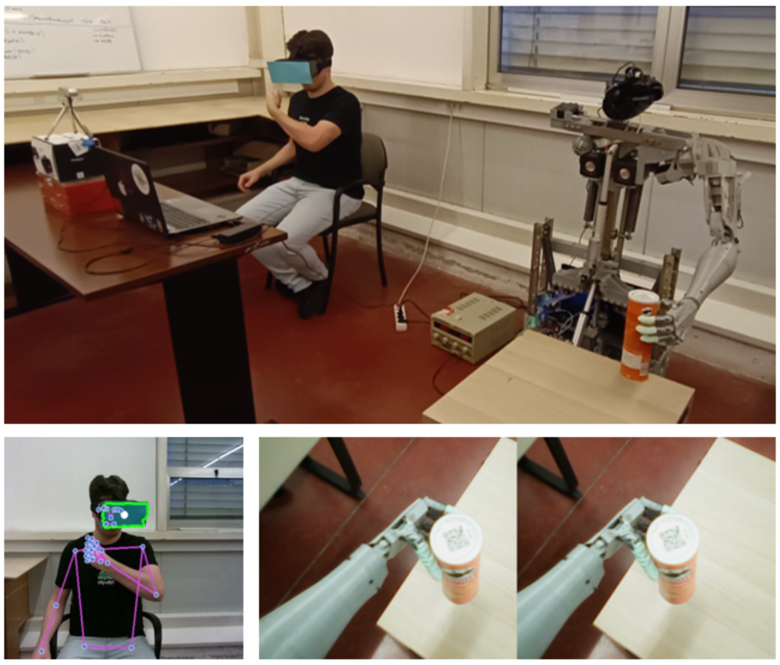


Figure 5.8: Object manipulation test through VR.

5.5 Conclusions

In this chapter, the transition from the simulated environment to the real-world setting is undertaken. A comprehensive overview of the capabilities of both the anthropomorphic arm and the robot's neck is provided.

The experiments conducted in this chapter were similar to those performed in the simulation environment, with the first experiment focusing on controlling the robotic arm. The second and final experiment encompassed all previous tests, involving the manipulation of an object using the robotic arm through the virtual reality interface.

The results of these experiments displayed robustness in both the pose estimation algorithm and neck control, enabling the operator to effectively control the robot remotely with a high degree of precision.

Chapter 6

Conclusions

The work presented in this dissertation concerned the introduction of the theoretical foundations for the development of a pose estimation algorithm, capable of controlling an anthropomorphic robot through mimicry and the use of virtual reality, thus allowing its teleoperation.

Initially, a literature review was conducted, presenting the solutions that existed at that time in the various topics addressed, namely service robots, teleoperation, virtual reality and pose estimation, in order to build a strong theoretical basis for understanding the basic concepts addressed during the dissertation.

The second part of the dissertation focused on the development of the pose estimation algorithm using the Mediapipe framework, with an RGB-D camera capable of measuring the depth of objects and thus transforming 2D projections into 3D projections. For the choice of camera, a detailed comparison was made between two options on the market: the Intel Realsense D455 and the Microsoft Kinect XBOX 360. During this part, an in-depth analysis of the operation of the various Mediapipe frameworks was conducted, namely pose, hand and face detection.

A simulation environment was developed in CoppeliaSim, using the NAO robot to simulate the robot used for the real tests. To control it, the ROS framework was chosen to facilitate the communication between the simulator and the control script. The first test carried out in simulation was to control the robot through simple movements, testing the communication and conversion of human movements to robot movements. In the second test, the objective was to manipulate an object, using the robot to manipulate an object placed on a surface. It was also tested sending images corresponding to the vision of each of the robot's eyes, through a Web framework, to be later accessed through the VR headset. The last test consisted of a compilation of all the tests previously carried out, with the objective of manipulating an object using the robot's vision as a guide, but it was not possible due to the low frame rate.

The last part of the dissertation focused on testing the algorithm in a real-world environment. A detailed description of the functionalities of the anthropomorphic arm and neck of the robot was provided. The first test involved testing the arm control by positioning the operator in front of the robot and manipulating an object. The second and final test was the culmination of the dissertation, where the objective was to manipulate an object using the robot's vision as a guide. The operator was positioned besides the robot to have a clear perspective of the mimicry performed by the robot. Both the robotic arm and the neck of the robot were able to successfully follow the movements of the operator and complete the task.

