FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Development of an Augmented Reality System for Interaction and Assistance in Industrial Processes

Tiago Alexandre Moreira Ribeiro

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Dr. Vítor Hugo Pinto

July 29, 2025

Resumo

Esta dissertação incide sobre o desenvolvimento de uma aplicação de realidade aumentada (RA) destinada a potenciar a interação entre humanos e robôs em ambientes industriais. A aplicação foi concebida para utilização com os Microsoft HoloLens 2 e o braço robótico UR3e da Universal Robots, integrando tecnologias de RA e robótica com o objetivo de otimizar processos industriais, formação de operadores e eficiência das tarefas.

A aplicação foi desenvolvida para permitir a monitorização em tempo real do sistema robótico, através de um *Digital Twin (DT)* do ambiente físico. Este *DT* fornece uma representação gráfica do estado do robô, permitindo aos utilizadores monitorizar os movimentos do braço e do *gripper*, bem como interagir com o sistema através de gestos manuais e rastreamento ocular. A interface foi desenhada para ser intuitiva, visando facilitar a operação e a formação em contextos industriais e educacionais.

Durante o processo de desenvolvimento, a aplicação foi sujeita a várias fases de testes de modo a garantir a sua funcionalidade em diferentes cenários operacionais. Os testes abrangeram tarefas industriais típicas, tais como manipulação de objetos, comandos de movimento do robô e visualização de informações do estado. Embora os testes tenham demonstrado que a aplicação cumpriu a maioria dos seus objetivos, foi identificada uma limitação ao nível do desempenho dos HoloLens 2, em particular durante a gravação de vídeo, o que provocou alguma latência na execução do sistema.

Apesar desta limitação, a dissertação destaca a contribuição da realidade aumentada no domínio da robótica, oferecendo um sistema que pode ser utilizado para melhorar a segurança, a produtividade e a formação de operadores em ambientes industriais. As limitações de desempenho identificadas abrem caminho para futuras melhorias, incluindo a otimização da aplicação e a exploração de novas plataformas de RA com maior capacidade de processamento. Este trabalho sugere ainda a integração com plataformas de simulação, como o Gazebo, para uma validação mais detalhada das interações entre o robô e o ambiente antes da implementação em contexto real, bem como a expansão da aplicação para suportar múltiplos robôs e cenários de trabalho mais complexos.

Palavras-chave: Realidade Aumentada (RA), Interação Humano-Robô, Automação Industrial, Robótica Colaborativa, Digital Twin, Visualização em Tempo Real

Abstract

This dissertation focuses on the development of an augmented reality (AR) application aimed at enhancing human-robot interaction in industrial environments. The application was designed for use with the Microsoft HoloLens 2 augmented reality headset and the UR3e robotic arm from Universal Robots, integrating AR and robotics technologies to optimize industrial processes, operator training, and task efficiency.

The application was developed to allow real-time visualisation of the robotic system through a digital twin of the physical environment. This digital twin provides a graphical representation of the robot's state, allowing users to monitor arm and gripper movements and interact with the system via hand gestures and eye tracking. The interface was designed to be intuitive, with the goal of facilitating operation and training in industrial and educational contexts.

During the development process, the application underwent several testing phases to ensure its functionality in different operational scenarios. The tests involved typical industrial tasks, such as object manipulation, robot movement commands, and state information visualisation. Although the tests demonstrated that the application met most of its objectives, a performance limitation was identified with the HoloLens 2, particularly during video recording, which caused some latency in system execution.

Despite this, the dissertation highlights the contribution of augmented reality in the field of robotics, offering a system that can be used to improve safety, productivity, and operator training in industrial environments. The identified performance limitations pave the way for future improvements, including optimization of the application and the exploration of new AR platforms with greater processing power. This work also suggests integration with simulation platforms such as Gazebo for more detailed validation of robot-environment interactions before real-world implementation, as well as expanding the application to support multiple robots and more complex work scenarios.

Keywords: Augmented Reality (AR), Human-Robot Interaction, Industrial Automation, Collaborative Robotics, Digital Twin, Real-time Visualisation

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to promote a more inclusive, safe, and sustainable future. This dissertation supports several of these goals by enhancing industrial productivity, promoting innovation, fostering safe work environments, and reducing the environmental footprint of training and development through digital solutions [1].

The specific Sustainable Development Goals relevant to this work are:

SDG 4 Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all

SDG 8 Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 12 Ensure sustainable consumption and production patterns

SDG 13 Take urgent action to combat climate change and its impacts

SDG 17 Strengthen the means of implementation and revitalize the global partnership for sustainable development

SDG	Target	Contribution	Performance Indicators and Metrics
4	4.4	The system supports technical and vocational training in robotics and AR, fostering future-ready skills.	Number of students or trainees using the AR system; improvements in test scores or training time.
	4.7	Promotes knowledge and skills for sustainable development through hands-on, immersive learning.	Integration of sustainability concepts in industrial training modules.
8	8.2	AR-robotics integration increases manufacturing productivity and reduces error rates.	Task completion time, reduced downtime, and improved operator performance.
	8.6	Immersive AR training encourages youth engagement in industrial and engineering careers.	Number of young professionals trained; retention and feedback data.
9	9.4	Enhancing industrial workflows using AR and digital twins reduces waste and increases efficiency.	Reduced material waste, shorter prototyping cycles, and increased process automation.
	9.5	Supports applied research and innovation in robotics and AR fields.	Number of novel features developed; interdisciplinary integration (e.g., AI, simulation).
12	12.6	Use of virtual tools for training and testing minimizes material usage and promotes sustainable practices.	Fewer physical mockups required; increased reliance on simulations and digital workflows.
13	13.3	Promotes climate awareness by reducing the energy and materials consumed in traditional industrial training.	Carbon footprint reduction by substituting physical resources with virtual alternatives.
17	17.17	Collaboration between academia and industry (GrowSkills) fosters sustainable innovation ecosystems.	Documentation of partnership results; contributions to open-source tools or shared best practices.

Acknowledgements

I would like to express my sincere appreciation to my supervisor, Prof. Vitor Hugo Pinto, for his guidance and for allowing me the freedom to bring my ideas to life throughout this project.

I am also grateful to GrowSkills and all its members, especially the engineer Pedro Festas, for his patience and assistance whenever possible during the process.

I want to extend my heartfelt thanks to my family for their unwavering support, and in particular to my girlfriend Marta, whose encouragement has been invaluable.

Finally, I am deeply grateful for the opportunity to develop this project, which closely aligns with my future career aspirations. This experience has been both inspiring and motivating as I continue to pursue my professional goals.

Tiago Ribeiro

""Augmented reality is going to change the way we use technology forever." "

Tim Cook

Contents

List of Figures

List of Tables

List of Acronyms

3D	Three-dimensional
6DoF	6 degrees of freedom
AR	Augmented Reality
C#	C Sharp
CAD	Computer-Aided Design
Cobots	Collaborative Robots
DT	Digital Twin
GUI	Graphical User Interface
IMU	Inertial Measurement Unit
IoT	Internet of Things
MR	Mixed Reality
MRTK	Microsoft Mixed Reality Toolkit
SDK	Software Development Kit
TCP	Tool Center Point
TCP/IP	Transmission Control Protocol / Internet Protocol
UDP	User Datagram Protocol
UI	User Interface
UR	Universal Robots
XR	Extended Reality

Chapter 1

Introduction

This chapter introduces the context, motivation, and structure of the dissertation. First, it discusses the industrial relevance and technological foundations behind the proposed AR-based system. Then, it presents the personal and professional motivations that led to the development of this work. Finally, it outlines the structure of the entire dissertation to guide the reader through the upcoming chapters.

1.1 Context

The continuous evolution of industrial automation and smart manufacturing demands advanced tools that enhance operator efficiency, reduce errors, and facilitate seamless interaction with complex machinery. In this context, AR technologies have emerged as a transformative solution by overlaying digital information directly onto the physical environment, thereby enabling intuitive and real-time assistance during industrial processes.

This dissertation focuses on the development of an AR system designed specifically for interaction and assistance in industrial workflows. The system aims to provide operators with contextual information, guided instructions, and remote support capabilities to optimize task execution and maintenance activities. To ground this work in practical applications, the GrowSkills Educational Workstation, featuring the Universal Robots (UR) Didactic Kit, is employed as the testbed platform.

The GrowSkills Educational Workstation is a modular and versatile training environment widely adopted in academic and industrial settings to simulate real-world robotic applications. It integrates collaborative robotic arms from Universal Robots, known for their flexibility, precision and ease of programming. The UR Didactic Kit within this workstation provides a hands-on environment for learning, experimentation, and prototyping industrial automation tasks.

By leveraging the capabilities of the UR robotic arms, the dissertation explores how an AR system can enhance operator interaction through immersive visualisation and gesture-based controls. The integration of AR into this educational and industrial platform enables a unique convergence of human factors engineering, robotics, and mixed reality technologies, promoting safer, more efficient, and user-friendly industrial processes. The overarching objective of this research is to design, implement, and evaluate an AR-based assistance system that not only supports operators in performing complex tasks but also fosters a collaborative and adaptive working environment. This involves addressing challenges related to system usability, real-time data integration, and robustness in industrial scenarios.

Through this work, the dissertation contributes to the growing body of knowledge on AR applications in industry, demonstrating practical benefits and providing a foundation for further innovation in augmented reality-driven industrial automation.

1.2 Motivation

The motivation behind this dissertation arises from an interest in collaborative robotics and emerging industrial technologies. Exposure to advanced industrial systems, particularly during a robotics fair featuring real-world applications, revealed the potential for developing innovative technological solutions. This experience underscored the significant value of AR in enhancing efficiency, training, and process visualisation within manufacturing environments.

This dissertation, developed in collaboration with an active industry partner, aims to explore the capabilities of AR to modernize production systems, fostering advances that are valuable both to the scientific community and to the industrial sector. This work aligns closely with professional aspirations, offering an opportunity for the application of technical knowledge toward impactful, practical, and research-driven solutions.

1.3 Objectives

The main objective of this dissertation is to design and implement an AR system that enhances human-robot interaction in industrial environments. To achieve this goal, the following specific objectives were defined:

- Integrate AR capabilities with Microsoft HoloLens 2 and the UR3e robotic arm to create a functional mixed reality system.
- Enable real-time monitoring of the robotic system by developing a Digital Twin that accurately reflects the physical environment and robot state.
- Implement intuitive user interaction mechanisms using hand gestures and eye tracking to allow natural and efficient control of the robotic system.

- Optimize industrial workflows by improving operator training, enhancing task efficiency, and reducing operational errors through AR-based assistance.

These objectives guided the design, implementation, and evaluation phases of the project, ensuring alignment with both academic research goals and industrial relevance.

1.4 Dissertation Structure

The remainder of this dissertation is organized to provide a comprehensive understanding of the research and development process. Chapter 2 presents a review of the state of the art, focusing on Industry 4.0 technologies, augmented reality, collaborative robotics, and the relevant software and hardware platforms. Chapter 3 outlines the methodology, detailing the design decisions, selected tools, and system architecture. It also describes the implementation phase, including the configuration of the robot, development of the AR interface, creation of the digital twin, and overall system integration. This chapter also includes the system evaluation, with testing procedures and performance analysis. Finally, Chapter 4 concludes the dissertation by summarizing the main contributions, acknowledging limitations, and proposing directions for future work. For a visual demonstration of the developed application and access to the source code, refer to Appendix A. Additionally, a comprehensive user guide for the application is provided in Appendix B.

Chapter 2

Background

This chapter provides a comprehensive overview of the foundational technologies and concepts that support the development of augmented reality systems for industrial applications. It examines the principles of Industry 4.0 and how digital transformation is reshaping manufacturing, while exploring the transformative role of AR within this evolving industrial landscape. It also discusses the evolution and capabilities of AR, distinguishes between traditional and collaborative robotics, and highlights key hardware platforms such as the Microsoft HoloLens 2. Additionally, it covers essential (SDKs), game engines vital for AR application development, the concept of Digital Twins and their associated 3D models, as well as communication protocols like TCP/IP that enable seamless interaction between physical and virtual systems.

2.1 Industry 4.0

Industry 4.0 marks the fourth major shift in manufacturing, often referred to as the "digital transformation" of industry. It involves integrating advanced digital technologies such as cyber-physical systems, the Internet of Things (IoT) and artificial intelligence directly into production processes. This shift enables the development of "smart factories" where realtime data is collected, analyzed and used to make faster and more accurate decisions [2]. The core pillars of Industry 4.0 solutions are visually represented in Figure 2.1.

Smart factories leverage interconnected machinery, real-time data streams and automation to optimize production, adapt swiftly to market demands and reduce operational inefficiencies. One of the primary goals of Industry 4.0 is to foster a more agile, responsive manufacturing environment. Data analytics, for example, can be used to monitor machine performance and predict maintenance needs before issues arise, which minimizes downtime and enhances productivity [3].

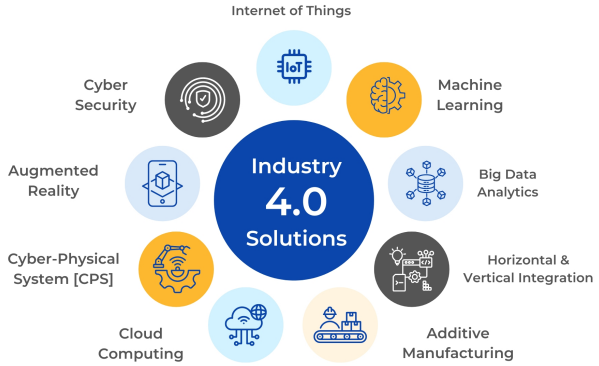


Figure 2.1: Industry 4.0.
[4]

As Industry 4.0 reshapes industrial practices, the need for systems capable of working seamlessly with existing and emerging technologies becomes crucial. This interconnected environment promotes collaboration among machines, systems and workers, allowing all parts to operate together efficiently.

2.2 Augmented Reality

AR is a technology that enhances a user's perception of the real world by overlaying virtual information in real time. According to Carmigniani *et al.* [5], AR combines three essential elements: integration of real and virtual objects, interaction in real time, and registration within a 3D environment. Unlike Virtual Reality (VR), which immerses users in a fully synthetic environment, AR maintains a real-world presence, enhancing it with digital elements rather than replacing it entirely.

The conceptual framework for AR is commonly understood through Milgram and Kishino's "Reality–Virtuality Continuum," where AR lies closer to the real world, and Augmented Virtuality (AV) approaches full virtual environments. This continuum, as illustrated in Figure 2.2, captures the hybrid nature of AR systems, which exist between purely physical and purely virtual realities [6].

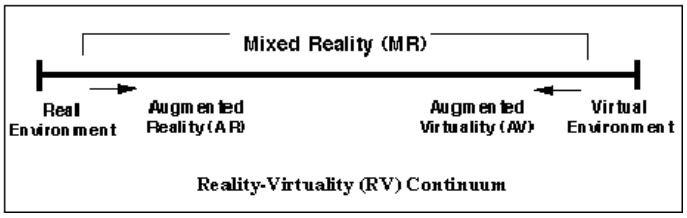


Figure 2.2: Milgram's Reality-Virtuality Continuum.
[5]

The evolution of AR has spanned several decades, beginning with early concepts like the Sensorama in the 1950s, and moving through significant milestones such as Sutherland's head-mounted display in the 1960s, and the Virtual Fixtures system developed by Rosenberg in the 1990s. The term "Augmented Reality" itself was coined at Boeing by Caudell and Mizell as they sought ways to assist workers with wiring aircraft [5, 6].

AR technologies encompass a broad array of systems, including computer vision algorithms, 3D tracking, display hardware like head-mounted displays (HMDs), and mobile interfaces. These systems enable applications in diverse fields, from industrial maintenance and medical visualisation to gaming and tourism [7].

Notably, AR is not limited to visual augmentation. Researchers have explored multi-sensory augmentation, including auditory, haptic, and even olfactory overlays, to support users with sensory impairments or to enhance interaction.

Carmigniani *et al.* [5] also outline the challenges AR must overcome to transition from research to widespread industrial use. These include improving the accuracy of object tracking, achieving real-time performance on mobile platforms and creating intuitive user interfaces.

Role of Augmented Reality in Industry 4.0

AR has emerged as a transformative tool in Industry 4.0 by providing a visual bridge between digital data and the physical world. AR overlays digital information, such as data visualisations or instructions, onto a worker's field of vision, allowing them to understand and act on complex information without needing to look away from their tasks. This approach enhances safety, productivity and efficiency across various applications [8].

For example, in industrial maintenance, AR can guide technicians by displaying step by step instructions directly on equipment. Instead of consulting a manual, workers receive real-time, contextual information that helps them troubleshoot and resolve issues faster. This results in reduced downtime as technicians have immediate access to the exact information needed to perform repairs or maintenance accurately. By supporting better task execution, AR decreases errors and accelerates task completion, ultimately enhancing operational efficiency [9].

In training scenarios, AR can be particularly effective by reducing the time it takes for new workers to learn complex tasks. Visual aids overlaid in real-time help trainees understand each step of a process intuitively, which improves retention and reduces the likelihood of mistakes.

2.3 Traditional Robots vs Collaborative Robots

The evolution of industrial automation has introduced two primary categories of robotic systems: Collaborative Robots and Traditional Industrial Robots. While both play pivotal roles in enhancing manufacturing processes, they differ significantly in design, functionality, and application suitability.

This section delves into a comparative analysis of these two robotic paradigms, examining key factors such as safety protocols, flexibility, ease of integration, precision, and cost-effectiveness.

Traditional Robots

Traditional industrial robots are characterized by their exceptional efficiency and accuracy, making them well-suited for high-speed, repetitive operations. They are particularly effective in processes such as welding, painting, and palletizing where consistent performance and durability are critical. These robots are engineered to withstand harsh industrial conditions, ensuring reliable operation in demanding manufacturing environments. Within our automation offerings, such traditional robotic systems, are customized to align with the unique requirements of each client [10, 11].

Collaborative Robots

Collaborative Robots, commonly known as Cobots, can basically perform the same tasks as an industrial robot, only they are a bit smaller and lighter. These robots are specifically engineered to operate safely in close proximity to human workers. Their versatility and adaptability allow them to undertake various tasks, including assembly, material handling and quality inspection. With user-friendly programming and operation, they present an excellent solution for companies aiming to enhance the efficiency of their manufacturing workflows [12–14].

2.4 Differences Between Traditional Robots and Collaborative Robots

Collaborative robots and traditional robots serve distinct roles within modern manufacturing environments, each offering specific advantages and facing particular limitations. The critical differences, outlined in Table 2.1, lie in safety, flexibility, installation, speed, payload capacity, and ideal use cases. Figure 2.3 provides a visual representation of a collaborative robot alongside a traditional industrial robot.

Feature	**Collaborative Robots**	**Traditional Robots**
Safety	Operate safely alongside humans, no barriers	Require safety barriers and guarding
Flexibility	Highly adaptable, easy to program	Less flexible, designed for repetitive tasks
Installation	Simple installation, low training required	Complex installation, specialized expertise needed
Speed and Payload	Lower speed and payload capacity	High speed and payload capacity
Ideal Use Cases	Low-volume, frequent task changes	High-volume, repetitive, hazardous tasks
Cost	Lower upfront investment	Higher upfront investment
Performance	Good for varied tasks, moderate precision	Superior precision and durability
Workplace Impact	Enhances human-robot collaboration	Replaces humans in hazardous or monotonous jobs

Table 2.1: Comparison between Collaborative Robots and Traditional Robots.

Cobots are engineered to operate safely alongside human workers without the need for expensive protective barriers, thanks to their advanced sensor systems and integrated safety features. This capability enhances workplace safety and allows for more flexible deployment in dynamic settings. Furthermore, cobots are highly adaptable and easy to program, making them ideal for low-volume production scenarios that require frequent task changes and diverse operations. Their simplified installation and user-friendly programming reduce training demands and associated service costs, enabling quicker integration into existing workflows.

