

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Human-robot collaborative workspaces supported by AR head-mounted displays

Vítor Hugo Tavares Santos

MASTER'S THESIS REPORT



Master's Degree in Multimedia, Specialization in Interactive Technologies and Digital Games

Supervisor: Marcelo Petry (PhD)

October 28, 2025

Human-robot collaborative workspaces supported by AR head-mounted displays

Vítor Hugo Tavares Santos

Master's Degree in Multimedia, Specialization in Interactive
Technologies and Digital Games

October 28, 2025

Abstract

The integration of humans and robots in shared environments raises critical challenges in ensuring both safety and usability. In emerging collaborative workspaces, operators are expected to interact closely with robotic systems, which requires solutions that foster trust, efficiency, and intuitive interaction. This dissertation addresses these challenges by exploring how Augmented Reality can be employed not only as a visualization medium but also as an active interface for safe human–robot collaboration.

A system was developed for a collaborative manipulator that combines dynamic collision avoidance, holographic previews of robot trajectories, and an interactive user interface. The implementation included mechanisms to detect and respond to the proximity of the operator’s hands and head, as well as a tutorial and exploratory mode to facilitate learning and testing. Validation was conducted in two stages: controlled experiments compared alternative execution strategies in terms of efficiency, continuity, and safety, while a user study with 15 participants evaluated usability and workload through standardized questionnaires.

The results demonstrated that the proposed approach enables fast and fluid robot execution without compromising safety. In a pre-programmed task, the dynamic system achieved an average task completion time of 15.15 seconds, outperforming both the fixed zone configuration (20.12 seconds) and the alternative system (24.27 seconds). Participants reported that the system was easy to learn, consistent, and enjoyable to use, with generally low workload ratings and full task success across all trials. The System Usability Scale score reached 84.0, reflecting excellent usability. Qualitative feedback emphasized trust in the safety mechanisms and strong engagement with the interface. These findings suggest that Augmented Reality can effectively support both safety and usability in collaborative workspaces, offering a robust foundation for future industrial integration.

Keywords: Human–Robot Collaboration, Augmented Reality, Safety, Collision Avoidance, Usability

Resumo

A integração de humanos e robôs em ambientes partilhados levanta desafios críticos no que respeita à garantia de segurança e usabilidade. Nos espaços de trabalho colaborativos emergentes, espera-se que os operadores interajam de forma próxima com sistemas robóticos, o que exige soluções que promovam confiança, eficiência e uma interação intuitiva. Esta dissertação aborda estes desafios ao explorar como a Realidade Aumentada pode ser utilizada não apenas como um meio de visualização, mas também como uma interface ativa para uma colaboração segura entre humanos e robôs.

Foi desenvolvido um sistema para um manipulador colaborativo que combina desvio dinâmico de colisões, pré-visualizações holográficas de trajetórias do robô e uma interface interativa. A implementação incluiu mecanismos para detetar e responder à proximidade das mãos e da cabeça do operador, bem como um tutorial e um modo de exploração livre para facilitar a aprendizagem e os testes. A avaliação foi conduzida em duas etapas: experiências controladas compararam estratégias alternativas de execução em termos de eficiência, continuidade e segurança, enquanto um estudo com 15 participantes avaliou a usabilidade e a carga de trabalho através de questionários normalizados.

Os resultados demonstraram que a abordagem proposta permite uma execução rápida e fluida do robô sem comprometer a segurança. O sistema dinâmico obteve um tempo médio de conclusão de uma tarefa pré-programada de 15,15 segundos, superando tanto a configuração de zona fixa (20,12 segundos) como o sistema alternativo (24,27 segundos). Os participantes relataram que o sistema era fácil de aprender, consistente e agradável de utilizar, com classificações de carga de trabalho geralmente baixas e sucesso total em todas as tentativas. A pontuação no System Usability Scale atingiu 84,0, refletindo uma excelente usabilidade. O feedback qualitativo destacou a confiança nos mecanismos de segurança e o forte envolvimento com a interface. Estes resultados sugerem que a Realidade Aumentada pode apoiar eficazmente tanto a segurança como a usabilidade em ambientes de trabalho colaborativos, oferecendo uma base robusta para futura integração industrial.