6.1 Further Work

This dissertation presents a comprehensive study that lays the foundation for controlling the CHARMIE robot through teleoperation and virtual reality. It also introduces the theoretical underpinnings for the development of advanced projects that use pose estimation control, thereby serving as a catalyst for future advancements in the field.
To enhance both efficiency and the detection of human pose in virtual reality environments, new developments and features are necessary. Upgrades that are currently being considered for future study include:

1. The implementation of an innovative technique to enhance the precision of rotational calculations, specifically for Wrist Yaw and Elbow Yaw, by using Inertial Measurement Units (IMUs).

2. The use of data from the gyroscope on a user's cell phone, integrated with the VR headset, to achieve greater accuracy in tracking head movements.

3. The creation of a user-friendly interface that allows for seamless access to the robot's outputs within the virtual environment, enhancing the overall user experience.

Bibliography

[1] Pollen Robotics. ”Reachy”, 2021. [Online]. Available: https://www.pollen-robotics.com/reachy/. [Acessed: 2021-1-3].

[2] Virtual Reality Oasis. ”I Remotely Control A ROBOT Using Virtual Reality”, 2021. [Online]. Available: https://www.youtube.com/watch?v=f64NRspWR4E&t=1s. [Acessed: 2021-12-27].

[3] Tiago Ribeiro, Fernando Gonçalves, Inês S Garcia, Gil Lopes, and António F Ribeiro. Charmie: A collaborative healthcare and home service and assistant robot for elderly care. *Applied Sciences*, 11(16):7248, 2021.

[4] Tiago Ribeiro, Fernando Gonçalves, Marco Luís, Eduardo Fernandes, Rui Silva, António Dias, Nuno Oliveira, Diogo Martins, Syed Fawad, Gil Lopes, et al. Lar@ home 2023 team description paper.

[5] Robocup. ”Robocup@Home”, 2023. [Online]. Available: https://2023.robocup.org/. [Acessed: 2022-11-22].

[6] Robocup. ”Robocup@Home Rules”, 2023. [Online]. Available: https://athome.robocup.org/rules/. [Acessed: 2022-11-22].

[7] Doina Pisla, Hannes Bleuler, Aleksandar Rodic, Calin Vaida, and Adrian Pisla. *New Trends in Medical and Service Robots: Theory and Integrated Applications*, volume 16. Springer, 2013.

[8] International Federation of Robotics. ”Service Robots”, 2021. [Online]. Available: https://ifr.org/service-robots/. [Acessed: 2021-12-31].

[9] International Federation of Robotics. ”IFR Press Conference”, 2020. [Online]. Available: https://ifr.org/downloads/press2018/Presentation_WR_2020.pdf. [Acessed: 2021-12-31].

[10] S. Lichiardopol. *A survey on teleoperation*. DCT rapporten. Technische Universiteit Eindhoven, 2007. DCT 2007.155.

[11] Thomas B Sheridan. Telerobotics. *Automatica*, 25(4):487–507, 1989.

[12] Thomas B Sheridan. Teleoperation, telerobotics and telepresence: A progress report. *Control Engineering Practice*, 3(2):205–214, 1995.

[13] NASA. ”Robonaut 2”, 2015. [Online]. Available: www.nasa.gov/robonaut2. [Acessed: 2022-1-1].

[14] Paolo Arcara and Claudio Melchiorri. Control schemes for teleoperation with time delay: A comparative study. *Robotics and Autonomous systems*, 38(1):49–64, 2002.

[15] Tamas Haidegger and Zoltan Benyo. Surgical robotic support for long duration space missions. *Acta Astronautica*, 63(7-10):996–1005, 2008.

[16] Volker Urban, Matthias Wapler, Jens Neugebauer, Andrea Hiller Dipl.-Ing, Jan Stallkamp Dipl.-Ing, and Thomas Weisener Dr.-Ing. Robot-assisted surgery system with kinesthetic feedback. *Computer Aided Surgery: Official Journal of the International Society for Computer Aided Surgery (ISCAS)*, 3(4):205–209, 1998.

[17] Intuitive. ”About da Vinci Systems, 2022. [Online]. Available: www.davincisurgery.com/da-vinci-systems/about-da-vinci-systems. [Acessed: 2022-1-1].