In contrast, traditional robots excel in high-speed, repetitive tasks that demand exceptional precision and durability. Their robustness allows for consistent product quality and reliability even under continuous operation in harsh industrial conditions. These robots are particularly suitable for hazardous or monotonous tasks, effectively improving worker safety by handling dangerous operations.

However, traditional robots often require significant upfront investment, both in terms of acquisition and complex installation procedures, which frequently necessitate specialized expertise and modifications to the workspace. Additionally, ensuring operator safety involves costly physical barriers, adding to the overall expense and inflexibility of deployment. While collaborative robots prioritize safety, flexibility, and ease of use, they generally offer lower speed and payload capacity compared to traditional robots. Consequently, cobots may be less appropriate for tasks involving heavy lifting or rapid cycles. On the other hand, traditional robots provide superior performance and precision but at the cost of reduced adaptability and higher implementation complexity. Understanding these differences is essential for manufacturers seeking to optimize automation strategies tailored to their specific operational needs [10, 15].

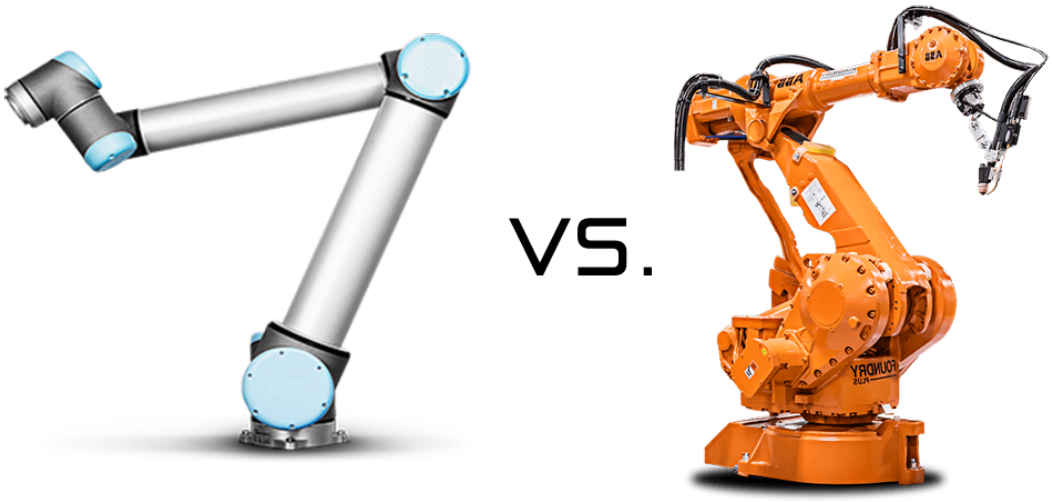


Figure 2.3: Cobots vs Traditional Robots.
[16]

2.5 Universal Robots

Universal Robots, a Danish company founded in 2005, is a pioneer in collaborative robots designed to work safely alongside humans without the need for safety cages. Their product lineup includes various models tailored for different industrial applications.

This section provides a detailed comparison of the various Universal Robots models, highlighting their respective strengths and limitations.

Collaborative Robots from Universal Robots

Collaborative robots, developed by Universal Robots, are transforming manufacturing by enabling flexible automation solutions across a wide range of industrial sectors. These robotic systems are designed to perform tasks such as assembly, painting, palletizing, screw-driving, packaging, polishing, injection molding, and welding, among others. Their adaptability and ease of integration make them particularly valuable for manufacturers of varying scales, helping to enhance productivity and maintain competitiveness in the global market [17].

Universal Robots offers two main series of collaborative robots (cobots):

- **e-Series**: Known for reliability and versatility, suitable for a wide range of applications.
- **UR Series**: High-performance cobots designed for demanding tasks requiring higher payloads and longer reach.

Model Overview of the available models

The comparative table 2.2, based on the Universal Robots collective data sheet, offers a comprehensive technical overview of the company's collaborative robot portfolio, including the UR3e, UR5e, UR10e, UR16e, UR20, and the more recent UR30 model. It outlines key specifications such as payload capacity (ranging from 3 to 30 kg), reach (500 to 1750 mm), repeatability (±0.03 mm to ±0.05 mm), joint speed, power consumption, IP ratings, and other environmental performance parameters. Figure 2.4 illustrates all UR models.

Specification	**UR3e**	**UR5e**	**UR10e**	**UR16e**	**UR20**	**UR30**
Payload (kg)	3	5	12.5	16	20	30
Reach (mm)	500	850	1300	900	1750	1300
Weight (kg)	11.2	20.6	33.5	33.1	64	63.5
Footprint (mm)	128	149	190	190	245	245
Repeatability (mm)	±0.03	±0.03	±0.05	±0.05	±0.05	±0.05
Degrees of Freedom	6	6	6	6	6	6
IP Rating	IP54	IP54	IP54	IP54	IP65	IP65
Power Consumption (Typical)	100 W	200 W	350 W	350 W	500 W	500 W
Cleanroom Certification	ISO Class 5	ISO Class 5	ISO Class 5	ISO Class 5	N/A	N/A

Table 2.2: Comparison of Universal Robots Collaborative Robot Models.

All models feature six degrees of freedom and integrated force/torque sensors. The UR Series, particularly the UR20 and UR30, exhibits notable improvements in reach, payload, and motion speed, addressing the demands of more complex industrial tasks. This comparison serves as a valuable reference for selecting the most suitable robotic arm according to specific application and operational requirements [18].

Figure 2.4: Robot models.
[19]

2.6 Robotiq Grippers

Robotiq's grippers, including the 2F-85, 2F-140 and Hand-E models, offer a range of solutions tailored to different industrial needs. Their emphasis on adaptability, precision, and ease of integration makes them a strong choice for collaborative robotics applications. While other brands may excel in specific areas such as gripping force or payload capacity, Robotiq's comprehensive

approach to gripper design ensures that users can find a solution that meets their specific requirements [20, 21].

Robotiq Gripper Models: Comparative Analysis

2F-85 and 2F-140 Adaptive Grippers

The 2F-85 and 2F-140, shown in Figure 2.5, are rotational adaptive grippers designed for versatility in handling objects of various shapes and sizes. Their unique parallel linkage mechanism allows the fingertips to remain parallel while closing, enabling an encompassing grip. These grippers are ideal for applications requiring flexibility and adaptability.

Figure 2.5: 2F-85 and 2F-140 Grippers.
[22]

Advantages

- **Versatility:** Capable of handling a wide range of object shapes without the need for custom fingertips.
- **High Gripping Force:** The parallel linkage design provides a higher gripping force compared to some other adaptive grippers that use flexible materials.
- **Integrated Control:** Offers control over position, force, and speed, allowing for precise handling of delicate items.

Limitations

- **Size Constraints:** The gripper's design may not be suitable for extremely small or large objects without additional customization.

- **Complexity:** The encompassing grip mode may not be necessary for all applications, potentially adding unnecessary complexity.

Hand-E Gripper

The Hand-E, depicted in Figure 2.6, is a translational parallel gripper designed for precision tasks in high-mix, low-volume environments. Its 50-mm stroke and sealed design make it suitable for applications such as assembly and CNC machining [20, 21].

This gripper is employed in the GrowSkills workbench due to its precise handling capabilities and ease of integration, making it ideal for educational and prototyping tasks involving diverse small to medium-sized components. Its robust, sealed construction ensures durability for repeated use, while the plug-and-play design simplifies setup for users with varying levels of robotics experience.

Figure 2.6: Hand-E Gripper.
[21]

Advantages

- **Precision:** The parallel stroke and high accuracy make it ideal for tasks requiring meticulous handling.
- **Durability:** The sealed design ensures reliability in harsh manufacturing conditions.
- **Ease of Integration:** Features plug-and-play installation and intuitive programming, reducing setup time.

Limitations

- **Limited Stroke:** The 50-mm stroke may limit its ability to handle larger objects without additional customization.
- **Payload Capacity:** With a maximum payload of 7 kg, it may not be suitable for heavy-duty applications.

2.7 Microsoft HoloLens 2

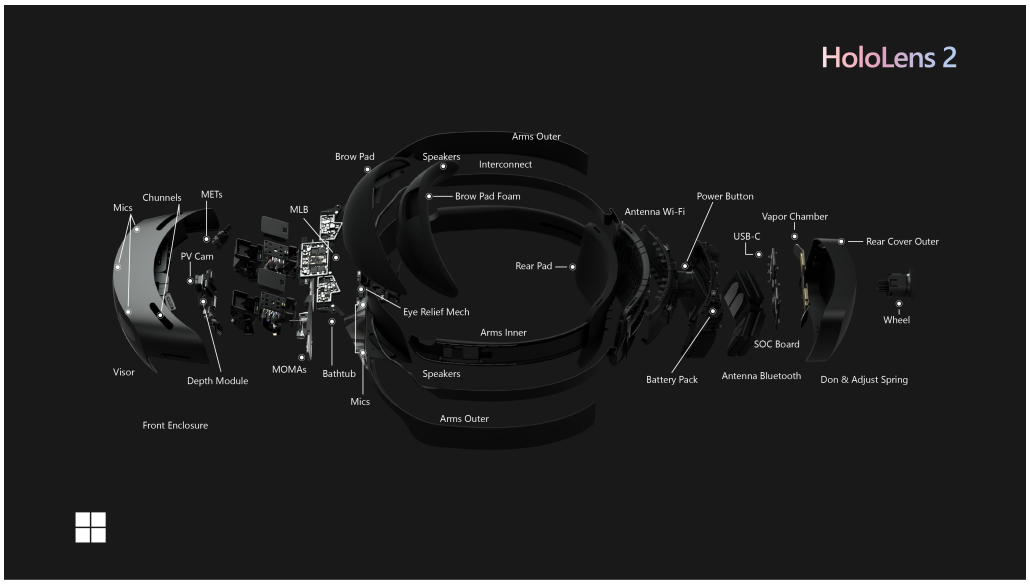


Figure 2.7: Exploded view of Microsoft HoloLens 2, showcasing internal components. [23]

HoloLens 2 is an untethered, self-contained holographic computer running Windows Holographic (based on Windows 10). It improves comfort, immersion and collaborative capabilities over its predecessor [23]. As shown in the exploded view in Figure 2.7, its internal components are meticulously integrated.

This section provides a detailed analysis of the Microsoft HoloLens 2, focusing exclusively on the hardware specifications and functionalities outlined in Microsoft's official documentation. The HoloLens 2 is a cutting-edge AR device designed to deliver an immersive, ergonomic, and highly functional experience, particularly well-suited for industrial and professional environments. Its design integrates advanced sensors, a sophisticated optical system, and a powerful computational core, making it an ideal platform for supporting complex industrial processes.

In addition, to situate the HoloLens 2 within the broader AR market, I compare it with other leading AR headsets, examining their strengths and drawbacks in terms of display quality, field of view, comfort, tracking, and ecosystem integration.

Hardware Components and Ergonomics

The hardware and ergonomic design of the device are carefully engineered to ensure user comfort and functionality, as detailed in Table 2.3. The visor and headband house all critical sensors and display components, with the visor designed to flip up easily. The adjustable wheel on the headband allows for a secure and comfortable fit, accommodating users who wear eyeglasses.

Physical controls are conveniently positioned on either temple for brightness and volume adjustments, while power and charging ports are accessible at the back. Included accessories like interchangeable brow pads, an optional overhead strap, USB-C charger, and a microfiber cloth enhance the user experience by providing comfort, extended wearability, and proper maintenance options [23].

Component	**Details**
Visor & Headband	Houses sensors and displays; visor can flip up. Adjustable wheel on headband for secure, comfortable fit (including over eyeglasses).
Controls	Brightness buttons on left temple, volume on right, power and USB-C port on back.
In-Box Accessories	Changeable brow pad, optional overhead strap for extended wear, USB-C charger (18W/9V 2A), cable, microfiber cloth.

Table 2.3: Hardware Components and Ergonomics Overview.

Display and Optics

The device employs advanced waveguide lenses that provide see-through holographic optical displays, enabling an immersive augmented reality experience. The visual system consists of dual 2K light engines with a 3:2 aspect ratio and a radiant density exceeding 2.5K, delivering sharp and vibrant imagery, as outlined in Table 2.4. Additionally, eye-based rendering technology dynamically adjusts the projected images according to the user's eye position, ensuring optimal clarity and reducing visual fatigue by personalizing the display output in real time [23].

Component	**Details**
Waveguide Lenses	See-through holographic optical displays for immersive mixed reality experience.
Resolution	Dual 2K light engines (3:2 aspect ratio) with over 2.5K radiants density for sharp and vibrant visuals.
Eye-Based Rendering	Dynamically adjusts the projection per eye position to optimize clarity and reduce fatigue.

Table 2.4: Display and Optics Specifications.

Sensors and Tracking

To facilitate precise spatial awareness and user interaction, the device integrates a comprehensive suite of sensors, summarized in Table 2.5. Four visible-light cameras enable accurate head tracking with a wide field of view, while two infrared cameras provide real-time eye tracking for intuitive control and focus adjustment. Depth sensing is achieved through a 1-megapixel Time-of-Flight sensor that captures spatial data for environmental mapping. An inertial measurement unit

(IMU) consisting of accelerometer, gyroscope, and magnetometer sensors tracks motion and orientation. An 8-megapixel RGB camera supports still image capture and video recording at 1080p resolution, enabling mixed reality capture and sharing [23].

Sensor Type	Details
Head Tracking	Four visible-light cameras (front-facing stereo, 98.6 mm baseline, 96.1° diagonal FOV).
Eye Tracking	Two infrared (IR) cameras enabling real-time eye monitoring.
Depth Sensing	1 MP Time-of-Flight depth sensor for spatial awareness and environment mapping.
IMU Suite	Includes accelerometer, gyroscope, and magnetometer for motion and orientation tracking.
RGB Camera	8 MP resolution; supports 1080p video at 30 fps for mixed-reality capture.

Table 2.5: Sensor and Tracking Capabilities.

Audio and Speech

The audio subsystem is designed to deliver clear and spatially accurate sound, enhancing both communication and immersion. As detailed in Table 2.6, a five-channel microphone array captures voice commands and environmental sounds with precision, facilitating natural speech recognition and noise cancellation. The built-in spatial audio speakers produce immersive 3D soundscapes that anchor virtual audio elements within the user's real-world surroundings, enriching the mixed reality experience [23].

Component	Details
Microphones	Five-channel microphone array for voice input and environmental sound capture.
Speakers	Built-in spatial audio speakers providing immersive 3D sound.

Table 2.6: Audio and Speech Interaction Features.

Compute and Connectivity

At its core, the device is powered by a Qualcomm Snapdragon 850 processor augmented with a second-generation custom Holographic Processing Unit (HPU), optimized for mixed reality workloads. It includes 4 GB of LPDDR4x RAM and 64 GB of UFS 2.1 storage, balancing performance and capacity for applications and data as presented in Table 2.7. Wireless connectivity supports fast Wi-Fi 802.11ac with 2×2 MIMO and Bluetooth 5.0 for peripheral devices. A versatile USB-C port enables data transfer, charging, and accessory connection, ensuring compatibility and convenience [23].

Component	Details
SoC	Qualcomm Snapdragon 850 with custom 2nd-generation Holographic Processing Unit (HPU).
Memory	4 GB LPDDR4x RAM.
Storage	64 GB UFS 2.1 internal storage.
Wireless	Wi-Fi 802.11ac (2×2 MIMO), Bluetooth 5.0.
Port	USB-C Dual Role Power (DRP) port for charging and connectivity.

Table 2.7: Computing Hardware and Connectivity Options.

Power and Form Factor

Battery life and device weight are critical for usability in extended scenarios. The device utilizes a lithium battery capable of approximately 2 to 3 hours of active use and up to two weeks on standby, providing flexibility for daily activities. It supports full functionality while charging and requires a minimum of 15 watts to maintain battery health. The design emphasizes passive cooling with a fanless architecture, minimizing noise and maintenance, as summarized in Table 2.8. Weighing 566 grams and featuring a single adjustable band, the device balances robustness with wearer comfort, accommodating glasses without discomfort [23].

Specification	Details
Battery	Lithium battery with approximately 2–3 hours of active use and up to 2 weeks standby.
Charging	Device is fully functional while charging; requires $\geq$ 15 W for battery maintenance.
Cooling	Fanless passive cooling system for silent operation.
Weight	566 g.
Fit	Single adjustable band, compatible with eyeglasses.

Table 2.8: Power, Cooling, and Form Factor Specifications.

Capabilities

The device's core capabilities enable rich and natural interaction with the augmented environment. As outlined in Table 2.9, human understanding is facilitated through fully articulated two-hand tracking, allowing users to manipulate holograms intuitively. Real-time eye tracking enhances interface responsiveness and context awareness. Voice control is supported both on-device and through integration with Cortana, enabling hands-free command and control. Environmental awareness includes six degrees of freedom (6 DoF) spatial tracking and world-scale positional awareness for accurate mapping and navigation. Real-time spatial mapping enables dynamic interaction with surroundings, while mixed-reality capture allows users to record photos and videos combining holograms with the real world [23].

Capability Area	Features
Human Understanding	Two-hand tracking, real-time eye tracking, voice control with on-device and Cortana support.
Environmental Awareness	6 DoF spatial tracking, real-time spatial mapping, and mixed-reality capture.

Table 2.9: Interactive and Environmental Capabilities.

Pre-Installed Software

The device runs on Windows Holographic OS, a tailored operating system designed for augmented reality applications. It comes preloaded with a comprehensive suite of Microsoft apps, as shown in Table 2.10, including productivity tools like 3D Viewer, Mail and Calendar, and File Explorer; communication assistants like Cortana; enterprise solutions such as Dynamics 365 Guides and Remote Assist; as well as media and utility apps like Photos, Movies and TV, Edge browser, and OneDrive cloud storage. This software ecosystem supports a wide range of use cases out-of-the-box, enabling users to be productive immediately [23].

Category	Applications Included
Operating System	Windows Holographic OS.
Productivity	3D Viewer, Cortana, Dynamics 365 Guides and Remote Assist, Mail & Calendar, File Explorer.
Media and Utilities	Edge, Store, OneDrive, Photos, Movies & TV, Feedback Hub, Settings, Tips.

Table 2.10: Pre-Installed Software and Applications.

Safety, Compliance and Packaging

To ensure user safety and regulatory compliance, the device meets multiple eye-protection standards including ANSI Z87.1, CSA Z94.3, and EN 166, as detailed in Table 2.11. The packaging is designed to protect the unit during shipping and handling, with the retail box dimensions approximately 379 × 248 × 163 mm and weight around 2.9 kg. The shipper box is slightly larger and heavier to accommodate multiple units or additional packaging materials. The product is covered by a standard limited warranty and includes a final-sale policy. It is recommended for users aged 13 and older, reflecting safety and usability considerations for different age groups [23].

Aspect	**Details**
Certifications	Meets ANSI Z87.1, CSA Z94.3, and EN 166 eye-protection standards.
Box Dimensions	Unit: 379 × 248 × 163 mm, 2.9 kg; Shipper: 446 × 258 × 172 mm, 3.3 kg.
Warranty	Standard limited warranty; final-sale policy.
User Age Recommendation	Intended for users aged 13 and above.

Table 2.11: Safety Certifications, Packaging, and User Requirements.

Why to choose HoloLens 2 vs Other Headsets

Feature	**HoloLens 2**	**Meta Quest 3**	**Apple Vision Pro**
Release Date	2019	2023	2024
Type	AR / Mixed Reality	VR with passthrough (Mixed Reality)	XR (Full Mixed Reality with high-end AR)
Field of View	∼52° diagonal	∼110° horizontal	∼100° (estimated), ultra-high-res
Display	Waveguide (2K per eye)	LCD (2064×2208 per eye)	Micro-OLED (23M total pixels)
Refresh Rate	60 Hz	90–120 Hz	90–100 Hz
Input / Control	Hand, eye, voice	Hand, controllers, voice	Hand, eye, voice
Tracking	Inside-out (6DoF, hand + eye tracking)	Inside-out, hand tracking, depth sensor	Inside-out with spatial sensors
Weight	566 g	∼515 g	∼600–650 g (external battery pack)
Compute	Snapdragon 850 + HPU	Snapdragon XR2 Gen 2	Apple M2 + R1 chip
Battery	2–3 hours (internal)	2–3 hours (internal)	2 hours (external pack)
Enterprise Integration	Strong (Azure, Dynamics)	Limited	Early-stage (Apple ecosystem)
Price (approx)	$3,500 USD	$500 USD	$3,499 USD

Table 2.12: Comparison between HoloLens 2, Meta Quest 3, and Apple Vision Pro.