Palavras-chave: Colaboração Humano-Robô, Realidade Aumentada, Segurança, Evitamento de Colisões, Usabilidade

Acknowledgements

I would like to express my deepest gratitude to all those who supported me throughout the development of this dissertation.

First and foremost, I am sincerely thankful to researcher Gabriel Moura for his invaluable help in establishing the connection between application and robot. His support was fundamental to bridging the work I was developing with the real robot. Many times, he put aside his own projects to assist me, and his dedication and generosity were of immense value. I am especially grateful for all the knowledge he shared with me, which had a lasting impact on this work.

I also wish to thank researcher João Martins for all the time and guidance he dedicated to me. He provided crucial insights for the design of the algorithm, particularly in the early stages with the batch execution approach, as well as in the conversion of the UR30 model axes to real-world parameters for Blender. His constant availability and motivation were a great encouragement.

My heartfelt thanks go as well to my supervisor, Professor Marcelo Petry, for his continuous guidance, valuable suggestions, and close accompaniment throughout this journey. Without his initiative, this research topic would not have been possible, and his willingness to work with me from day one was of great significance to the success of this dissertation.

I would also like to acknowledge the two laboratories at INESC TEC, iiLab and TRIBE Lab, for providing the resources, materials, and workspace that made the realization of this project possible.

Finally, I am deeply grateful to my family, partner, and friends, who continuously motivated me, never allowed me to give up, and supported me unconditionally throughout my academic journey. I owe them more than words can express.

Vítor Santos

“Journey before destination.”

Brandon Sanderson

Contents

1	Introduction	1
1.1	Motivation and Context	1
1.1.1	The Rise of Human–Robot Collaboration (HRC) in Industry 4.0	1
1.1.2	Industry 5.0: A Human-Centric Evolution	1
1.1.3	Challenges in Human–Robot Interaction	1
1.1.4	Augmented Reality as an Enabler for HRC	2
1.1.5	Motivation	2
1.2	Objectives and Research Questions	3
1.3	Contributions	4
1.4	Thesis Structure	4
2	Foundational Concepts	5
2.1	Industry 4.0 and Industry 5.0 in Collaborative Robotics	5
2.2	Collaborative Robotics and Safety Standards	6
2.3	Human–Robot Interaction	6
2.4	Mixed Reality and Augmented Reality	8
2.4.1	AR Interfaces and Display Modalities	9
2.5	Tracking Methods in AR for HRC	9
2.5.1	Marker-Based Tracking	9
2.5.2	Markerless Tracking	10
2.5.3	Hybrid Approaches	10
2.5.4	Multimodal and Future Directions	10
2.6	Digital Twins and Simulation	10
2.7	Human Factors in HRC	11
2.8	Networking and Real-Time Communication	12
2.8.1	Robot Communication Interfaces	12
2.8.2	Latency Requirements	12
2.8.3	Middleware and Synchronisation	13
2.8.4	Relation to Dynamic Safety	13
2.9	Conclusions	13
3	Literature Review	15
3.1	Safety Approaches in Human–Robot Collaboration	15
3.1.1	Static Safety Mechanisms	15
3.1.2	Dynamic Safety Mechanisms	16
3.2	Augmented Reality in HRC: Guidance, Visualization, and Safety	18
3.2.1	AR for Programming & Guidance	18
3.2.2	AR for Trajectory Visualization	18

CONTENTS

3.3	Path Planning, Prediction & Simulation for HRC	19
3.3.1	Unity-Based Path Planning Platforms	19
3.4	Conclusions of the Literature Review	20
4	System Design and Architecture	23
4.1	Motivation and Scenario	23
4.2	Design Requirements	25
4.2.1	Functional Requirements	25
4.2.2	Non-Functional Requirements	26
4.2.3	System Constraints	26
4.3	User Story and Workflow	26
4.4	System Architecture	28
4.5	Summary	29
5	Implementation	31
5.1	Introduction	31
5.2	Development Environment	31
5.2.1	Hardware	32
5.2.2	Software	32
5.2.3	Networking Setup	32
5.3	Robot Communication Layer	33
5.3.1	Script Overview: <code>ur_data_processing</code>	33
5.4	Holographic robotic arm	34
5.5	Waypoint Generation Algorithm	37
5.5.1	Base Obstacle (static arc detour)	38
5.5.2	Human Obstacles (dynamic vertical lift)	38
5.5.3	Rationale for the two approaches	39
5.6	User Interface	39
5.6.1	Robot Menu	40
5.6.2	Waypoint Editor	43
5.6.3	Status Bar	44
5.6.4	Tutorial Guidance	44
5.7	Safety System	45
5.7.1	Proximity Detection via Colliders	45
5.7.2	Dynamic End-Effector Detours	48
5.8	Superposition of holograms via QR Markers	49
6	Validation	53
6.1	Methodology	53
6.1.1	Methodological Alignment with Research Questions	53
6.1.2	Variables and Metrics	53
6.1.3	Participants	54
6.1.4	Equipment	54
6.2	Experimental Validation: Program Efficiency	54
6.2.1	Metrics and Data Collection	54
6.2.2	Results and Discussion	55
6.2.3	Conclusion of Experiment 1	56
6.3	Experimental Validation: Participant Study	56
6.3.1	System Usability Scale	57