[18] Geraldo Wellington Rocha Fernandes and José André Peres Angotti. Homem e máquina: Entre o real e o virtual man and machine: Between the real and virtual.

[19] G Coates. Program from invisible site - a virtual sho, 1992. a multimedia performance work presented by George Coates Performance Works, San Francisco, CA, March, 1992.

[20] Frank Biocca. Virtual reality technology: A tutorial. *Journal of communication*, 42(4):23–72, 1992.

[21] Tyler Ray Bell. *High-quality, real-time 3D video visualization in head mounted displays*. PhD thesis, Iowa State University, 2014.

[22] Ben Lang. ”Oculus Quest 2 Review – The Best Standalone Headset Gets Better in (Almost) Every Way”, 2020. [Online]. Available: https://www.roadtovr.com/oculus-quest-2-review-better-in-almost-every-way/, [Acessed: 2021-1-2].

[23] Smart VR Lab. ”3DoF vs 6DoF in VR, 2021. [Online]. Available: https://www.smartvrlab.nl/3dof-vs-6dof-in-vr/. [Acessed: 2021-1-2].

[24] Mun Wai Lee and Ram Nevatia. Body part detection for human pose estimation and tracking. In *2007 IEEE Workshop on Motion and Video Computing (WMVC’07)*, pages 23–23. IEEE, 2007.

[25] Google Github. ”MediaPipe Face Detection”, 2023. [Online]. Available: https://google.github.io/mediapipe/solutions/face_detection, [Acessed: 2022-6-10].

[26] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. Blazeface: Sub-millisecond neural face detection on mobile gpus. *arXiv preprint arXiv:1907.05047*, 2019.

[27] Google Github. ”MediaPipe Hands”, 2023. [Online]. Available: https://google.github.io/mediapipe/solutions/hands, [Acessed: 2022-6-20].

[28] Google Research. ”On-device, Real-time Body Pose Tracking with MediaPipe BlazePose”, 2023. [Online]. Available: https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html, [Acessed: 2022-4-14].

[29] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. Blazepose: On-device real-time body pose tracking. *arXiv preprint arXiv:2006.10204*, 2020.

[30] Google Github. ”MediaPipe FaceMesh”, 2023. [Online]. Available: https://google.github.io/mediapipe/solutions/face_mesh, [Acessed: 2022-6-27].

[31] Ivan Grishchenko, Artsiom Ablavatski, Yury Kartynnik, Karthik Raveendran, and Matthias Grundmann. Attention mesh: High-fidelity face mesh prediction in real-time. *arXiv preprint arXiv:2006.10962*, 2020.

[32] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[33] Intel RealSense. ”Intel® RealSenseTM Product Family D400 Series Datasheet”, 2023. [Online]. Available: https://tinyurl.com/3vk9fsfa, [Acessed: 2022-9-2].

[34] Google Github. ”MediaPipe Hands”, 2023. [Online]. Available: https://tinyurl.com/z6yensaw, [Acessed: 2022-8-22].

[35] Xiao Xin Lu. A review of solutions for perspective-n-point problem in camera pose estimation. In *Journal of Physics: Conference Series*, volume 1087, page 052009. IOP Publishing, 2018.

[36] CoppeliaSim. "CoppeliaSim", 2021. [Online]. Available: https://www.coppeliarobotics.com/, [Acessed: 2022-2-15].

[37] Aldebaran. "NAO Software 1.14.5 documentation", 2022. [Online]. Available: http://doc.aldebaran.com/1-14/family/robots/joints_robot.html, [Acessed: 2022-3-12].

[38] Hafiz Syed Fawad Adil. "Anthropomorphic Robotic Arm Development, Control and Manipulation for Service Robots". Mechatronics, Robotics and Automation Engineering Master Thesis, University of Minho, 2022.

[39] Inmoov. "https://inmoov.fr/", 2022. [Online]. Available: https://inmoov.fr/, [Acessed: 2022-11-10].

[40] Laboratory of Automation and Robotics. "LAR@Home Qualification Video RoboCup@Home 2023 OPL - CHARMIE", 2022. [Online]. Available: https://www.youtube.com/watch?v=gNx9OYljlcw. [Acessed: 2021-12-22].