Table 2.12 provides a comparative analysis of three prominent extended reality (XR) headsets: the HoloLens 2, Meta Quest 3 and Apple Vision Pro [24–26]. Figure 2.8 displays all three models.

Despite being older, HoloLens 2 still appears to be a strong alternative for didactic industrial simulations and enterprise-focused, hands-free AR applications. Its advantages include:

- **Purpose-Built for AR Workflows:** It is designed for hands-free use in fields like manufacturing, medical and field service, not gaming or general media consumption.
- **Full Transparency & Ergonomics:** HoloLens 2 lets users stay fully aware of their environment, crucial for safety and collaboration on the factory floor.
- **Enterprise Ecosystem:** Seamless integration with Microsoft's suite (Azure, Dynamics 365, Teams) ensures real-time data overlays, remote assistance, and guided workflows.
- **Robust Hand & Eye Tracking:** Enables precision in gesture control and user intent recognition, critical for interaction without external controllers.
- **No External Pack:** Unlike the Apple Vision Pro, everything is integrated into a single wearable unit (no cables or external batteries).

Figure 2.8: Comparison of mixed reality devices: Microsoft HoloLens 2, Meta Quest 3 and Apple Vision Pro.

[25]

2.8 GrowSkills Educational Workstation: UR Didactic Kit

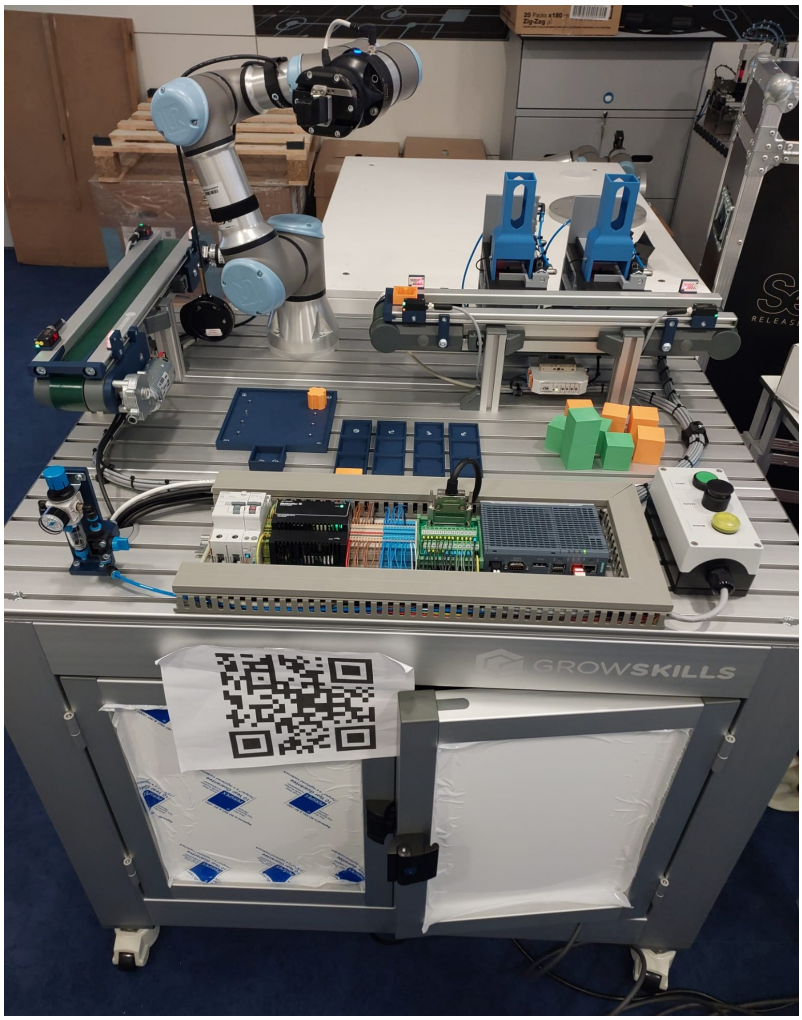


Figure 2.9: GrowSkills Educational Workstation setup.

The GrowSkills educational workstation (Figure 2.9), featuring the UR Didactic Kit, provides a practical and modular platform for learning collaborative robotics. This system integrates the Universal Robots UR3e collaborative robot arm, along with essential components such as the controller, teach pendant and the Robotiq Hand-E gripper. Designed specifically for educational and professional training environments, the workstation enables simulation of real industrial processes, offering students hands-on experience in automation and robotics.

This workstation is particularly suited for educational institutions, training centers, and companies aiming to upskill their workforce in automation and collaborative robotics technologies.

2.9 Software Development kits

A Software Development Kit (SDK) is a comprehensive set of tools and resources designed to help developers create applications for specific platforms or environments. It typically includes libraries, APIs, development environments, documentation, code samples, and testing tools, all aimed at simplifying and speeding up the development process while ensuring compatibility and quality. It typically bundles several essential components: [27]

- **Libraries**: Reusable code routines for common tasks.
- **APIs**: Interface endpoints that allow your code to use platform-specific features.
- **IDEs or Plugins**: Development environments for writing, debugging and compiling code.
- **Documentation & Code Samples**: Guides and examples to help developers learn and integrate efficiently.
- **Testing Tools**: Debuggers and compilers to catch and fix errors early.

There are several Augmented Reality Software Development Kits (AR SDKs) that devolpers frequently use to build AR applications across different platforms [28].

Vuforia is one of the most widely adopted AR SDKs, now owned by PTC. Written in C++, it supports robust features such as image and object recognition, extended tracking, ground-plane detection and offers both cloud and local database options. It is compatible with Android via Java APIs, iOS through C++, and Unity using C# [29].

ARToolKit is a free, open-source library that has been available since 2001, boasting over one million downloads. It leverages computer vision algorithms for camera calibration and object orientation tracking, primarily supporting marker-based AR across multiple platforms including Windows, Linux, macOS and SGI IRIX [30].

Google ARCore is Google's free and powerful AR SDK, designed for widespread smartphone use. Key features include 6 degrees of freedom (6DoF) motion tracking, surface detection (plane finding), and light estimation. It supports development with Java/OpenGL, Unity and Unreal Engine [31].

Apple ARKit is Apple's native AR framework for iOS devices, introduced in 2017. It offers advanced functionalities like people occlusion, motion capture, simultaneous use of front and back cameras, multiple face tracking and collaborative multi-user AR sessions [31].

MaxST, founded in 2010, provides separate toolkits for 2D and 3D AR. The 2D toolkit supports image and spatial tracking, while the 3D toolkit focuses on real-world object detection and tracking. It also includes QR code scanning capabilities and supports Android, iOS, Windows, and macOS platforms. Licensing ranges from a free version to paid Pro and Enterprise editions [32].

Finally, Wikitude and ZapWorks are also notable AR SDKs. Wikitude provides markerless SLAM and image/object tracking [28], whereas ZapWorks offers a universal AR SDK tailored for both web and app-based content creation [33].

In addition to these AR SDKs the Mixed Reality Toolkit (MRTK) is an open-source development framework provided by Microsoft that facilitates the creation of cross-platform mixed reality (MR) applications. MRTK v2 is designed to support rapid prototyping and scalable production deployments by offering a modular and extensible architecture. It provides a comprehensive set of building blocks for handling spatial input, user interactions, and platform-specific services, making it especially well-suited for devices such as the Microsoft HoloLens, Windows Mixed Reality headsets, Meta Quest, and other XR-enabled platforms via Unity XR. Among its key features are a robust input system that supports gaze, gesture, voice, and controller-based interactions, a spatial awareness system for mesh and surface mapping, and a collection of customizable user interface (UI) components optimized for immersive environments [34].

Additionally, MRTK supports editor-time simulation, allowing developers to test and iterate without needing to deploy to a physical device. It is organized into multiple Unity packages such as Foundation, Tools, Extensions and Examples, enabling fine-grained control over project dependencies.

Despite the release of MRTK v3, which introduces further modularization and enhanced support for Unity's XR Interaction Toolkit, MRTK v2 remains widely adopted and was considered more suitable for the objectives of this project. Development initially began using MRTK v3; however, several essential features were either unavailable or not fully integrated in the newer version, which led to the decision to proceed with MRTK v2.

In summary, the landscape of augmented and mixed reality development is supported by a diverse array of Software Development Kits (SDKs) and frameworks, each offering unique capabilities tailored to different platforms and use cases. General AR SDKs provide essential tools for image recognition, motion tracking, and environment understanding, enabling developers to create immersive AR experiences across mobile and wearable devices. Complementing these, the Mixed Reality Toolkit (MRTK) for Unity extends this ecosystem by delivering a comprehensive, modular and cross-platform framework specifically designed for mixed reality applications. MRTK enhances development productivity through its robust input systems, spatial awareness

features, and user interface components, while also supporting rapid prototyping with editor-time simulation.

2.10 Game Engines

AR game engines are specialized software platforms designed to facilitate the development of AR-based experiences, including games, applications and other forms of interactive content. These engines offer a comprehensive suite of tools, frameworks, and pre-built functionalities that enable developers to design and implement AR applications more efficiently, eliminating the need to build foundational systems from the ground up.

A variety of AR game engines are available on the market, each offering distinct capabilities, toolsets, and feature sets tailored to different development needs and platforms..The most common game engines utilized for developing AR applications are Unity3D and Unreal [35].

Unity is a leading real-time 3D development platform and game engine, renowned for its flexibility in enabling developers to build both 2D and 3D interactive content across over 20 platforms, including desktop, mobile, consoles, VR/AR and web browsers. The engine supports a comprehensive workflow, offering tools for scene construction, physics simulation, asset management, character animation, UI design, audio integration, and scripting via C# and .NET, all within an intuitive and extensible editor interface . Unity also emphasizes performance profiling through features like the Unity Profiler, Memory Profiler and Frame Debugger, facilitating optimization throughout the development cycle [36].

Unreal Engine, developed by Epic Games, is a leading real-time 3D creation platform, widely recognized for its powerful graphics rendering capabilities and extensive cross-platform support. Its most recent major version, Unreal Engine 5, introduces groundbreaking technologies such as Nanite, a virtualized micropolygon system that allows real-time rendering of highly detailed geometry without the need for manual Level-of-Detail management, and Lumen, a fully dynamic global illumination system that responds instantly to changes in lighting and geometry. Other key features include World Partition, which enables developers to manage and stream expansive open worlds more efficiently and MetaSounds, an advanced audio system that offers complete control over audio generation, similar to how materials are handled in the engine. The engine supports both C++ programming and Blueprints, a visual scripting system that allows developers to build complex logic without writing code. Due to its versatility, Unreal Engine is used not only in game development but also in virtual production, architecture, automotive design, and immersive technologies like AR and VR, making it a comprehensive tool across creative and industrial domains [37].

2.11 Digital Twin

Digital Twin (DT) technology has emerged as a foundational pillar of Industry 4.0, offering a high-fidelity virtual representation of physical systems that enables real-time synchronization, simulation, and intelligent decision-making. In industrial robotics, Digital Twins serve as a critical interface for enhancing motion planning, virtual commissioning, and monitoring of physical assets [38, 39]. A conceptual model illustrating the interaction between physical entities, virtual models, and service systems within a Digital Twin framework is presented in Figure 2.10.

According to Du *et al.* [40], a Digital Twin system can significantly improve the safety and efficiency of industrial robot operations by providing a simulation environment that mirrors the robot's physical behavior. The authors propose a four-dimensional architecture consisting of a physical entity, virtual model, connection interface, and application services. This structure allows for detailed motion simulation, enhanced collision detection using a hybrid bounding box algorithm, and real-time monitoring capabilities via Unity3D, enabling effective trajectory validation before deployment on real hardware.

Complementarily, Delfino *et al.* [41] explore the broader integration of Digital Twins within IoT environments. They highlight the potential of Digital Twins to act as semantic bridges between heterogeneous industrial systems, facilitating not only simulation but also predictive analytics, lifecycle monitoring, and process optimization through real-time data processing and artificial intelligence. Their study emphasizes the need for a standardized reference architecture that encompasses properties such as runtime environment, API integration, interoperability, scalability, and secure communication which are all essential for successful implementation in dynamic production ecosystems.

Together, these perspectives underscore the transformative role of Digital Twins in smart manufacturing and justify their use as a core component in this dissertation's objective: the development of an augmented reality system to support operator interaction and real-time assistance in industrial processes. By aligning AR interfaces with Digital Twin architectures, the proposed system aspires to enhance situational awareness, enable remote diagnostics, and reduce operational errors in environments such as the GrowSkills Educational Workstation.

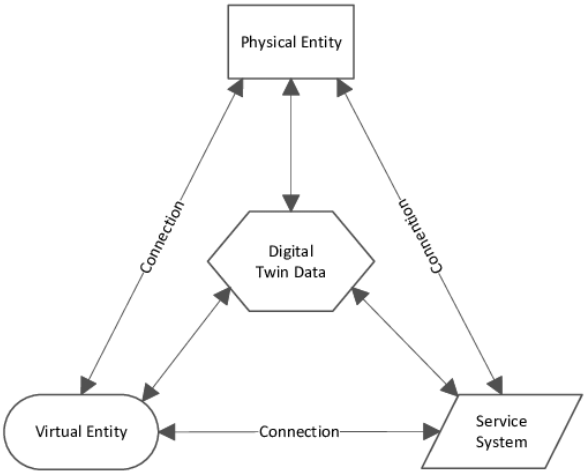


Figure 2.10: Conceptual model of a digital twin.
[42]

3D Models for Digital Twins

Three-dimensional (3D) modeling plays a pivotal role in the creation and implementation of DT systems, serving as the foundational layer that bridges the physical and virtual realms. As highlighted by Menon *et al.* [43], a Digital Twin is not only a digital representation of a physical asset, but a dynamic, interactive and evolving model that relies heavily on real-time data integration, accurate simulation and immersive visualisation.

In the context of DTs, 3D models are used to replicate the physical geometry, spatial configuration, and operational dynamics of real-world entities. These virtual replicas enable engineers, operators and decision-makers to visualize and interact with complex systems through intuitive interfaces. The use of computer-aided design (CAD) tools is especially prominent in the modeling phase, allowing the development of detailed and adaptable virtual structures that can be updated continuously as sensor data streams in [43, 44].

Furthermore, interactive tools such as Unity 3D and AR platforms like Microsoft HoloLens 2 are integrated with DT systems to enhance model interactivity and user immersion. These platforms allow for real-time manipulation, monitoring, and simulation of the system's behavior in an intuitive and spatially contextualized manner [45].

Despite the benefits, the creation of accurate and responsive 3D models for DTs poses challenges. These include ensuring data fidelity, achieving interoperability with legacy systems, and maintaining synchronization between the physical asset and its digital counterpart.

In the development of Digital Twin systems, the selection of an appropriate 3D modeling tool is critical to ensure geometric accuracy, real-time responsiveness, and seamless integration with simulation platforms.

Among the various 3D modeling tools available, SolidWorks and Blender stand out as two of the most widely used due to their distinct advantages. SolidWorks offers industry-grade precision, robust parametric modeling, and seamless integration with engineering workflows, making it ideal for designing mechanical components and assemblies [46]. Blender, on the other hand, is favored for its intuitive interface, powerful mesh modeling capabilities, and open-source flexibility,making it especially suitable for creating detailed visualisations and interactive assets. Their balance of usability, versatility, and broad community support gives them an edge over more specialized or less accessible alternatives [47].

This project adopts Blender, an open-source and cross-platform 3D modeling software, due to its powerful capabilities and flexibility in supporting complex modeling tasks. Blender provides a comprehensive suite of features including mesh modeling, procedural geometry through Geometry Nodes, sculpting, UV mapping, and animation tools and they are all essential for constructing detailed and interactive virtual representations of physical systems. Its support for widely-used file formats (e.g., `.obj`, `.fbx`, `.gltf`) facilitates smooth integration with engines such as Unity and Unreal, which are commonly employed in Digital Twin visualisation and AR applications. Moreover, Blender's active community and extensive library of plugins enable rapid prototyping and customization, making it a suitable platform for educational, industrial, and research-based Digital Twin implementations. In this work, Blender is used to create accurate 3D models of robotic components and workstation layouts, which are then integrated into the augmented reality interface to support interactive simulation and task assistance [47].

2.12 TCP/IP Communication

TCP/IP (Transmission Control Protocol/Internet Protocol) is the fundamental communication protocol suite that governs data exchange across interconnected computer networks, including the internet. It provides a standardized framework that allows devices to communicate reliably and efficiently, regardless of their underlying hardware or operating systems [48].

The TCP/IP model is structured into four layers, each responsible for specific aspects of communication [49]:

- **Application Layer** – This top layer provides network services directly to the user or application, such as HTTP, FTP, and DNS. It handles high-level protocols and data formatting.
- **Transport Layer** – The core of this layer is the Transmission Control Protocol (TCP) which ensures reliable data delivery through error checking, acknowledgments and retransmissions. Alternatively, the User Datagram Protocol (UDP) is used for faster, connectionless communication when reliability is less critical.

- **Internet Layer** – This layer is responsible for logical addressing and routing. The Internet Protocol (IP) assigns unique IP addresses and directs packets between source and destination across multiple networks.
- **Network Access Layer** – Also known as the link layer, it manages the physical transmission of data over the network interface (e.g., Ethernet, Wi-Fi), handling hardware addressing and data framing.

TCP/IP communication is essential for enabling interoperability among devices in distributed systems. In practical applications such as in robotics, AR systems or client-server architectures, allows various components to exchange data seamlessly, enabling real-time control, monitoring or interaction over a local network or the internet.

Chapter 3

Methodology and Implementation

This chapter delineates the methodology and work plan employed to guide the research and development undertaken during the dissertation phase. It provides a comprehensive account of the challenges faced and the key decisions and strategies adopted throughout the project.

This chapter also outlines the practical implementation of the system developed to enable real-time interaction between a physical UR3e robotic arm and its virtual counterpart within a mixed reality environment using the HoloLens 2. The focus is on integrating data communication, virtual representation and user interaction to support tasks commonly associated with industrial automation and digital twins.

Key aspects of the implementation include the retrieval and processing of real-time data from the robot, such as gripper position and input/output states, using TCP/IP socket communication. This data is then used to drive the simulation inside Unity, enabling accurate synchronization between the physical robot and its digital model.

Furthermore, this chapter describes the design and functionality of various user interface elements, including the Slate Bar, Dashboard, and Robot Control Panel. These components not only display live robot data but also allow the user to send commands remotely and interact with the system through a structured and intuitive AR interface.

3.1 Methodology

The aim of this project was to develop a mixed reality system that allows users to monitor and interact with a collaborative robot in real-time. The chosen approach integrates hardware (UR3e robot and HoloLens 2), software components (Unity, Mixed Reality Toolkit), and communication protocols (TCP/IP sockets) to synchronize physical and virtual behavior.

The methodology is divided into the following stages:

1. **Preliminary Research and Familiarization:** During the initial phase of the project, particularly within the Introduction to Research course unit, a preliminary review of the state of the art was conducted to establish familiarity with the research domain and the technologies relevant to the proposed implementation. This was guided by discussions with the dissertation supervisor. Subsequently, a more in-depth investigation was performed to broaden the knowledge base and more effectively support the practical development phase.

2. **Technology Selection and Environment Setup:** After defining the goals and scope of the project, the most suitable tools and frameworks were selected. The application was developed using Unity 3D, with the Mixed Reality Toolkit (MRTK) integrated to support spatial interactions and interface design on the HoloLens 2. Key features such as spatial awareness, input systems and UI components from MRTK were configured according to the project's specific needs. On the robotics side, URScript and the UR3e robot's real-time data streams were explored, allowing seamless integration with the virtual environment via socket communication.