CONTENTS

6.3.2	NASA-TLX Workload Assessment	58
6.3.3	Impact of User AR Experience	58
6.3.4	Observed Behavioral Patterns	58
6.3.5	Conclusions of Experiment 2	59
7	Conclusions and Future Work	61
7.1	Conclusions	61
7.2	Main Contributions	62
7.3	Future Work	63
7.4	Final Remarks	63
	References	65
A	Demonstration Video	69
B	System Usability Scale - Answers	71
C	NASA-TLX - Answers	77
D	Open feedback from participants	81

CONTENTS

List of Figures

2.1	Cooperation levels between human and robot	7
2.2	Workers' and robots' workspaces	8
3.1	Projected UR5 and ABB IRB 2600 workspaces	16
3.2	Safety zone arrangements example	17
3.3	Dynamic safety zone example and available free space	17
3.4	Interactive generation of gradient fields in HRC	17
3.5	Phases of the cobots' workflow	19
3.6	Virtual robot workspace aid	20
4.1	System context showing the interaction between operator, AR application, robotic arm, and QR-based calibration.	25
4.2	Workflow of the AR-based collaborative system from calibration to trajectory execution. The flowchart illustrates the operator's interaction sequence, including calibration, connection, waypoint authoring, preview and execution.	28
5.1	UR30 link structure and DH parameters [Rob23]	36
5.2	UR30 ghost arm adapted in Blender and pre-flipped for correct Unity import. . .	37
5.3	Exemplification of the waypoint generation algorithm	37
5.4	Dynamic avoidance example: A) Path with arc around the base; B) Path with arc around the base and avoidance on the MRTK2 hand.	39
5.5	Robot menu during experimental test with Microsoft HoloLens 2	40
5.6	Waypoint editor during experimental test with Microsoft HoloLens 2	40
5.7	Connection panel of the AR interface. (A) Disconnected state, where the status box is shown in red to indicate loss of connection. (B) Connected state, where the status box turns green to confirm a successful TCP/IP connection with the UR30.	41
5.8	Diagnostics panel of the AR interface, showing real-time feedback from the UR30. The panel displays the Cartesian position and orientation of the TCP (X, Y, Z, RX, RY, RZ), as well as the six joint angles (J1–J6). This transparency allows the operator to validate the robot's current state and verify that data streaming from port 30013 is functioning correctly.	42
5.9	Test Move panel of the AR interface, implemented as a joystick-style control. The operator can issue incremental <code>speed1</code> commands along the Cartesian axes (X, Y, Z) and rotations (RX, RY, RZ). This feature provides a simple way to verify connectivity and responsiveness of the UR30 before executing complex trajectories.	42
5.10	Waypoint Editor with Status Bar in Unity. The interface enables operators to create, preview, and execute trajectories using the ghost arm, with safety ensured through an immediate stop command.	43