3. **System Architecture and Communication Design:** The architecture was divided between the physical robot layer and the virtual reality layer. A TCP socket-based communication system was established to transmit live data from the UR3e robot to the HoloLens application. Custom scripts were developed to manage the reception, parsing, and real-time representation of this data inside Unity.

4. **Iterative Implementation and Integration:** The development followed an iterative approach. Components such as the GripperPositionDisplay system, the Robot Control interface and the Dashboard were individually implemented, tested and integrated. Each feature was validated in isolation before being combined into the final mixed reality experience.

5. **User Interface and Interaction Design:** The interface was designed to balance usability and technical functionality. Panels such as the Start Menu, Slate Bar, Robot Control and Dashboard were implemented using MRTK's UI system, allowing intuitive user interactions via hand tracking. Elements like the gripper position, program status and digital outputs were displayed in real time, ensuring transparency and feedback.

6. **Testing and Refinement:** The final stages involved rigorous testing in both simulated and real-world settings. Particular attention was given to communication stability, latency and user experience. Edge cases and reset conditions were handled to ensure robustness and system reliability.

3.2 Architecture

In this section, the solution for the system's architecture is presented, detailing its design and how it evolved during the dissertation phase. In general terms, a Digital Twin model consists of

two primary sections: the Physical side of the system and the Digital side, interconnected by a communication protocol.

At a preliminary stage of this project, an architecture was initially considered based on existing, widely adopted frameworks. Specifically, integrating the Robot Operating System (ROS) was explored as a communication protocol. ROS is a robust open-source framework prevalent in the robotics domain, offering extensive tools and libraries for hardware abstraction and messaging. The initial thought was that implementing the communication layer within the ROS framework could enhance the system's adaptability and compatibility with various industrial robots [50].

However, this decision introduced significant complexities. The integration of ROS with the Universal Robots (UR3e) controller presented considerable academic and technical challenges, including complex configuration requirements and persistent errors due to missing libraries, which made this approach impractical for the scope and timeline of the dissertation. The necessity of building and managing a separate server to publish and subscribe to ROS nodes, specially in C#, added undesirable layers of intricacy to the communication process, increasing latency and potential points of failure.

On the other hand, direct communication approaches with the UR3e robot proved to be more effective, offering lower latency and significantly simplifying the system architecture. This alternative approach allowed for a streamlined communication process directly between the head-mounted display and the robot controller, enhancing the system's efficiency and robustness by reducing potential points of failure and simplifying data extraction paths.The final architecture of the developed Digital Twin, as presented in Figure 3.1, maintains the core Physical and Digital sections, but with a refined and direct communication protocol. This communication primarily takes place through TCP/IP sockets, simplifying the path and allowing for direct and centralized extraction of all necessary information.

3.2.1 Physical System

The physical aspect of the system comprises the core robotic elements. In this work, the Universal Robots UR3e collaborative robot is the central component. The real-world actions of the robot are meticulously managed and ordered by its Robot Controller. It is the controller's responsibility to manage the robot's movements and expose its operational data.

Additionally, the Teach Pendant acts as a direct human-machine interface (HMI), enabling the operator to interact with and command the robot directly and independently from the augmented reality system.

3.2.2 Digital System

The digital side of the system corresponds to the domain of the AR Digital Twin, entirely constructed within the HoloLens 2 application. Here, the HoloLens 2 establishes a direct line of communication with the robot controller, leveraging the data it receives to compute and replicate real-world movements and actions within a virtually identical digital representation of the robot and its environment.

The HoloLens 2 performs Data Processing of the incoming robot streams, using this information to drive the Virtual Robot (applying forward kinematics to animate the 3D model) and update the Virtual Workbench (synchronizing the state of virtual objects with physical processes).

The Digital Twin application, being self-contained within the HoloLens 2, relies solely on the data values retrieved through the communication interface, making it adaptable to future changes in the communication protocol or modifications in the physical infrastructure without compromising its core functionality. This modular design aims for flexibility in both the digital and physical sections, enabling future advancements and adjustments to be integrated into the system.

3.2.3 Communication Protocol

The communication architecture is structured around a bidirectional TCP/IP socket interface, enabling real-time data acquisition and command transmission between the Universal Robots system and the HoloLens 2 application. This connection, illustrated as the TCP/IP link in Figure 3.1, can be divided into two main channels:

1. **Data Streaming (Robot to HoloLens 2):** Real-time data (joint orientations, Cartesian positions, standard digital I/O) is streamed from the UR3e's Realtime Client Interface on Port 30013. Additionally, a custom URScript on the robot streams configurable digital I/O states and gripper position data on Port 30021, ensuring comprehensive data acquisition by the HoloLens 2's Data Processing unit.

2. **Command Transmission (HoloLens 2 to Robot):** Control commands originating from the HoloLens 2 application (e.g., digital output toggles) are sent back to the robot controller via TCP/IP on Port 30003.

This direct TCP/IP approach, instead of a middleware like ROS, significantly simplifies the path for data and commands, streamlining the overall system and minimizing latency, which is crucial for real-time augmented reality applications.

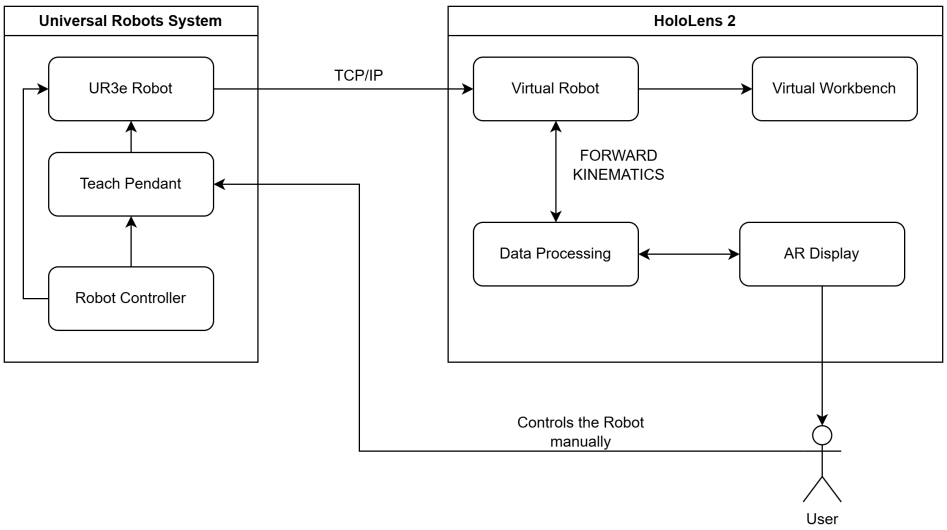


Figure 3.1: System architecture diagram used in the implementation.

3.3 UR3e Robot Configuration

Given the type of gripper being used, it's necessary to configure the robot by adjusting the values for payload, center of gravity and TCP (Tool Center Point), which is the specific point on a robot where the end-effector (such as a gripper or welding torch) is attached and represents the exact location in space where the tool interacts with the workpiece and serves as the reference for the robot's positioning and movement 3.2, as illustrated in Figure 3.3 . This configuration can be accessed directly from the teach pendant by navigating to Installation -> General -> TCP and Installation -> General -> Payload.

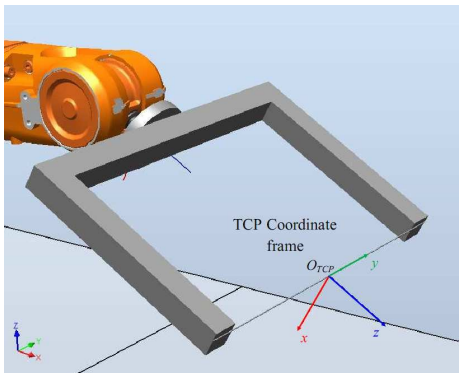


Figure 3.2: Tool Center Point example.
[51]

Figure 3.3: Configuration of the center of gravity and the payload.

During execution of the main program, the palletizing function available in the teach pendant is occasionally used. This function organizes and moves parts in a systematic way from an initial position, either random or fixed, to predefined final positions. These final positions depend on the desired palletizing layout, which could be linear, grid-based, or irregular. It's also possible to define a separator between layers, but this isn't applicable in the current project since it involves only a single layer. The palletizing function can be found by going to Program -> Templates -> Palletizing on the teach pendant.

The force control function is used when it's necessary to insert the inner part of a piece into a box. This is important because the inner part may be held by the robot in a random orientation, and successful insertion requires a specific alignment. The force function allows the robot to make this adjustment by continuously comparing the current TCP position with a reference point. In this case, force control is applied only along the Z and RZ axes, meaning the robot will move downward and rotate with a controlled force. This function can also be accessed through Program -> Templates on the teach pendant.

Eventually, the inner part will fit into place. At that point, the function get_actual_tcp_pose() is used to compare the current TCP position with a reference Z-value, which represents the height of the TCP when the piece is properly inserted. Since the XY coordinates remain unchanged, only the third element of the returned position vector (Z) is relevant. The check is performed as pos_tcp[2] > 0.03507. Both the palletize and force functions involved in this process are illustrated in Figure 3.4.

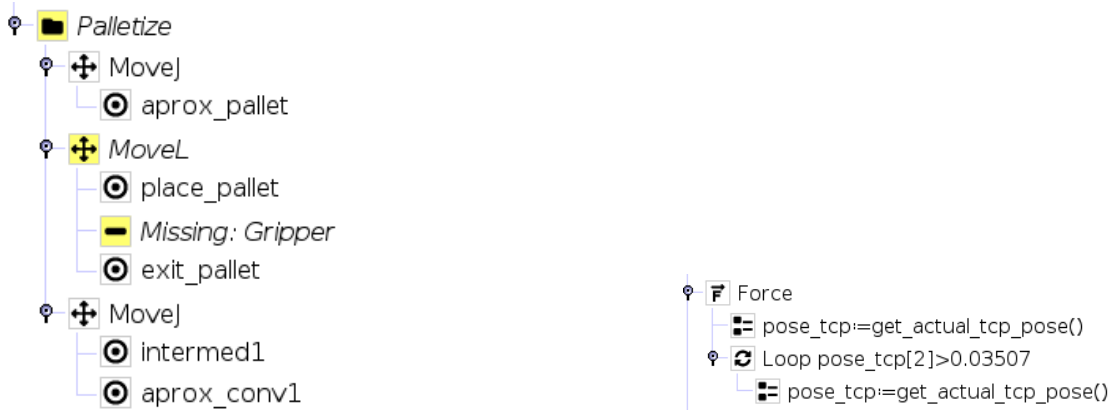


Figure 3.4: Code used for paletize and force.

3.4 Hololens 2 Configuration

To enable effective interaction, the Microsoft HoloLens 2 device was configured using MRTK, as introduced in Section 2.9.

A key component in delivering immersive experiences is the Spatial Awareness system. This subsystem captures the physical surroundings by generating spatial meshes that represent the geometry of the real world. These meshes enable holograms to interact naturally with physical objects, such as placing virtual items on tables or occluding virtual content behind walls [52].

Within MRTK, developers have control over how these spatial meshes are visualized. Options range from completely hiding the meshes, allowing holograms to appear uninhibited by the environment to fully rendering them as visible surfaces or using them solely for occlusion, where physical objects block virtual ones without displaying the mesh geometry.

For this project, the decision was made to disable mesh rendering to prevent the spatial mesh from obstructing virtual content. This choice enhances the user's perception of virtual objects seamlessly integrated into the environment, avoiding visual barriers caused by the physical geometry representation.

In addition to spatial awareness, MRTK handles other essential configurations such as input management, enabling hand tracking, voice commands and eye gaze interaction.

3.5 QR Code Tracking

To enable accurate spatial alignment between virtual and physical elements in the AR environment, two key scripts were adapted and integrated into the project: QRcodeFinder.cs and AnchorLocator.cs. These scripts, originally developed for general AR applications, were customized to meet the specific requirements of this system, ensuring compatibility with the HoloLens 2 and the digital twin of the workbench.

The QRcodeFinder.cs script is responsible for detecting and tracking QR codes in the physical environment. It uses the Microsoft.MixedReality.QR namespace to identify QR codes, extract their unique identifiers, and obtain the corresponding spatial graph node IDs. This information is stored in a concurrent dictionary, allowing efficient access for anchoring virtual objects. The specific QR code utilized for the workstation setup is depicted in Figure 3.5.

The AnchorLocator.cs script builds upon this data to locate and position virtual elements accurately within the AR scene. It retrieves the appropriate spatial graph node associated with a detected QR code and continuously updates the position and orientation of virtual objects relative to the physical space. This ensures that the digital components remain precisely aligned as the user moves or interacts with the environment.

By adapting and integrating these scripts, the application achieves reliable spatial anchoring of the digital twin. The QR codes serve as dynamic markers that facilitate real-time synchronization between virtual and physical objects. This alignment is critical for delivering an immersive and responsive AR experience, enabling users to explore and interact with the digital twin in a natural and intuitive manner. The flexibility offered by QR code tracking also enhances the system's practicality compared to static displays or fixed anchors, allowing for a more dynamic and adaptable AR solution.

Figure 3.5: QR code used for the workstation setup.

3.6 3D Models

To create a digital twin of the UR3e robot and its surrounding environment, 3D models were utilized to represent the robot, the workbench, conveyors and all adjacent components within the setup. Initially, a search was conducted for existing 3D models of the UR3e robot. After thorough research, a model that closely matched the requirements was identified and selected. The workbench, as shown in Figure 3.6, conveyors, Figure 3.7 , and the remaining components, Figure 3.8, were provided by Growskills, which already had accurate 3D models available. This significantly streamlined the modeling process.

All 3D models were assembled and prepared in Blender. It was also necessary to modify the robot model to include the Robotiq gripper, which was not present in the original file. Additionally, adjustments were made to the referential structure of each robotic component to ensure that every joint aligned correctly with its corresponding pivot point. This alignment was essential for the correct implementation of kinematic behaviors in Unity, ensuring that each link rotated appropriately without causing the model to break apart. This configuration follows a hierarchical parent–child relationship as illustrated in Figure 3.9.

Furthermore, material creation and assignment were also carried out in Blender. This step was critical, as Blender does not render colors properly when exporting to other formats unless materials are explicitly defined. Throughout the process, model complexity was intentionally minimized to optimize the application's performance.

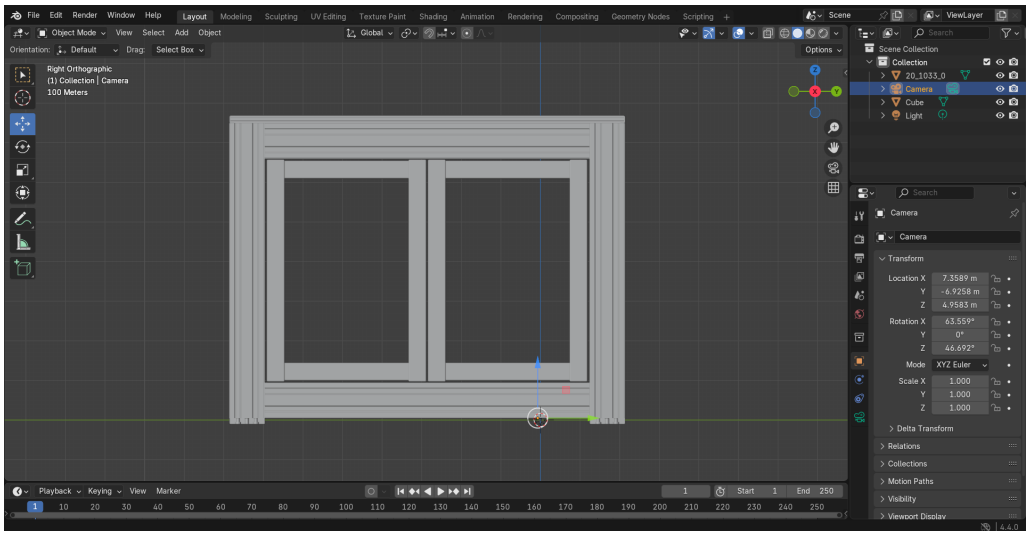


Figure 3.6: Workbench 3D Model.

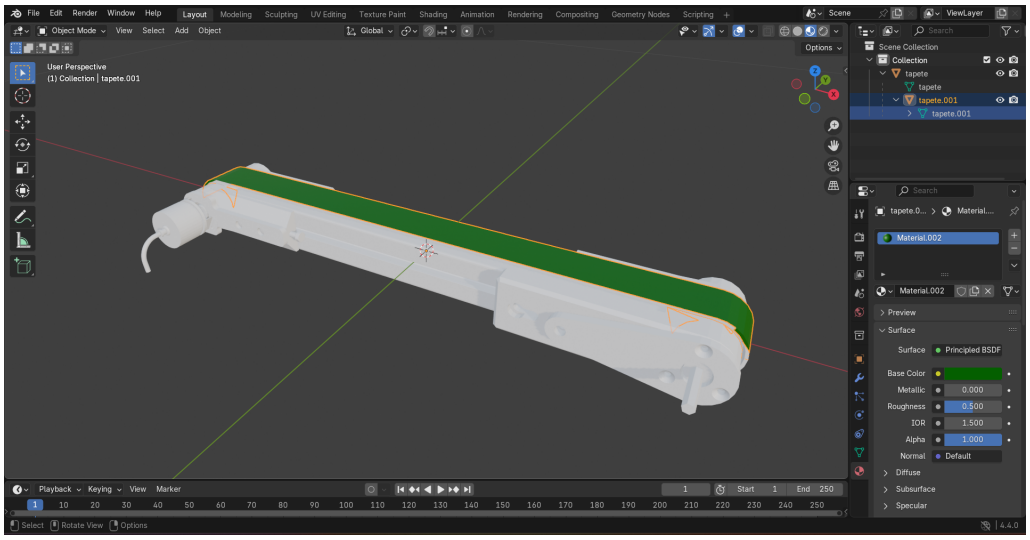


Figure 3.7: Conveyor 3D Model.

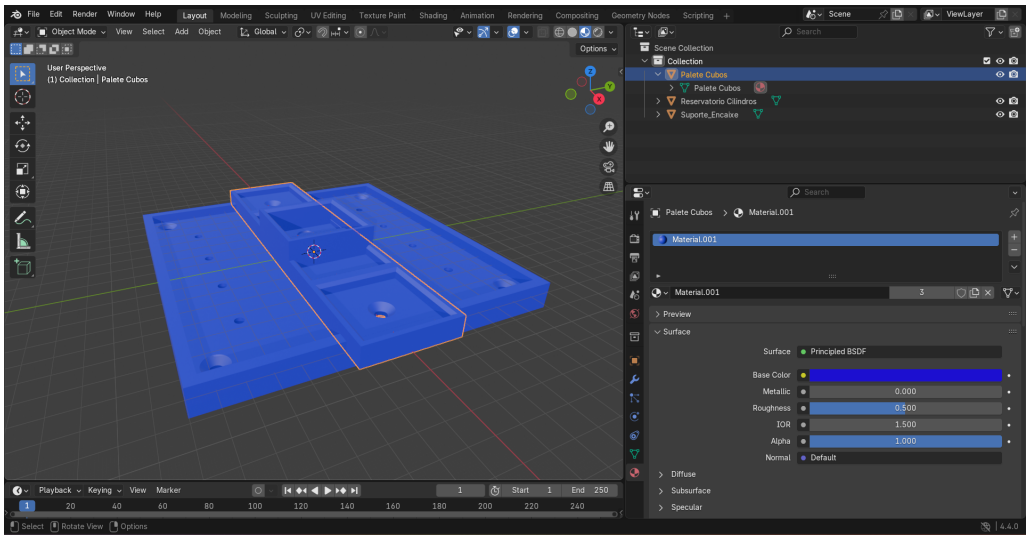


Figure 3.8: Warehouse 3D Model.

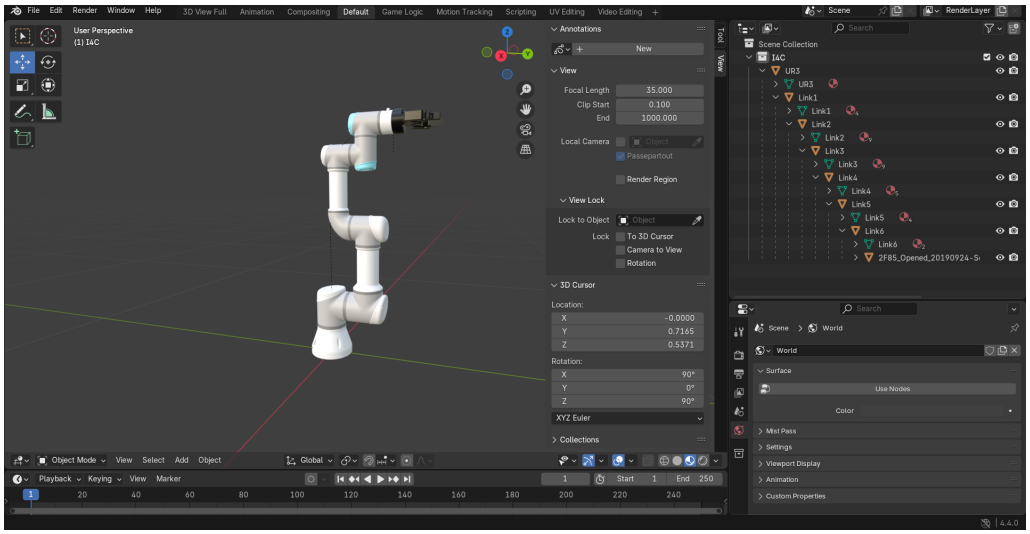


Figure 3.9: UR3e Robot 3D Model.

In addition to these models, 3D representations of the cube with the cylinder (Figure 3.10), the completely closed cube (Figure 3.11), and the cube without the top lid (Figure 3.12) were created.

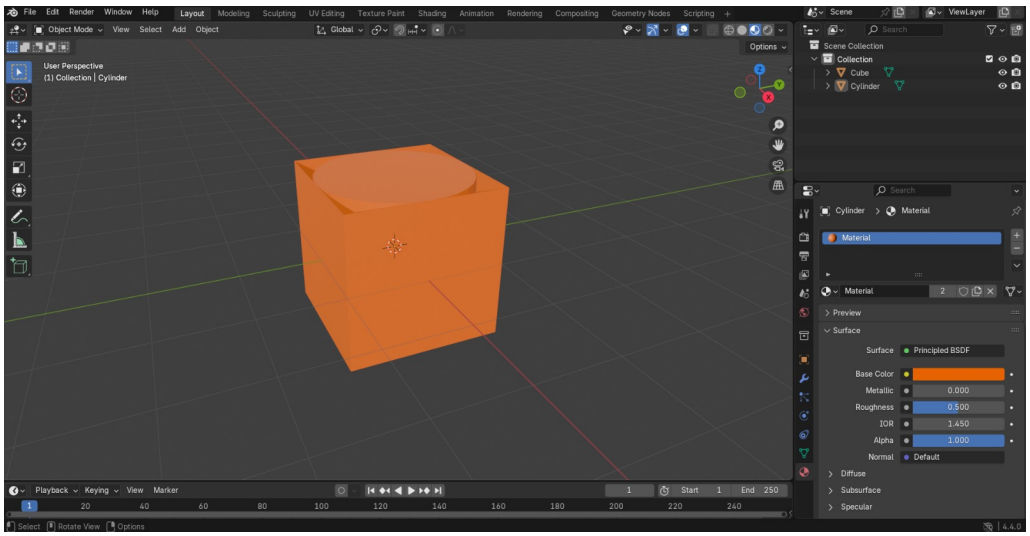


Figure 3.10: Model of the cube with central cylinder.

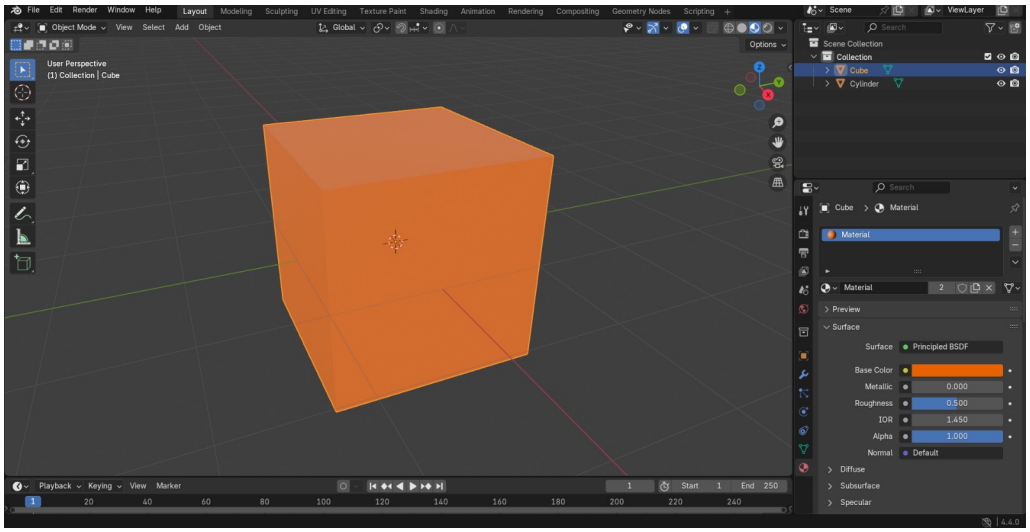


Figure 3.11: Model of the completely closed cube.

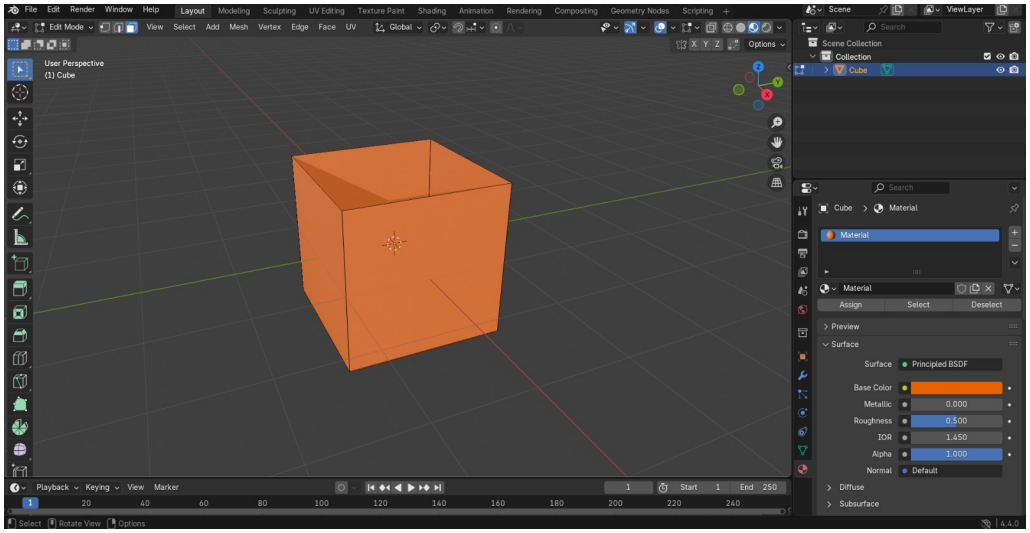


Figure 3.12: Model of the cube without the top lid.

3.7 Digital Twin

One of the primary objectives of this dissertation is the development of a digital twin to accurately represent the processes occurring on the workbench. To achieve this, it was essential that the 3D models used were properly animated and could be controlled by the application developed in Unity for the HoloLens 2.

This section details the solutions implemented for representing the robot's kinematics, the extraction and visualisation of real-time data from the physical system, and the integration of all elements that comprise the digital workbench.

3.7.1 UR3e Robot Data Processing

During development, several strategies were evaluated for retrieving data from the UR3e robotic arm. The Robot Operating System (ROS) was explored, an open-source framework offering hardware abstraction, messaging, and numerous tools for robotic applications [50]. However, integrating ROS with C# and Unity proved challenging due to complex configuration requirements and missing libraries, which led to persistent errors and ultimately made this approach impractical.

Subsequently, the Real-Time Data Exchange (RTDE) interface was adopted, a protocol well-documented by Universal Robots that facilitates synchronous data exchange using TCP/IP [53]. While the RTDE setup allowed for robust joint and tool feedback, some URScript commands did not function as expected, resulting in intermittent data loss.

Finally, the Realtime Client Interface was implemented successfully. This interface provides high-frequency (up to 500 Hz) access to key robot variables, including joint angles, temperatures, and digital I/O statuses over TCP/IP [54]. The overall structure of the data processing code, including the ur_data_processing class and its internal components, is presented in Figure 3.13.

The implementation consists of a main controller class (ur_data_processing) and two internal classes: UR_Stream for receiving data and UR_Control for sending commands. These classes operate asynchronously using dedicated threads and TCP/IP socket communication.

At the start of the application, two key IP configurations are set:

- **Port 30013** is used for streaming sensor and joint data.
- **Port 30003** is used for sending control commands.

The main class also defines shared static data containers for both streamed sensor data (UR Stream Data) and control signals (UR Control Data). These include arrays to store joint orientation, Cartesian position, temperatures, and digital I/O states.

3.7.2 Real Time Stream Handling - UR Stream

The system extracts several critical data fields from the incoming robot data stream:

- Joint Orientations (J_Orientation)
- Cartesian Position and Orientation (C_Position, C_Orientation)
- Joint Temperatures (J_Temperature)
- Digital Inputs/Outputs: Interpreted by converting double values to 64-bit integers and extracting individual bits

The use of the BitConverter.ToDouble() function, combined with byte offsets, allows precise mapping of the binary data structure received from the robot. To maintain real-time performance, a stopwatch mechanism is used to regulate data polling based on the interval defined in UR_Stream_Data.time_step.

Control Robot – UR_Control

The UR_Control class complements the stream reader by managing outgoing commands to the robot. These include:

- Digital output toggles
- Custom URScript byte commands

Commands are written to the network stream at fixed intervals, ensuring synchronization with the robot's control cycle.

Main Logic – FixedUpdate()

Unity's FixedUpdate() function orchestrates the entire communication process:

- On a connection request, both UR_Stream and UR_Control threads are initiated.
- Based on input flags (such as button presses), logic is computed to manage robot movement or activate digital outputs.
- Upon disconnection, both communication threads are properly shut down.

This logic ensures that data is continuously received and processed, updating real-time variables that are subsequently used for visualisation, control, or augmented reality projection within Unity. Each communication class operates within its own thread.

```
UR_Stream_Data.J_Orientation[0] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (32 * offset));
UR_Stream_Data.J_Orientation[1] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (33 * offset));
UR_Stream_Data.J_Orientation[2] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (34 * offset));
UR_Stream_Data.J_Orientation[3] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (35 * offset));
UR_Stream_Data.J_Orientation[4] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (36 * offset));
UR_Stream_Data.J_Orientation[5] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (37 * offset));

UR_Stream_Data.C_Position[0] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (56 * offset));
UR_Stream_Data.C_Position[1] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (57 * offset));
UR_Stream_Data.C_Position[2] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (58 * offset));

UR_Stream_Data.C_Orientation[0] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (59 * offset));
UR_Stream_Data.C_Orientation[1] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (60 * offset));
UR_Stream_Data.C_Orientation[2] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (61 * offset));

UR_Stream_Data.J_Temperature[0] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (87 * offset));
UR_Stream_Data.J_Temperature[1] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (88 * offset));
UR_Stream_Data.J_Temperature[2] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (89 * offset));
UR_Stream_Data.J_Temperature[3] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (90 * offset));
UR_Stream_Data.J_Temperature[4] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (91 * offset));
UR_Stream_Data.J_Temperature[5] = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (92 * offset));

// Digital OUTPUT (offset 131)
double digitalOutputValue = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (131 * offset));
ulong outputBits = (ulong)digitalOutputValue;
for (int i = 0; i < UR_Control_Data.current_digital_outputs.Length; i++)
    UR_Control_Data.current_digital_outputs[i] = (outputBits & (1UL << i)) != 0;

// Digital INPUT (offset 86)
double digitalInputValue = BitConverter.ToDouble(packet, packet.Length - first_packet_size - (86 * offset));
ulong inputBits = (ulong)digitalInputValue;
for (int i = 0; i < UR_Control_Data.current_digital_inputs.Length; i++)
    UR_Control_Data.current_digital_inputs[i] = (inputBits & (1UL << i)) != 0;
```

Figure 3.13: Overview of the data processing code.

While most robot data such as joint orientation, cartesian pose and standard digital I/O was successfully retrieved using the RealTime Client Interface, accessing configurable digital inputs and outputs required a different approach. This is because the RealTime Client Interface does not expose offsets or structured access for these specific signals.

The URIOReceiver class is responsible for establishing a TCP listener within the Unity application (running on the HoloLens 2) to receive configurable input/output (I/O) data from the Universal Robots controller. It operates on a background thread, continuously waiting for incoming messages on port 30021. These messages are expected to contain the current states of the robot's configurable digital inputs and outputs, which are parsed and reflected in the user interface as virtual LEDs.

As a workaround, a custom URScript was developed and deployed on the robot. This script uses the robot's TCP socket API to send the status of configurable digital inputs and outputs over a dedicated TCP/IP port to the HoloLens application. Within the script, the states of the eight configurable digital inputs and outputs are sequentially queried using the UR Script commands get_configurable_digital_in() and get_configurable_digital_out(), respectively. These values are converted to strings and concatenated into a single message.The URScript code for retrieving configurable inputs/outputs is shown in Figure 3.14.

```
socket_open("192.168.0.110", 30021)

msg = ""

i = 0
while i < 8:
      msg = msg + to_str(get_configurable_digital_in(i)) + " "
      i = i + 1
end

i = 0
 while i < 8:
      msg = msg + to_str(get_configurable_digital_out(i)) + " "
      i = i + 1
end

socket_send_string(msg + "\n")
sleep(0.1)socket_close()
```

Figure 3.14: Script for the UR3e Robot to get configurable inputs/outputs.

3.7.3 Robot Links

Forward kinematics is a foundational concept in the field of robotics, concerned with determining the spatial position and orientation of a robot's end-effector, such as a gripper or tool, based on the known joint parameters, including the angles and link lengths of each segment. This process involves mapping the robot's joint space into its Cartesian workspace. A widely adopted approach for performing forward kinematics is the Denavit-Hartenberg (DH) convention, which standardizes the description of kinematic chains by defining a set of parameters for each joint. These parameters describe the relative position and orientation between successive links [55]. Through the application of homogeneous transformation matrices derived from the DH parameters, a sequential chain of coordinate transformations is established from the robot's base frame to the end-effector frame. This procedure enables the precise computation of the end-effector's position and orientation in three-dimensional space [56]. A schematic representation of Link 1 of the UR3e robot is shown in Figure 3.15, and the corresponding code snippet for its implementation is provided in Figure 3.16.

In this specific case, considering that the UR3e robotic arm is composed of multiple joints and interconnected links, forward kinematics is applied to compute the precise position and orientation of each joint, as well as the end-effector (gripper). This computation is essential for determining the spatial configuration of the manipulator based on known joint parameters, enabling accurate control and integration in tasks such as object manipulation, path planning and collaborative automation.

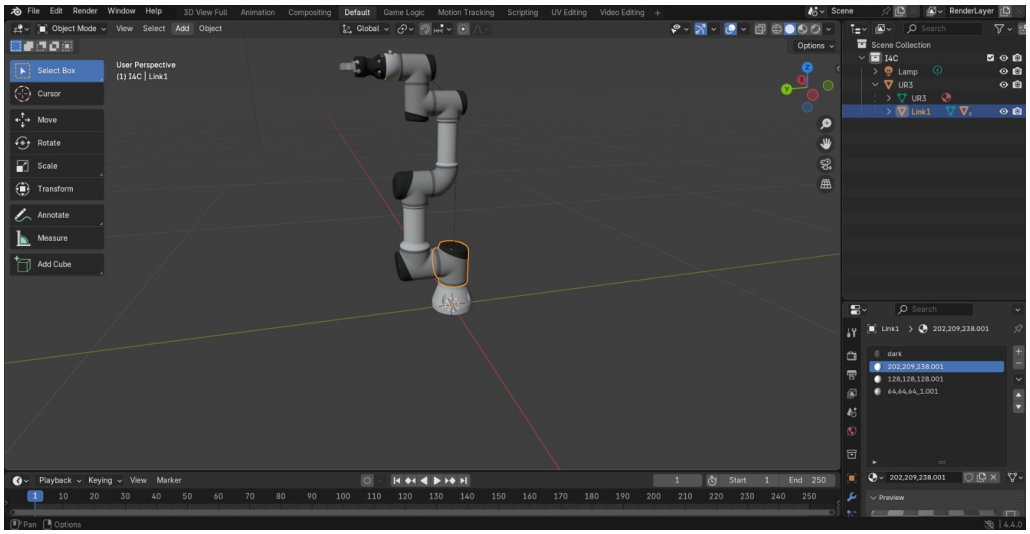


Figure 3.15: Schematic representation of link 1.

```
// System
using System;
// Unity
using UnityEngine;
using Debug = UnityEngine.Debug;

Unity Script (11 asset references) | 0 references
public class ur3_link1 : MonoBehaviour
{
    Unity Message | 0 references
    void FixedUpdate()
    {
        try
        {
            transform.localEulerAngles = new Vector3(0.0f, (-1.0f) * (float)(ur_data_processing.UR_Stream_Data.J_Orientation[0] * (180.0 / Math.PI)), 0.0f);
        }
        catch
        {

        }
    }
    Unity Message | 0 references
    void OnApplicationQuit()
    {
        Destroy(this);
    }
}
```

Figure 3.16: Code snippet related to link 1 implementation.

3.7.4 Eye Gazing

In Mixed Reality applications developed with MRTK for HoloLens 2, eye tracking enables intuitive and natural user interactions by leveraging the user's gaze direction as an input method. The eye-gazing system in MRTK allows developers to determine where the user is looking in the environment, enabling features such as object selection, well-based interaction or attention-aware UI adjustments. To implement eye tracking, the MRTK input system must have eye tracking enabled, and the application must request the appropriate capability (Gaze Input) in the app manifest. Within Unity, the EyeGazeProvider component, part of the MRTK input system, provides access to real-time eye gaze data. Developers can then use this gaze ray to trigger interactions with scene objects, using standard IMixedRealityPointerHandler interfaces or by leveraging components like OnLookAtShowHandMesh or TargetGroupIterator. Eye tracking is especially valuable in hands-free scenarios or when combined with other input modalities, such as voice commands or hand gestures, enhancing the overall user experience in mixed reality [57].

To enhance user interaction through gaze in the mixed reality environment, a modular system was developed using two complementary scripts. This system enables the display of contextual information panels,such as the one shown in Figure 3.17, when the user focuses their gaze on specific robotic joints or components within the scene.

The GazeRaycastManager script is responsible for continuously projecting a ray from the user's head (camera) in the forward direction. When this ray intersects with objects belonging to a predefined layer (box colliders attached to robot joints), it triggers the interaction. If a joint is detected, the script checks whether it has changed from the previously gazed object and, if so, invokes the corresponding handler script to display the associated information panel. If the user stops looking at the object for a certain period, the panel is automatically hidden.

The JointGazeHandler script manages the actual display and positioning of the information panel relative to the user's view. It ensures that only one panel is visible at a time, dynamically aligning it in front of the user's gaze while maintaining readability. A timer-based system is used to automatically hide the panel after a short delay, minimizing visual clutter.

Together, these scripts provide a responsive, gaze-driven feedback mechanism that supports hands-free interaction, improving the intuitiveness and usability of the augmented interface in a collaborative or industrial setting.