LIST OF FIGURES

5.11	Status bar of the AR interface. It provides persistent feedback on system readiness, showing either a "Ready" state or a "No Waypoint" state depending on the current configuration.	44
5.12	Tutorial example in sequential order	45
5.13	Capsule colliders placed on the robot's links for proximity monitoring.	46
5.14	Protective stop triggered during system testing when a user places the hand close to a collider.	48
5.15	QR-based calibration as seen through the AR glasses. The system detects the pose of the marker in the world frame (WT_M) and combines it with a fixed offset to the robot base (MT_B), resulting in the robot base pose in world coordinates (WT_B). . .	51
6.1	Experimental validation: sample of participants and the environment in which the study was conducted.	57
A.1	Thumbnail of the demonstration video. Click the image to open the video online.	69
B.1	SUS Question 1: I think that I would like to use this system frequently.	71
B.2	SUS Question 2: I found the system unnecessarily complex.	72
B.3	SUS Question 3: I thought the system was easy to use.	72
B.4	SUS Question 4: I think that I would need support of a technical person to be able to use this system.	73
B.5	SUS Question 5: I found the various functions in this system were well integrated.	73
B.6	SUS Question 6: I thought there was too much inconsistency in the system. . . .	74
B.7	SUS Question 7: I would imagine that most people would learn to use this system very quickly.	74
B.8	SUS Question 8: I found the system very cumbersome/awkward to use.	75
B.9	SUS Question 9: I felt very confident using the system.	75
B.10	SUS Question 10: I needed to learn a lot of things before I could get going with this system.	76
C.1	NASA-TLX Question 1: Mental Demand - How mentally demanding was the task?	77
C.2	NASA-TLX Question 2: Physical Demand - How physically demanding was the task?	78
C.3	NASA-TLX Question 3: Temporal Demand - How hurried or rushed was the pace of the task?	78
C.4	NASA-TLX Question 4: Performance - How successful were you in accomplishing what you were asked to do?	79
C.5	NASA-TLX Question 5: Effort - How hard did you have to work to accomplish your level of performance?	79
C.6	NASA-TLX Question 6: Frustration - How insecure, discouraged, irritated, stressed, and annoyed were you?	80
D.1	Open feedback, comments or suggestions from participants of the experimental study.	81

List of Tables

3.1	Comparative summary of related works in HRC safety and AR visualization . . .	21
5.1	UR30 Denavit–Hartenberg parameters and dynamic properties [Rob23]	35
6.1	Experimental results for program efficiency validation on 20 waypoints	55

LIST OF TABLES

Abbreviations

AR	Augmented Reality
MR	Mixed Reality
VR	Virtual Reality
XR	Extended Reality
Cobots	Collaborative Robots
INESC TEC	Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência
iiLab	Industry and Innovation Laboratory
HMD	Head-Mounted Display
HRC	Human-Robot Collaboration
HRIs	Human-Robot Interactions
HHD	Hand-Held Displays
IMU	Inertial Measurement Units
I/O	Input/Output
IoT	Internet of Things
UR	Universal Robots
URSim	Universal Robots Simulator
TCP	Tool Center Point
DH	Denavit-Hartenberg
ROS2	Robot Operating System 2
NASA-TLX	NASA Task Load Index
SUS	System Usability Scale
SSM	Speed and Separation Monitoring
PFL	Power and Force Limiting
SRMS	Speed and Risk Monitoring System
VIO	Visual-Inertial Odometry
SLAM	Simultaneous Localization and Mapping
IDMP	Interactive Distance-field Mapping and Planning
CPU	Central Processing Unit
GPU	Graphics Processing Unit
RAM	Random Access Memory
SDK	Software Development Kit
QoS	Quality of Service

Chapter 1

Introduction

This chapter introduces the motivation, problem statement, objectives, contributions, and structure of this dissertation, which investigates the role of Augmented Reality (AR) in improving safety, efficiency, and usability in human–robot collaboration (HRC).

1.1 Motivation and Context

1.1.1 The Rise of Human–Robot Collaboration (HRC) in Industry 4.0

Industry 4.0 has transformed manufacturing, logistics, and service sectors through the integration of cyber–physical systems, the Internet of Things (IoT), and advanced automation. One of its most significant shifts is the transition from traditional robotic automation, restricted to isolated, repetitive tasks, to collaborative robots (cobots) designed to operate safely alongside human workers [DUR⁺16]. This paradigm enables flexible production environments where robots can adapt dynamically to variations in product design, production volumes, and workflow requirements, eliminating the rigid barriers that once separated humans and machines. However, such proximity introduces critical safety challenges, requiring cobots to embed safeguards that prevent injury and build operator trust [CPD20].