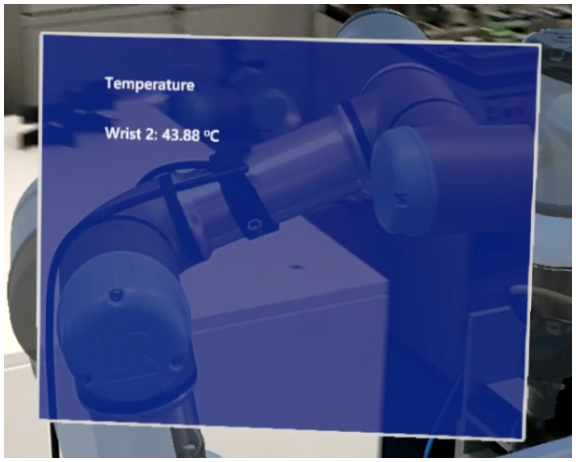


Figure 3.17: Eye Gazing Panel.

3.7.5 Gripper Link

The LinkGripper script (Figure 3.18) visually animates the opening and closing of a robot gripper in Unity by moving leftGripper and rightGripper (which represent the virtual fingers) based on real-time gripper position data received from another script GripperPositionDisplay.

Every frame (Update()), it reads the raw gripper position (between 1 = closed and 49 = fully open) from the GripperPositionDisplay script. It calculates how far open the gripper is as a percentage between closed and open (inverted, because lower values = closed). Then it maps that percentage to a visual distance range (from 0 to 25 units).

```
using UnityEngine;

public class LinkGripper : MonoBehaviour
{
    public Transform leftGripper, rightGripper;

    private GripperPositionDisplay gripperDisplay;

    private void Start()
    {
        gripperDisplay = FindObjectOfType<GripperPositionDisplay>();
        if (gripperDisplay == null)
        {
            // Debug.LogError("GripperPositionDisplay nao encontrado");
        }
    }

    private void Update()
    {
        if (gripperDisplay == null) return;

        // 1 (fechado) ate 49 (aberto)
        float percentage = 1f - ((float)gripperDisplay.RawPosition - 1f) / (49f - 1f);

        // 0% corresponde a 0, 100% corresponde a 25 unidades de movimento
        float gripperPos = percentage * 25f;

        leftGripper.localPosition = new Vector3(0, gripperPos, 0);
        rightGripper.localPosition = new Vector3(0, -gripperPos, 0);
    }
}
```

Figure 3.18: C# code for linking the Robotiq gripper.

3.8 Gripper position and Virtual scenario representation

Initially, the application was developed and tested using a UR5e robot, as the physical workbench setup was still under construction. During this development phase,A convenient method for retrieving the gripper's position was identified by connecting directly to the Robotiq gripper port (63352). This method involved establishing a TCP/IP connection to the gripper and sending a specific command ("GET POS") to request its current position. The gripper would respond with its positional data, which was then read, validated, and parsed into a numerical value for use in the application. If the response was malformed or unexpected, an exception was raised. The C# code for this initial approach to get the gripper position is shown in Figure 3.19. The overall scene used in the project, including the virtual workbench and robot, is depicted in Figure 3.20.

```csharp
private async Task<int> GetGripperPosition(string host, int port)
{
    using (TcpClient client = new TcpClient())
    {
        await client.ConnectAsync(host, port);
        NetworkStream stream = client.GetStream();

        byte[] sendBuffer = Encoding.ASCII.GetBytes("GET POS\n");
        await stream.WriteAsync(sendBuffer, 0, sendBuffer.Length);

        byte[] receiveBuffer = new byte[256];
        int bytesRead = await stream.ReadAsync(receiveBuffer, 0, receiveBuffer.Length);
        string response = Encoding.ASCII.GetString(receiveBuffer, 0, bytesRead).Trim();

        if (response.StartsWith("POS ") && int.TryParse(response.Substring(4), out int position))
            return position;

        throw new Exception("Resposta inválida: " + response);
    }
}
```

Figure 3.19: C# code for the 1st approach go get the gripper position.

However, when testing transitioned to the final workbench setup, which uses a UR3e robot, this approach proved incompatible. As a result, a different strategy was subsequently adopted, inspired by the method used for configurable input/output communication. A custom script was developed to run on the robot using URScript, which retrieves the gripper's position and sends it over a socket connection to the HoloLens 2. This script runs on a separate thread, allowing it to operate alongside the main robot program without causing delays or interruptions 3.21. On the HoloLens side, a listener receives this data in real time for use within the application.

The GripperPositionDisplay script is responsible for establishing real-time integration between data received from the physical robotic gripper and its virtual representation. The main goal is to visually replicate the gripper's behavior in an interactive virtual environment, allowing users to observe the progression of a simulated assembly process.

The system operates by receiving continuous data from the robot via a TCP connection. Each incoming value corresponds to the current position of the gripper (ranging from 1 to 49), which is used to determine whether the gripper is open or closed. These transitions (opening and closing) trigger specific actions in the simulation, such as picking up an object from the ground, displaying the object inside the gripper, and later placing it in a final position to simulate an assembly or palletization process. The C# code demonstrating how boxes are managed in the scenario is provided in Figure 3.22.

The full cycle is divided into four sequential steps, each representing a stage in a virtual assembly task:

- When the gripper closes, it simulates grabbing a part by disabling the initial part on the floor and enabling a corresponding part within the gripper.
- When the gripper opens, it places that part in a predefined final position, disables the one inside the gripper, and prepares for the next step.

This continues until all steps are completed. On the final step, the simulation displays the fully assembled product in the gripper, and then places it in the palletizing area. After the final placement, the entire setup is reset and the cycle begins again.

To ensure responsiveness and visual accuracy, the gripper's state is updated as quickly as possible, without smooth transitions or delays. All game objects involved (parts on the floor, in the gripper, and in final positions) are activated or deactivated in real time based on the gripper's state.

The script also runs a separate background thread to handle TCP communication. This ensures that receiving data from the robot does not interfere with Unity's main thread, keeping the simulation responsive and fluid.

Overall, this setup allows for a tightly synchronized, real-time simulation that reflects the physical robot's behavior with high fidelity.

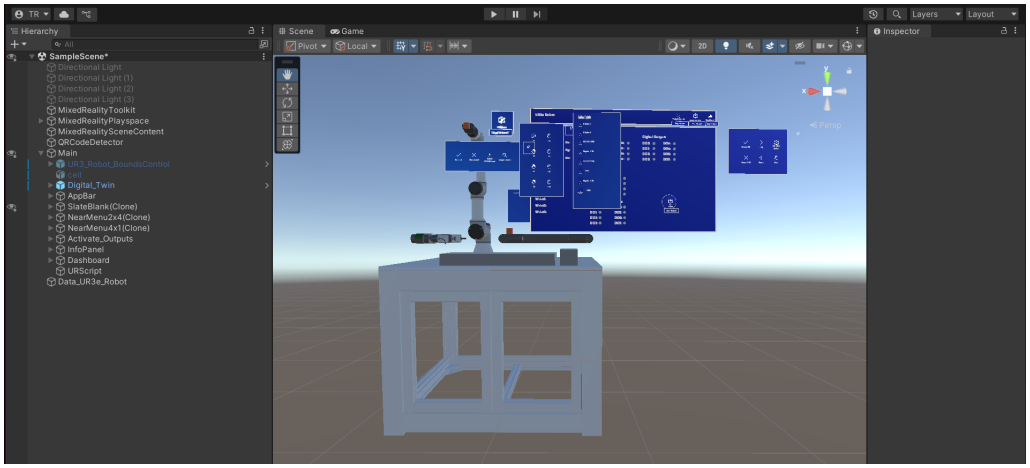


Figure 3.20: Scene used in the project.

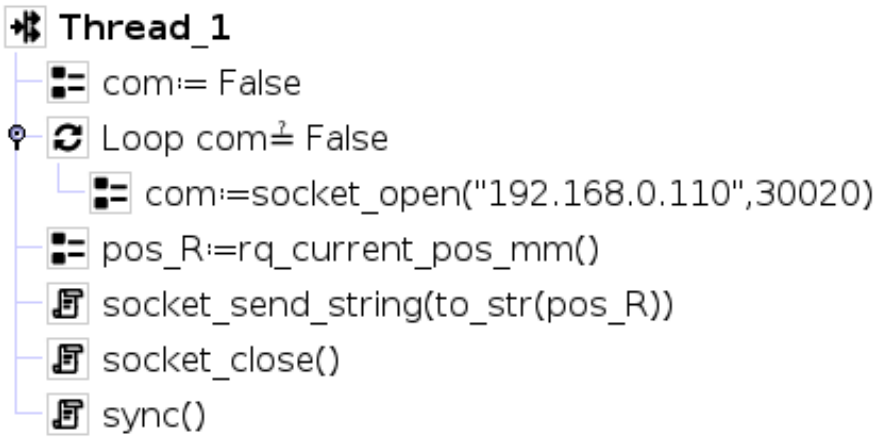


Figure 3.21: Robot script to get the gripper position.

```
private void AtualizaEstadoPecas(int gripperPos)
{
    bool gripperAberto = gripperPos >= gripperOpenThreshold;
    bool mudouParaAberto = gripperAberto && gripperAnteriorFechado;
    bool mudouParaFechado = !gripperAberto && !gripperAnteriorFechado;
    gripperAnteriorFechado = !gripperAberto;

    if (mudouParaFechado)
    {
        switch (estadoCiclo)
        {
            case 0:
                peca1Chao?.SetActive(false);
                peca1Gripper?.SetActive(true);
                break;
            case 1:
                peca2Chao?.SetActive(false);
                peca2Gripper?.SetActive(true);
                break;
            case 2:
                peca3Chao?.SetActive(false);
                peca3Gripper?.SetActive(true);
                break;
            case 3:
                prefabPecaFinal?.SetActive(false);
                peca4Gripper?.SetActive(true);
                break;
        }
    }
    else if (mudouParaAberto)
    {
        // Limpa tudo
        peca1Gripper?.SetActive(false);
        peca2Gripper?.SetActive(false);
        peca3Gripper?.SetActive(false);
        peca4Gripper?.SetActive(false);
        prefabPeca1Final?.SetActive(false);
        prefabPecaUnida?.SetActive(false);
        prefabPecaFinal?.SetActive(false);
        pecaFinalPaletizacao?.SetActive(false);

        // Ativa posição final correspondente
        switch (estadoCiclo)
        {
            case 0:
                prefabPeca1Final?.SetActive(true);
                break;
            case 1:
                prefabPecaUnida?.SetActive(true);
                break;
            case 2:
                prefabPecaFinal?.SetActive(true);
                break;
            case 3:
                pecaFinalPaletizacao?.SetActive(true);
                break;
        }

        estadoCiclo = (estadoCiclo + 1) % 4;

        if (estadoCiclo == 0)
            StartCoroutine(DelayEReiniciarPecas());
    }
}
```

Figure 3.22: C# code to understand how the boxes are managed in the scenario.

3.9 AR interface

When developing an application, especially one that integrates complex systems such as a digital twin, it is crucial to prioritize user experience by designing an interface that is both intuitive and informative. To support this, a custom user interface was developed to complement the visual representation of the digital twin. This interface facilitates a clearer understanding of the data extracted from the physical system by presenting it in a structured and accessible format.

Furthermore, it enables interaction with the augmented reality environment and, more specifically, with the robot itself. This interactive component significantly enhances user engagement and contributes to a more immersive and meaningful experience, which is a key factor in the overall effectiveness and appeal of the application.

3.9.1 Elements

Start Menu Bar

After reading the QR code, the menu that appears alongside the scenario is the application's main menu bar, which serves as the central control panel. As shown in Figure 3.23, it allows the user to connect and disconnect the robot using flags defined in the data processing script mentioned earlier. The menu also enables or disables other panels, such as the Robot Management menu (also referred to as the slate bar, discussed later), and includes the button to activate the Inspect Robot feature.

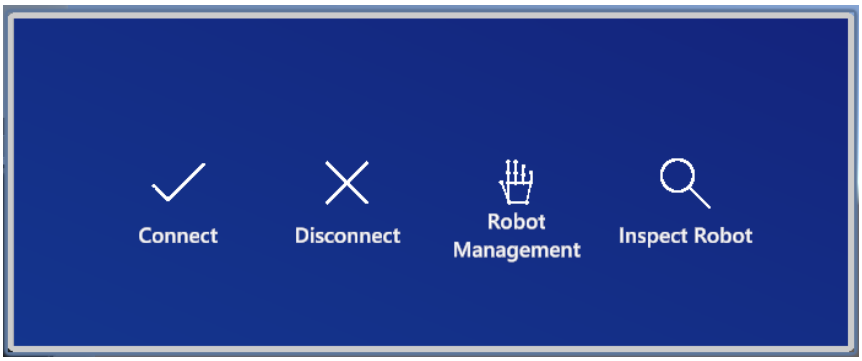


Figure 3.23: Start Menu Bar.

Inspect Robot feature

A second scenario, similar to the first one, was developed to allow the user to observe the process while being in a different location from the workbench. This is a valuable feature, as it enables remote monitoring of the operation without needing to be physically there. This second virtual scenario is presented in Figure 3.24.

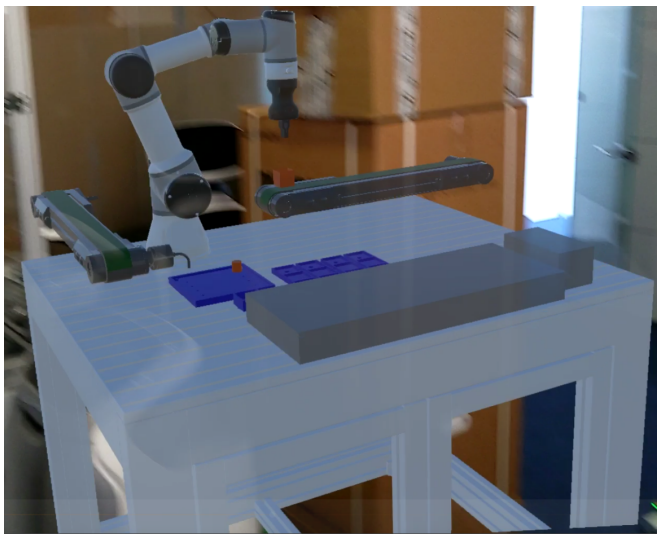


Figure 3.24: Second virtual scenario.

Slate Bar

A slate bar was developed to process and display data received from the robot. The table 3.1 presents all the values shown when the app is running. The visual representation of the slate bar with robot values and relevant information is provided in Figure 3.25.

Elements	Type	Representation
Robot Name	string	UR3e Robot
TCP Position	float	X: 0.00 Y: 0.00 Z: 0.00
TCP Orientation	float	Rx: 0.00 Ry: 0.00 Rz: 0.00
Digital Inputs	BOOL	DI0 : green/red DI4 : green/red DI1 : green/red DI5 : green/red DI2 : green/red DI6 : green/red DI3 : green/red DI7 : green/red
Digital Outputs	BOOL	DO0 : green/red DO4 : green/red DO1 : green/red DO5 : green/red DO2 : green/red DO6 : green/red DO3 : green/red DO7 : green/red
Configurable Inputs	BOOL	DI0 : green/red DI4 : green/red DI1 : green/red DI5 : green/red DI2 : green/red DI6 : green/red DI3 : green/red DI7 : green/red
Configurable Outputs	BOOL	DO0 : green/red DO4 : green/red DO1 : green/red DO5 : green/red DO2 : green/red DO6 : green/red DO3 : green/red DO7 : green/red
Digital Output Panel	Button	A button that enables the Digital Outputs Control Panel
Robot Control	Button	A button that enables the Robot Control Panel
Dashboard	Button	A button that enables the Dashboard Panel
Eye Gazing	Button	A button that enables Eye Gazing
Gripper Position	float	Gripper Position: 1 (closed) to 49 (opened)
Program Status	string	PROGRAM STATUS: PAUSED/PLAYING/STOPPED

Table 3.1: Elements shown in the slate bar with their types and representations.

In both digital and configurable inputs and outputs, the LED indicates their status: a red LED means the input or output is off, while a green LED means it is on. Status of configurable inputs

and outputs is only available when the robot program is running.

Figure 3.25: Slate bar with the robot values and relevant information.

Dashboard

The Dashboard is a control interface that enables remote operation of the robot by sending predefined commands through a TCP/IP socket on port 29999 [58]. This functionality is only available when the robot is set to remote control mode. The Dashboard interface is shown in Figure 3.26.

In this implementation, the Dashboard consists of six buttons, each assigned to a specific function:

1. POWER ON – Powers up the robot.
2. POWER OFF – Shuts down the robot.
3. Play – Starts the execution of the robot program.
4. Pause – Temporarily halts the program. Pressing Play again resumes execution from the point at which it was paused.
5. Stop – Stops the currently running program and resets it to the beginning.
6. Initialize Robot – Unlocks the robot and releases its brakes, preparing it for operation.

This interface allows for basic yet essential robot control without requiring physical interaction, making it particularly valuable in remote or virtual commissioning contexts.

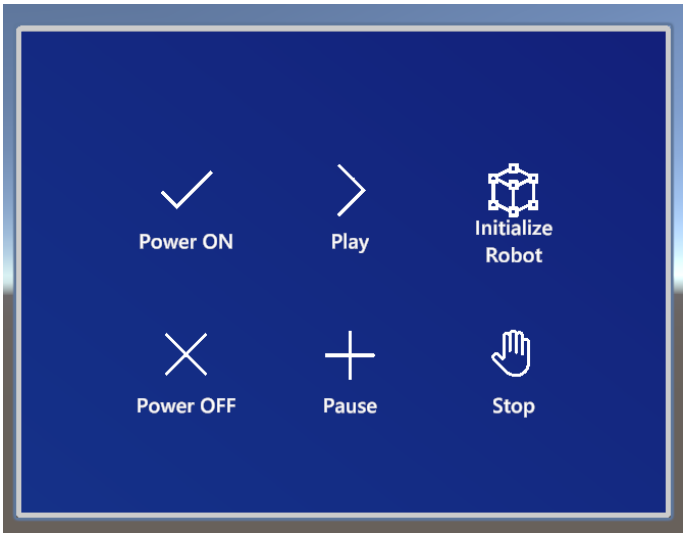


Figure 3.26: Dashboard that controls remotely the robot.

Robot Control

Robot Control can only be operated when the program status is either PAUSED or STOPPED. Otherwise, the control buttons are disabled and not visible in the Slate Bar, as shown in Figure 3.25. The Robot Control Panel is linked to a script that sends URScript commands. Using the previously created UR_Control class, a connection is established on port 30003 to transmit these commands. Pressing each button triggers movement along the corresponding robot axis.

The panel contains eight buttons: six control the tool's position along different axes and two control the most important tool's rotation, as shown in 3.29. Each button sends parameters indicating direction: negative values move the tool along the negative axis while positive values move it along the positive axis. As illustrated in Figure 3.27, these parameters are defined, aceleration and time included, before being sent with the command, as detailed in Figure 3.28.