1.1.2 Industry 5.0: A Human-Centric Evolution

Building upon Industry 4.0, Industry 5.0 shifts the focus towards human–centricity, resilience, and societal well-being. It seeks to integrate human creativity with robotic precision in shared workspaces [RSV⁺23]. Cobots are increasingly applied in tasks such as assembly, material handling, and logistics, reducing repetitive strain while enabling operators to focus on complex, higher-level decision-making. This evolution necessitates adaptive safety strategies capable of managing dynamic human–robot interactions (HRIs) without undermining productivity [CPD20].

1.1.3 Challenges in Human–Robot Interaction

While HRC offers operational benefits, several challenges remain unresolved:

- **Safety and Trust:** Operators may hesitate to engage with robots if their movements appear unpredictable or their intentions unclear.
- **Cognitive Load:** Conventional interfaces often fail to deliver intuitive, real-time feedback, reducing situational awareness.
- **Dynamic Workspaces:** Static safety measures such as fixed exclusion zones can halt operations unnecessarily, leading to productivity losses [SPM21].

These limitations highlight the need for interaction modalities that improve transparency, for instance, visualizing planned trajectories, and safety protocols that adapt dynamically to changing conditions.

1.1.4 Augmented Reality as an Enabler for HRC

Augmented Reality provides a promising avenue for bridging the gap between human intuition and robotic precision. By overlaying digital information directly onto the user’s field of view, AR allows human operators to perceive and interact with robotic systems in a more natural and intuitive way. Head-mounted displays (HMDs), such as the Microsoft HoloLens, enable immersive visualization of robotic trajectories, allowing operators to preview planned movements and receive holographic feedback in real time. This capability not only enhances the operator’s situational awareness but also fosters trust in the robot’s actions by making its intentions transparent [SPM21][SZH24].

In addition to visualization, AR facilitates hands-free guidance by providing contextual instructions, warnings, and operational cues without the need for physical interfaces. This reduces the cognitive load on the operator, enabling them to focus on collaborative tasks rather than on system navigation or control. Furthermore, AR contributes to proactive safety mechanisms: by continuously monitoring the operator’s position and movement, the system can generate real-time alerts or adapt robot trajectories to avoid hazardous interactions. Such features position AR as a key enabler in creating collaborative environments where humans and robots can safely and efficiently share the same workspace [MCP⁺24].

1.1.5 Motivation

In many industrial settings, collaborative robots still rely on static safety strategies such as pre-defined exclusion zones or spherical safety areas: if an operator enters this radius, robot motion halts regardless of the actual collision risk [MCP⁺24, SPM21]. While this guarantees operator protection, it also results in frequent and unnecessary interruptions, reducing workflow efficiency and overall system performance.

Recent studies highlight the importance of adaptive safety systems capable of modulating robot behavior according to real-time factors such as human proximity, trajectory prediction, and inferred operator intent [AWM⁺24, KKM⁺22]. Such approaches promise to increase both safety and productivity by minimizing unnecessary stoppages and maintaining operator trust. However,

existing systems rarely account for contextual information such as operator trajectory, intent, or task. As a result, robots cannot differentiate between benign proximity and genuine collision risks, and collaboration remains suboptimal.

Augmented Reality can help address these limitations by embedding safety information directly into the operator’s field of view through holographic overlays and contextual cues. In principle, AR provides situational awareness and transparency, but current AR-HMD interfaces face unresolved issues: ergonomic constraints, potential cognitive overload from excessive visualizations, and limited personalization of safety parameters [CPM22, BMAY⁺22, hos24].

To overcome these challenges, this dissertation proposes an *AR-enhanced dynamic safety framework*. The system projects holographic overlays of the robot’s workspace and planned trajectories, allowing operators to preview upcoming movements and better anticipate interactions. It also issues contextual alerts only when the operator’s predicted path is likely to intersect with the robot’s intended motion, ensuring that warnings remain relevant and non-intrusive. In doing so, the framework reduces unnecessary interruptions to the workflow while preserving strict safety standards, striking a balance between operational efficiency and operator protection.

The research problem can therefore be summarized as the inefficiency of static safety mechanisms in dynamic collaboration scenarios, together with the immaturity of current AR interfaces to deliver adaptive, low-strain, and user-centric safety visualizations. Bridging these gaps is essential to align collaborative robotics with the human-centric principles of Industry 5.0 [RSV⁺23].