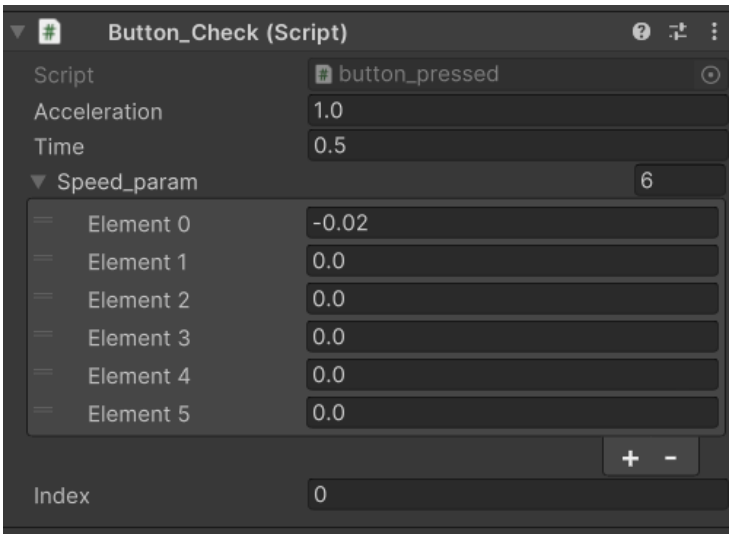


Figure 3.27: Defining the parameters for each value of X, Y, Z, RX, RY, and RZ.

```
public class Button_Check : MonoBehaviour
{
    public string acceleration = "1.0";
    public string time = "0.5";
    public string[] speed_param = new string[6] { "0.0", "0.0", "0.0", "0.0", "0.0", "0.0" };

    public int index;

    private UTF8Encoding utf8 = new UTF8Encoding();
    private Coroutine commandCoroutine;

    public void OnTouchStarted(HandTrackingInputEventData eventData)
    {
        ur_data_processing.UR_Control_Data.button_pressed[index] = true;
        commandCoroutine = StartCoroutine(SendSpeedCommandLoop());
    }

    public void OnTouchCompleted(HandTrackingInputEventData eventData)
    {
        ur_data_processing.UR_Control_Data.button_pressed[index] = false;
        if (commandCoroutine != null)
        {
            StopCoroutine(commandCoroutine);
        }

        // Stop the robot by sending zero speed
        string stopCommand = "speedl([0,0,0,0,0,0], a=" + acceleration + ", t=0.1)\n";
        ur_data_processing.UR_Control_Data.command = utf8.GetBytes(stopCommand);
    }

    private IEnumerator SendSpeedCommandLoop()
    {
        while (ur_data_processing.UR_Control_Data.button_pressed[index])
        {
            string cmd = "speedl([" + string.Join(",", speed_param) + "], a=" + acceleration + ", t=" + time + ")\n";
            ur_data_processing.UR_Control_Data.aux_command_str = cmd;
            ur_data_processing.UR_Control_Data.command = utf8.GetBytes(cmd);

            yield return new WaitForSeconds(0.1f); // send periodically
        }
    }
}
```

Figure 3.28: C# script to send the URScript commands that moves the robot.

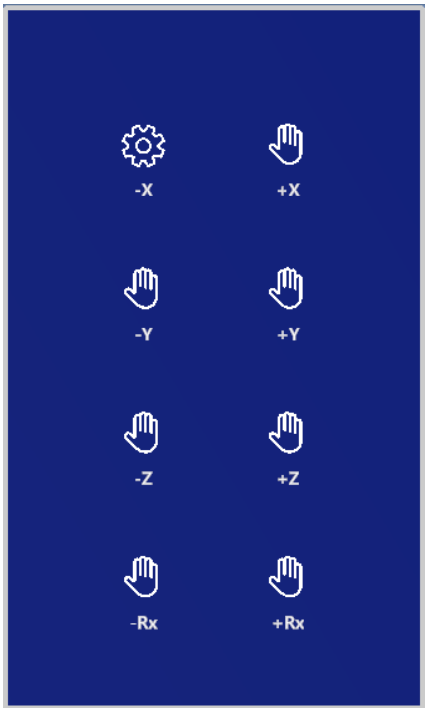


Figure 3.29: Robot Control Panel.

Managing the digital inputs/outputs

First of all, the configurable outputs should not be modified even if the robot is in remote control mode, as they are connected to critical components on the workbench that must not be altered. For this reason, only the digital outputs can be safely controlled through the interface.

This panel includes eight buttons, each corresponding to a digital output. Each button functions as a toggle switch: when the button displays "OFF," it indicates that the output is currently ON, and pressing it will turn it OFF. Conversely, if the button displays "ON," it means the output is OFF, and pressing it will turn it ON. This status is updated by using the information from 3.7.1.

The script used for this functionality is similar to the one implemented in the Robot Control script. It sends a URScript command that updates the status of the digital output based on its current state. The panel that remotely controls the digital inputs and outputs is shown in Figure 3.30.

Figure 3.30: Panel that remotely controls the digital inputs and outputs.

3.10 Testing and evaluation

The testing phase focused on assessing the functional reliability and usability of the developed AR system in a realistic industrial setting. The application was evaluated using the GrowSkills Educational Workstation, integrating the Universal Robots UR3e robotic arm and a HoloLens 2 device. Extensive testing was carried out.

Multiple test sessions were conducted to verify the integrity of real-time data synchronization, the responsiveness of the AR interface, and the system's ability to support typical industrial tasks such as assembly and object manipulation. These sessions involved scenarios like initiating robot commands, monitoring digital I/O states, and interacting with the virtual interface through hand gestures and gaze tracking. Each implemented feature was tested during the development process to ensure proper functionality before integrating it into the final system.

Although external evaluation was limited to one engineer, which may constrain the generalizability of the findings, the results demonstrated that the system successfully achieved its primary objectives. All critical functions, including digital twin synchronization, gripper visualisation, and robot control, performed with acceptable latency and reliability. No major failures or disconnections occurred, and feedback from testing highlighted the interface's usability and its potential value for operator training and process monitoring in industrial environments.

Chapter 4

Conclusion and Future Work

This dissertation presented the design and implementation of an augmented reality system aimed at improving human-robot interaction in industrial environments. By leveraging Microsoft HoloLens 2 and a collaborative robotic arm (UR3e), the developed application enables users to visualize and control a digital twin in real time. Through gesture and gaze-based interfaces, users can monitor robot states, visualize gripper positioning, and perform basic control tasks with intuitive, spatially anchored elements. The system was tested in a realistic industrial training environment using the GrowSkills Educational Workstation, and the results confirmed that the core functionalities, such as real-time synchronization, digital twin responsiveness, and user interaction, were successfully achieved.

Despite the limited number of testers, the feedback collected during these sessions highlighted the clarity of the interface, the effectiveness of the interaction methods, and the potential applicability of the tool in industrial training or operational support scenarios.

In addition, the development also revealed some performance limitations, particularly related to the efficiency of the application on the HoloLens 2. While the application ran reliably under normal use, it exhibited noticeable lag during more resource intensive operations, especially when simultaneous recording was enabled. This slowdown affected the overall smoothness of the experience and may present challenges in scenarios that require real-time responsiveness or documentation through mixed reality capture. These performance constraints, largely tied to the hardware limitations of the HoloLens 2, should be considered in future versions of the system, either through optimization of the Unity project or potential migration to more powerful AR platforms as they become available.

In conclusion, this project successfully demonstrates the potential of augmented reality to enhance interaction with collaborative robots in industrial environments. It contributes to the growing intersection of robotics, AR, and human-centered design, offering a foundation for future work

that can build upon its architecture, extend its capabilities, and refine its performance for broader deployment.

While the developed augmented reality system successfully demonstrates the integration of real-time data from a collaborative robot with a mixed reality interface, there remain several opportunities for future enhancements. One of the most immediate next steps would be to conduct broader user testing. In this dissertation, external testing was limited to a single engineer and although their feedback was valuable, a more comprehensive evaluation involving multiple users with varying technical backgrounds would help identify usability issues, optimize the interface, and validate the system's applicability in different industrial scenarios.

Additionally, expanding the system's interaction modalities could greatly improve its usability and accessibility. Currently, the system relies primarily on gesture and gaze input. Incorporating voice commands would allow for more intuitive, hands-free operation, especially in complex or multitasking environments. This could be particularly useful in noisy or constrained settings where hand gestures may not be practical.

Another promising direction involves improving the robustness of communication between the AR interface and the physical robot. While the current implementation over TCP/IP proved stable during tests, further enhancements, such as optimizing data packets or implementing additional error-handling mechanisms, could make the system more resilient in environments with high network latency or interference.

To increase the system's realism and training potential, integrating a simulation environment like Gazebo would be highly beneficial. Gazebo offers a physics-based 3D environment where robotic behaviors, including collisions, object manipulation, and sensor feedback, can be accurately simulated. This would enable developers and users to test and validate robot movements and interactions in a safe virtual setting before deploying them to the real hardware. Simulating scenarios such as grasping objects or avoiding obstacles in Gazebo could also serve as a development tool for refining control logic, visual feedback, and interaction flows in the AR interface. Furthermore, it could facilitate a more seamless transition between simulation and real-world deployment, bridging the gap between virtual prototyping and operational implementation.

Looking ahead, the system could also be scaled to support more complex industrial environments. For example, managing multiple robots within the same AR space, integrating additional sensors, or connecting to external cloud platforms for real-time data logging and analytics could greatly expand the system's applicability. Such developments would position the application not only as a training or support tool but also as part of a broader digital transformation strategy aligned with Industry 4.0 principles.

In summary, while the current work lays a strong foundation for AR-based interaction with collaborative robots, these proposed extensions, ranging from technical enhancements to simulation integration, would significantly enrich the system's functionality, robustness, and relevance in both educational and industrial contexts.

References

[1] United Nations. *The 17 Sustainable Development Goals*. https://sdgs.un.org/goals. Accessed: 27 June 2025. 2025.

[2] R.D. Vlahov Golomejić and T. Obradović Posinković. "A Systematic Literature Review of Industry 4.0 and Project Management". In: *European Project Management Journal* 13.2 (2023), pp. 51–62. DOI: 10.56889/xinw6398. URL: https://doi.org/10.56889/xinw6398.

[3] Frank Herrmann. "The Smart Factory and Its Risks". In: *Systems* 6.4 (2018). ISSN: 2079-8954. DOI: 10.3390/systems6040038. URL: https://www.mdpi.com/2079-8954/6/4/38.

[4] Plutomen. *9 Powerful Applications of Industry 4.0 Solutions*. 2025. URL: https://pluto-men.com/industry-40-solutions/.

[5] Julie Carmigniani et al. "Augmented reality technologies, systems and applications". In: *Multimedia Tools and Applications* 51.1 (2011), pp. 341–377.

[6] Fabio Arena et al. "An Overview of Augmented Reality". In: *Computers* 11.2 (2022), p. 28. DOI: 10.3390/computers11020028. URL: https://doi.org/10.3390/computers11020028.

[7] S Dargan, S Bansal, M Kumar, et al. "Augmented Reality: A Comprehensive Review". In: *Archives of Computational Methods in Engineering* 30 (2023), pp. 1057–1080. DOI: 10.1007/s11831-022-09831-7. URL: https://doi.org/10.1007/s11831-022-09831-7.

[8] Ricardo Buettner et al. "A Systematic Literature Review of Virtual and Augmented Reality Applications for Maintenance in Manufacturing". In: *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2022, pp. 544–552. DOI: 10.1109/COMPSAC54236.2022.00099.

[9] Jisu Kim et al. "Industrial Augmented Reality: Concepts and User Interface Designs for Augmented Reality Maintenance Worker Support Systems". In: *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. 2020, pp. 67–69. DOI: 10.1109/ISMAR-Adjunct51615.2020.00032.

[10] Niryo. *Collaborative Robots vs. Traditional Robots: A Comparative Guide*. Accessed: 13 June 2025. Apr. 2025. URL: https://niryo.com/collaborative-robots-vs-traditional-robots-comparative-guide/.

[11] Claudio Urrea and John Kern. "Recent Advances and Challenges in Industrial Robotics: A Systematic Review of Technological Trends and Emerging Applications". In: *Processes* 13.3 (2025). ISSN: 2227-9717. DOI: 10.3390/pr13030832. URL: https://www.mdpi.com/2227-9717/13/3/832.

[12] Pablo Jiménez, Víctor Pérez, and Carlos Martínez. “Advances and perspectives in collaborative robotics: a review of key technologies and emerging trends”. In: *Advanced Robotics* (2023). DOI: 10.1007/s44245-023-00021-8. URL: https://link.springer.com/article/10.1007/s44245-023-00021-8.

[13] Jaehyun Lee and Sanghoon Kim. “Collaborative robots in manufacturing and assembly systems: literature review and future research agenda”. In: *Journal of Intelligent Manufacturing* (2023). DOI: 10.1007/s10845-023-02137-w. URL: https://link.springer.com/article/10.1007/s10845-023-02137-w.

[14] Asmita Singh Bisen and Himanshu Payal. “Collaborative robots for industrial tasks: A review”. In: *Materials Today: Proceedings* 52 (2022). International Conference on Smart and Sustainable Developments in Materials, Manufacturing and Energy Engineering, pp. 500–504. ISSN: 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2021.09.263. URL: https://www.sciencedirect.com/science/article/pii/S2214785321061228.

[15] Cory Roehl. *Know Your Machine: Industrial Robots vs. Cobots*. Accessed: 13 June 2025. 2020. URL: https://www.universal-robots.com/blog/know-your-machine-industrial-robots-vs-cobots/.

[16] Wired Workers. *The Differences Between Industrial Robots and Collaborative Robots*. Accessed: 13 June 2025. June 2025. URL: https://www.wiredworkers.io/blog/the-differences-between-industrial-robots-and-collaborative-robots/.

[17] Universal Robots. *Applications - Universal Robots*. Accessed: 13 June 2025. URL: https://www.universal-robots.com/applications/.

[18] Universal Robots. *Collective Data Sheet*. https://www.universal-robots.com/media/1827367/05_2023_collective_data-sheet.pdf. Accessed: 27 February 2025. May 2023.

[19] Universal Robots A/S. *Produtos – Universal Robots*. https://www.universal-robots.com/pt/produtos/. Accessed: 30 June 2025. 2025.

[20] Alex Owen-Hill. *Parallel Robot Grippers: Which is Best?* Accessed: 17 April 2025. 2025. URL: https://blog.robotiq.com/adaptive-vs-parallel-grippers-which-is-best.

[21] Robotiq. *Adaptive Grippers*. Accessed: 17 April 2025. 2025. URL: https://robotiq.com/products/adaptive-grippers.

[22] Growskills. *Gripper 2F (85-140)*. Accessed: 26 May 2025. 2025. URL: https://growskills.pt/2f-85-2f-140-gripper/.

[23] Microsoft. *HoloLens 2 Hardware*. Accessed: 3 March 2025. 2025. URL: https://learn.microsoft.com/en-us/hololens/hololens2-hardware.

[24] Apple Inc. *Apple Vision Pro – Technical Specifications*. Accessed: 18 June 2025. 2025. URL: https://www.apple.com/apple-vision-pro/specs/.

[25] VR-Compare. *Compare HoloLens 2, Magic Leap 2, and Meta Quest Pro*. Accessed: 18 June 2025. 2025. URL: https://vr-compare.com/compare?h1=EkSDYv0cW&h2=0q3goALzg&h3=8aL1JxO3T.

[26] Meta Platforms, Inc. *Meta Quest 3 – Product Overview*. Accessed: 18 June 2025. 2025. URL: https://www.meta.com/quest/quest-3/?srsltid=AfmBOop5mdLzuhFXy-vbGEtMcjDEc81%5Chspace%7B0pt%7DWHFGVOJ9Y0cQRdO9-BybG-Bcn.

[27] Kinza Yasar and Linda Rosencrance. *What is a software development kit (SDK)?* https://www.techtarget.com/whatis/definition/software-developers-kit-SDK. Accessed: 21 June 2025. 2022.

[28] El Mostafa Bourhim and Aziz Akhiate. "Augmented Reality SDK's: A Comparative Study". In: *Intelligent Systems Design and Applications*. Ed. by Ajith Abraham et al. Cham: Springer International Publishing, 2022, pp. 559–566.

[29] Priyanka Fernandes et al. "Enhancing Architectural Visualization with AR: A Unity and Vuforia Approach". In: *ICT for Intelligent Systems*. Ed. by Jyoti Choudrie et al. Singapore: Springer Nature Singapore, 2024, pp. 409–420. ISBN: 978-981-97-6681-9.

[30] M. Fiala. "Comparing ARTag and ARToolkit Plus fiducial marker systems". In: *IEEE International Workshop on Haptic Audio Visual Environments and their Applications*. 2005. DOI: 10.1109/HAVE.2005.1545669.

[31] Zainab Oufqir, Abdellatif El Abderrahmani, and Khalid Satori. "ARKit and ARCore in serve to augmented reality". In: *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*. 2020, pp. 1–7. DOI: 10.1109/ISCV49265.2020.9204243.

[32] Dmytro M. Bodnenko et al. "Harnessing online services for creating augmented reality enhanced comics in primary education". In: *CEUR Workshop Proceedings* 3918 (2025), pp. 226–239.

[33] Zappar Ltd. *Zapworks – All-in-One WebAR Platform*. Accessed: 20 June 2025. 2025. URL: https://zap.works/.

[34] Microsoft Corporation. *MRTK-Unity Documentation (MRTK2)*. Accessed: 5 March 2025. 2022. URL: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05.

[35] Riccardo Palmarini et al. "A systematic review of augmented reality applications in maintenance". In: *Robotics and Computer-Integrated Manufacturing* 49 (2018), pp. 215–228. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2017.06.002. URL: https://www.sciencedirect.com/science/article/pii/S073658451730%5Chspace%7B0pt%7D0686.

[36] Sung Lae Kim et al. "Using Unity 3D to facilitate mobile augmented reality game development". In: *2014 IEEE World Forum on Internet of Things (WF-IoT)*. 2014, pp. 21–26. DOI: 10.1109/WF-IoT.2014.6803110.

[37] J. Lee. *Learning Unreal Engine Game Development*. Packt Publishing, 2016. URL: https://books.google.pt/books?id=RFpLDAAAQBAJ.

[38] Marc Stromberg. *Digital Twins for Smartest Process Optimization*. Accessed: 5 April 2025. 2025. URL: https://www.gbtec.com/resources/digital-twins/.

[39] Mohsin Raza et al. "A Digital Twin Framework for Industry 4.0 Enabling Next-Gen Manufacturing". In: *2020 9th International Conference on Industrial Technology and Management (ICITM)*. 2020, pp. 73–77. DOI: 10.1109/ICITM48982.2020.9080395.

[40] Yingying Du et al. "Industrial Robot Digital Twin System Motion Simulation and Collision Detection". In: *2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)*. 2021, pp. 1–4. DOI: 10.1109/DTPI52967.2021.9540114.

[41] Lorrainy Rembiski Delfino, Anilton Salles Garcia, and Ralf Luis de Moura. "Industrial Internet of Things: Digital Twins". In: *2019 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC)*. 2019, pp. 1–3. DOI: 10.1109/IMOC43827.2019.9317591.

[42] Lei Sun et al. “Dynamic Analysis of Digital Twin System Based on Five-Dimensional Model”. In: *Journal of Physics: Conference Series* 1486 (Apr. 2020), p. 072038. DOI: 10.1088/1742-6596/1486/7/072038.

[43] Devika Menon, Bharath Anand, and Chiranji Lal Chowdhary. “Digital Twin: Exploring the Intersection of Virtual and Physical Worlds”. In: *IEEE Access* 11 (2023), pp. 75152–75172. DOI: 10.1109/ACCESS.2023.3294985.

[44] AltexSoft Editorial Team. *Digital twins: Technology, use cases, and implementation tips*. https://www.altexsoft.com/blog/digital-twins/. Jan. 2025.

[45] Xianna Yang, Lei Yang, and Sijia Li. “A Digital Twin Application Framework in the Field of Industry”. In: *2023 IEEE 18th Conference on Industrial Electronics and Applications (ICIEA)*. 2023, pp. 1615–1619. DOI: 10.1109/ICIEA58696.2023.10241448.

[46] Dassault Systèmes. *SolidWorks – 3D CAD Design Software*. Accessed: 20 June 2025. 2024. URL: https://www.solidworks.com.

[47] Spencer Grey. 2021.

[48] Pranab Bandhu Nath and Md. Mofiz Uddin. “TCP-IP Model in Data Communication and Networking”. In: *American Journal of Engineering Research (AJER)* 4.10 (2015), pp. 102–107. ISSN: 2320-0936. URL: https://www.ajer.org/papers/v4(10)/N04101020107.pdf.

[49] Mohammed M. Alani. “TCP/IP Model”. In: *Guide to OSI and TCP/IP Models*. SpringerBriefs in Computer Science. Springer, 2014, pp. 19–50. DOI: 10.1007/978-3-319-05152-9_3. URL: https://link.springer.com/chapter/10.1007/978-3-319-05152-9_3.

[50] “ROSPlan: Planning in the Robot Operating System”. In: 25 (Apr. 2015), pp. 333–341. DOI: 10.1609/icaps.v25i1.13699. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/13699.

[51] R.M. Saidi, S.H.Y. Rizvi, and Y. Arif. “Automated hot wire cutting for complex 3D foam shapes with industrial robots”. In: *Robotics and Computer-Integrated Manufacturing* 41 (2016), pp. 115–124. DOI: 10.1016/j.rcim.2016.03.005.

[52] Microsoft Corporation. *Spatial Awareness Getting Started - Mixed Reality Toolkit*. Accessed: 3 March 2025. 2022. URL: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/spatial-awareness/spatial-awareness-getting-started?view=mrtkunity-2022-05.

[53] Universal Robots A/S. *Real-Time Data Exchange (RTDE) Guide*. Accessed: 10 March 2025. 2024. URL: https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/.

[54] A/S. *Realtime Client Interface G3/G5*. Accessed: 24 March 2025. Universal Robots A/S. 2021. URL: https://www.universal-robots.com.

[55] Daniel Constantin et al. “Forward Kinematic Analysis of an Industrial Robot”. In: *New Developments in Mechanics and Mechanical Engineering*. Vienna, Austria, 2015, pp. 90–95. DOI: 10.5281/zenodo.1234567. URL: https://www.researchgate.net/profile/Daniel-Constantin-4/publication/341353803_Forward_Kinematic_Analysis_of_an_Industrial_Robot/links/5ebbfb5945851562%5Chspace%7B0pt%7D6%5Cca6757c/Forward-Kinematic-Analysis-of-an-Industrial-Robot.pdf.

[56] RoboticsUnveiled. *Forward Kinematics using the Denavit–Hartenberg Convention (DH Parameters)*. Accessed: 24 May 2025. 2025. URL: https://www.roboticsunveiled.com/robotics-forward-kinematics-denavit-hartenberg-parameters/.

[57] Microsoft Corporation. *Eye Tracking Basic Setup – Mixed Reality Toolkit (MRTK) for Unity*. 2023. URL: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/input/eye-tracking/eye-tracking-basic-setup?view=mrtkunity-2022-05.

[58] Universal Robots A/S. *Dashboard Server – Communication Protocol*. 2024. URL: https://www.universal-robots.com/developer/communication-protocol/dashboard-server/.

Appendix A

Demonstration and Project Repository

Repository of the project scripts: Repository
Demo video of the AR application: Demo Video

Appendix B

Users' guide

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

GROWSKILLS

Guião passo a passo para aplicação HoloLens 2

Indíce

1.1 Introdução

Este documento serve como manual de instruções para aprender a testar e a utilizar a interface gráfica implementada no Unity para o HoloLens 2 que permite uma interação com o robot UR3e da Universal Robots, resultando num digital twin. Antes da leitura deste documento sugiro a leitura do manual de utilização do HoloLens2 para que seja possível entender como dar login na conta Microsoft, ligação à internet e configuração do *Eye Tracking Calibration*, fundamental para que seja possível reconhecer com precisão para onde estamos a olhar.

1.2 Objetivo

Este projeto foi elaborado no âmbito da Dissertação no mestrado de Engenharia Eletrotécnica e Computadores com o objetivo de desenvolver uma aplicação de Realidade Aumentada (RA) que facilite a interação com sistemas robóticos colaborativos em ambientes industriais. Através da utilização do Microsoft HoloLens 2 e da integração com o robô UR3e da Universal Robots, pretende-se proporcionar aos operadores uma interface intuitiva e imersiva que permita visualizar dados em tempo real, controlar equipamentos remotamente e receber assistência contextualizada durante a execução de tarefas. Esta aplicação visa aumentar a eficiência operacional, reduzir erros e melhorar a formação e manutenção em processos industriais, contribuindo para a modernização das práticas de fabrico no contexto da Indústria 4.0.

2. HoloLens 2

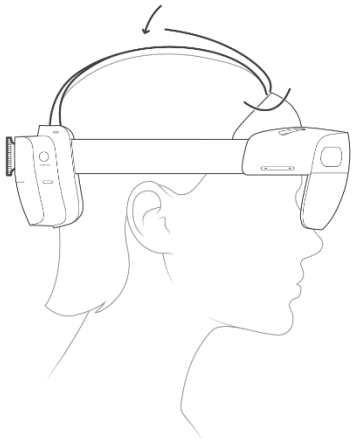
O Microsoft HoloLens 2 é um dispositivo de realidade aumentada (RA) que permite a sobreposição de elementos digitais no mundo real através de um visor transparente. Totalmente autónomo, sem fios ou necessidade de ligação a um computador, o HoloLens 2 oferece uma experiência imersiva baseada em interação por gestos, voz e seguimento ocular.

Este dispositivo é utilizado nesta aplicação como interface principal para visualização e controlo de um sistema robótico colaborativo, permitindo ao utilizador interagir com o robô UR3e em tempo real.

2.1 Interações básicas

2.1.1 Ajustar HoloLens 2 à cabeça

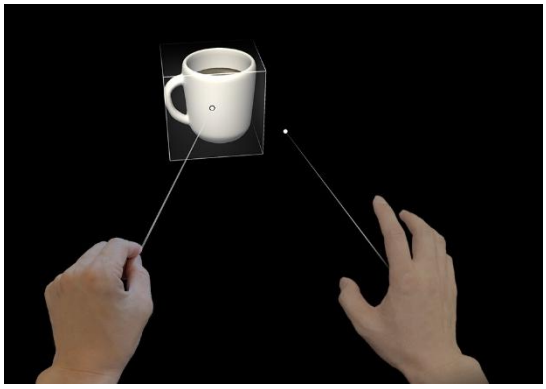
Segura no HoloLens 2 pelas laterais da armação (nunca pelo visor) e coloca-o cuidadosamente sobre a cabeça, como se fossem uns óculos normais. O visor deve ficar à frente dos olhos, mas sem tocar diretamente na cara. Usa a roda de ajuste localizada na parte de trás da fita para apertar ou alargar os óculos. Roda no sentido dos ponteiros do relógio para apertar e no sentido contrário para alargar. Garante que os óculos ficam firmes, mas confortáveis — não devem pressionar demasiado.



2.1.2 Hand Ray

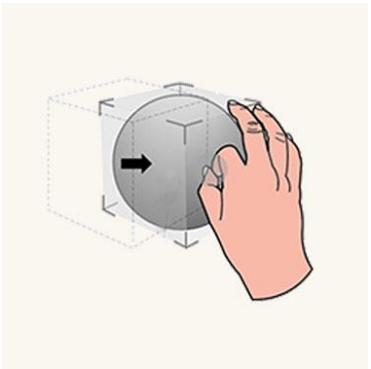
O HoloLens 2 permite aos utilizadores interagir com elementos virtuais através de gestos naturais, utilizando as mãos. Uma das principais formas de interação é através do raio da mão, também conhecido como *hand* ray.

O raio da mão é uma linha visível projetada a partir da palma ou ponta dos dedos quando a mão está estendida. Esta linha funciona como um ponteiro laser virtual, permitindo ao utilizador apontar para elementos da interface (botões, painéis, menus, etc.) mesmo à distância.



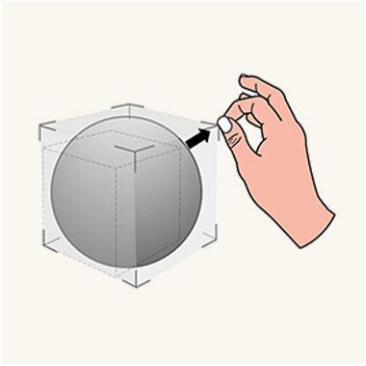
2.1.3. Agarrar e mover objeto

Para agarrar e mover um objeto virtual, aproxima a mão, fecha-a para agarrar, move-a para reposicionar o objeto e abre-a para o largar.



2.1.3 Redimensionar objetos

Agarre e utilize as alças de redimensionamento que aparecem nos cantos dos objetos 3D para conseguir redimensioná-los.



2.1.4. Pressionar botões

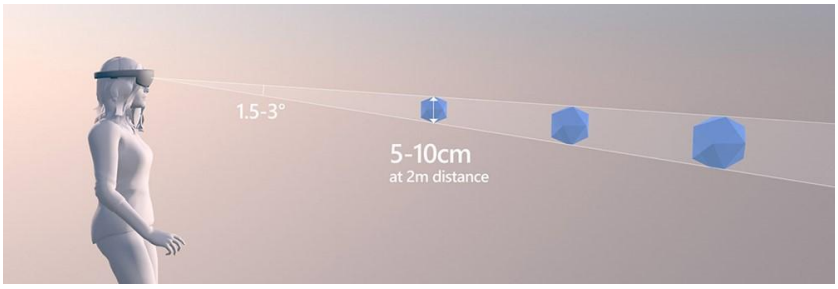
Para pressionar botões pode utilizar o gesto de selecionar como se de um botão físico se tratasse.

2.1.5. Menu Iniciar

Para abrir o Menu Iniciar deve colocar a palma da sua mão virada para si e aparecerá um ícone no seu pulso. Deve clicar no mesmo para que o menu abra.

2.1.6. Eye Gazing

Com o eye gazing, pode controlar e interagir com elementos da interface apenas com o movimento dos olhos, facilitando a navegação e seleção no HoloLens 2.

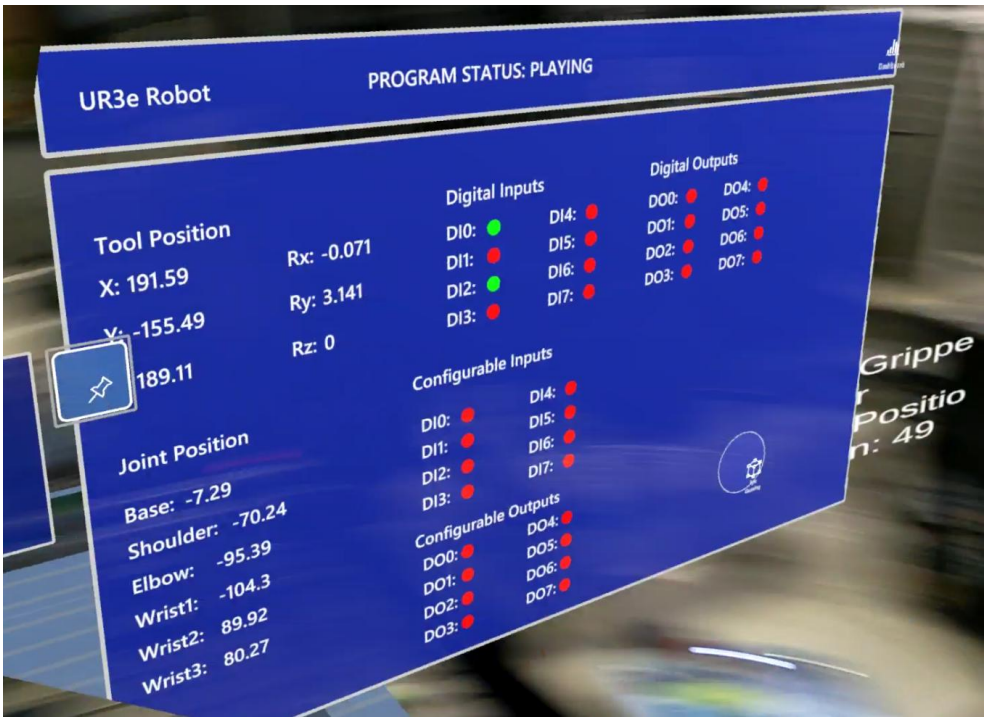


3. Como utilizar a aplicação

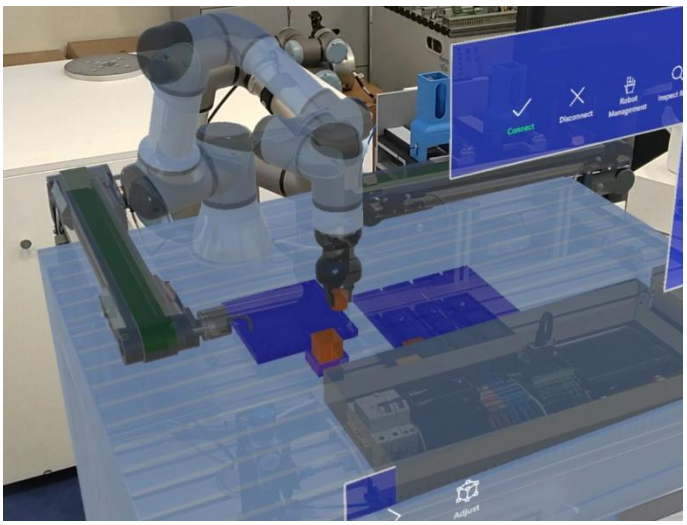
3.1. Elementos:

- Near menu

- Slate Bar com as informações do Robot



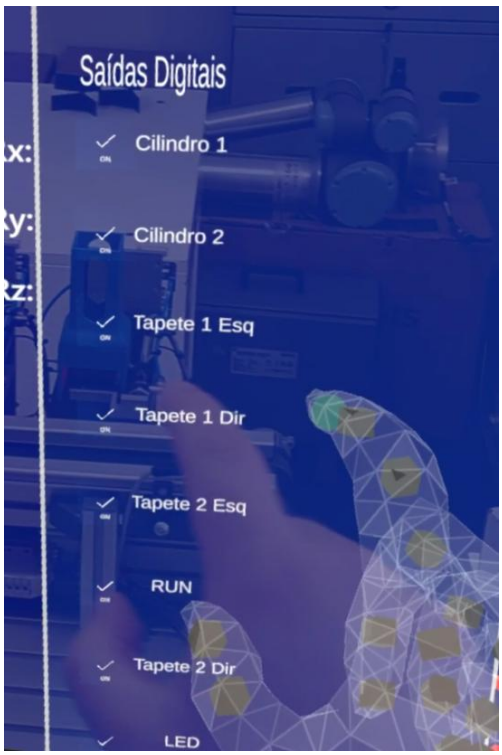
- 3D Cenário



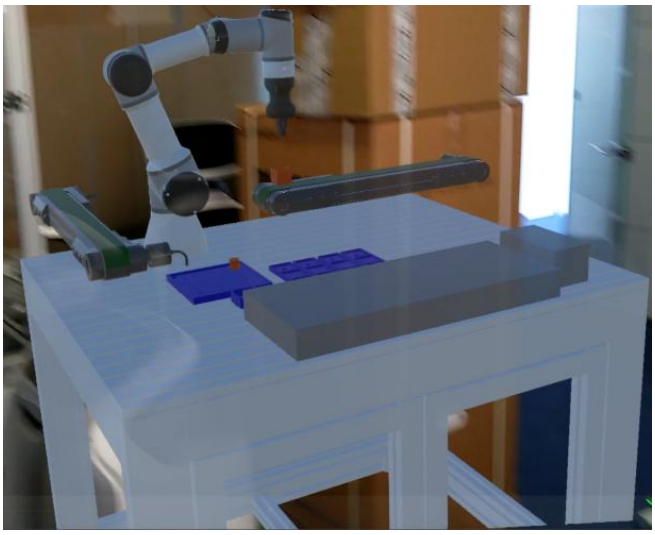
- Dashboard

- Controlo do robô

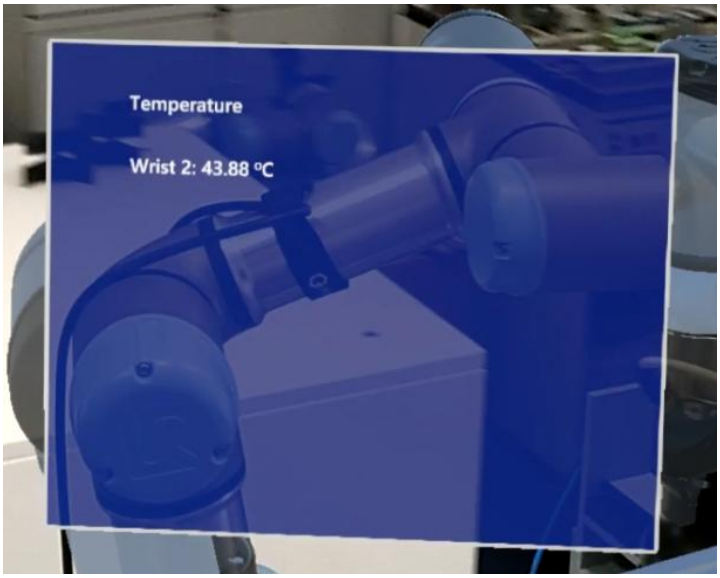
- Controlo das saídas digitais



- Segundo Cenário (botão nspect Robot)



- Eye gazing panel



3.2 Instruções

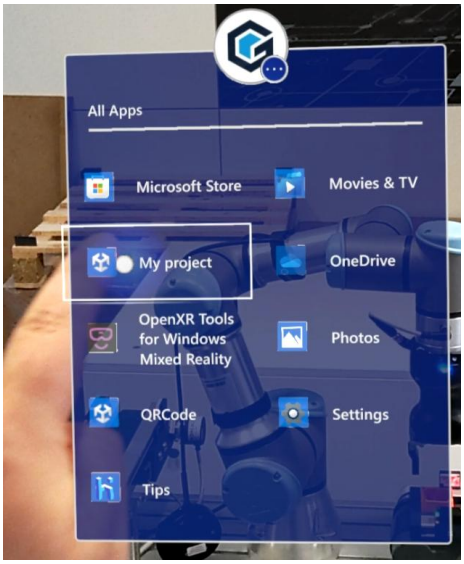
1. Antes de iniciar a aplicação é importante configurar o Robot. Para isso deve seguir os seguintes passos:
    1) Ligar o robot e destravar os travões.
    2) Abrir o script "Gripper 2" que contém toda a informação necessária, assim como o programa principal.
    3) No canto superior direito clicar no botão da UR e colocar o Robot em modo remoto.

2. Coloque o dispositivo na cabeça.

3. Abra o Menu Iniciar.

4. Carregue em "All Apps".

5. Procure e carregue na aplicação "My Project".



6. Depois que a aplicação inicia, aproxime-se do código QR de modo que o pequeno círculo no centro da tela esteja num canto do código. Não esquecer que o dispositivo deve estar o mais na direção do robô possível, paralelo ao chão.

7. Aparecerá o cenário assim como o near menu. Deve clicar no botão *connect* para estabelecer a conexão com UR3 e o braço do Robot deve posicionar-se na mesma posição que o Robot físico.

8. De seguida tem 2 opções: *Inspect Robot* para criar um segundo cenário que acompanha o que está a acontecer no cenário principal que pode manusear, rodar, redimensionar e analisar esse elemento como preferir *ou Robot Management* onde aparece a Slate Bar com todas as informações do robô. É possível reposicionar os elementos tal como foi explicado anteriormente.

9. Dentro do Menu do Slate bar, na parte superior direita temos 3 botões e o estado do programa do robô. Na parte frontal, temos um botão denominado "Eye Gazzing" que ativa e desativa a função que permite olhar para as juntas do robô e observar informação da própria junta.

Se o estado estiver "STOPPED" ou "PAUSED" é possível aceder aos botões de "Robot Control" e "Digital Outputs Control", caso contrário apenas o botão da Dashboard está visível.

10. O menu do Robot Control permite-nos mover o robô no sentido do eixo que o botão representa. (+x, -x, -y, +y, -z, +z, -rx, +rx)

11. O menu "Digital Outputs Control" permite controlar todas as saídas digitais de forma manual assim como perceber o seu estado. Se o botão indicar ON significa que a saída está desligada e precisa de pressionar para o ativar, se o botão indicar OFF significa que a saída está ligada e precisa de o pressionar para desligar.

12. O menu "Dashboard" permite ligar o robô (Power ON), desligar o robô (POWER OFF), retirar os travões (Initialize Robot), começar o programa (START), pausar o programa (PAUSE) e parar o programa (STOP). Durante a execução do programa do robô garanta que o programa está no início, de outra forma os comandos enviados pela dashboard não irão funcionar.

13. Só é possível reiniciar a aplicação fechando-a no menu iniciar e voltando a abrir, repetindo todo o processo.