1.2 Objectives and Research Questions

To address the challenges identified in the previous section, this dissertation investigates how AR-based interfaces can improve usability, safety, and efficiency in HRC. The specific objectives are to:

- Develop an adaptive AR-HMD safety framework that dynamically adjusts safety zones based on real-time human–robot proximity and predicts movement before execution, allowing the operator to preview actions in advance.
- Minimize cognitive and physical strain through context-aware holographic alerts tailored to operator state and task.
- Experimentally evaluate the framework against static safety zones, measuring its impact on safety, efficiency, and user experience.

These objectives are reflected in the following research questions:

1. **RQ1:** How can AR visualization improve transparency and trust in collaborative robotic systems?
2. **RQ2:** To what extent does an adaptive AR safety framework reduce unnecessary interruptions compared to static safety zones?

3. **RQ3:** What is the effect of AR-based interaction on usability, workload, and operator perception of safety?

1.3 Contributions

This dissertation makes the following contributions:

- A brand new AR-enhanced dynamic safety framework integrating trajectory prediction, real-time visual feedback, and adaptive safety management.
- An AR-based interface that allows immersive trajectory previews, holographic overlays, and contextual alerts aligned with operator movement.
- Experimental validation through two complementary studies: a system efficiency test comparing different control strategies, and a participant-based user study evaluating usability and workload.
- Insights into how AR can support human–robot trust, reduce workload, and increase efficiency in collaborative settings, with lessons for future industrial deployment.

1.4 Thesis Structure

In addition to this introduction, the dissertation is organized as follows. Chapter 2 presents the background and key concepts of HRC, AR, and safety standards. Chapter 3 reviews the state of the art in AR-based HRC and adaptive safety. Chapter 4 describes the overall system architecture, including hardware, software, and safety mechanisms. Chapter 5 details the implementation of the AR interface and robot integration. Chapter 6 reports experimental results from both system efficiency tests and the user study. Finally, Chapter 7 summarizes the main contributions and outlines directions for future work.

Chapter 2

Foundational Concepts

Understanding the fundamental principles and technologies that underpin human–robot collaborative workspaces enhanced with augmented reality is essential for contextualising the contributions of this dissertation. This chapter therefore presents an overview of the key concepts related to human–robot interaction, collaborative workspace design, AR head-mounted displays, safety frameworks, and enabling technologies such as tracking, digital twins, and communication systems. While Industry 5.0 has already been introduced in the context and motivation (Chapter 1), here we revisit it from a *technical perspective*, explaining how its principles translate into the requirements, standards, and technologies that shape the system proposed in this dissertation.

2.1 Industry 4.0 and Industry 5.0 in Collaborative Robotics

Industry 4.0 emphasised automation, connectivity, and data-driven optimisation. In contrast, Industry 5.0 introduces a human-centric perspective, prioritising resilience, sustainability, and human–machine collaboration [OS⁺23]. For collaborative robotics, this shift means robots are no longer designed to work in isolation or merely alongside humans, but rather to interact *with* humans in shared workspaces, adapting dynamically to operator behaviour.

From a technical standpoint, this evolution requires:

- **Transparency:** Robots must communicate intentions clearly through visual, auditory, or haptic cues.
- **Safety:** Adaptive methods must replace rigid fences and static safety zones.
- **Flexibility:** Systems must be reconfigurable, requiring AR interfaces and digital twins for intuitive setup.

2.2 Collaborative Robotics and Safety Standards

Collaborative robots are designed to operate in shared environments with humans, distinguishing them from traditional industrial robots that are confined behind fences [MMZ⁺19]. Their deployment is governed by standards such as ISO 10218 [iso11] and ISO/TS 15066 [iso16], which define four specific modes of safe operation:

- **Safety-rated monitored stop (SRMS)** In this mode, the robot ceases all motion whenever a human enters the collaborative workspace. Motion may only resume after the human leaves the area and a deliberate restart command is issued. This guarantees that no unintended motion occurs near the operator but often interrupts workflow fluidity.
- **Hand guiding** The robot is directly manipulated by the operator, typically via a dedicated device or a force-sensitive tool center point (TCP). While the robot executes no autonomous motion, the human physically “guides” its end-effector to program paths or perform collaborative tasks. This is safe and intuitive but limited in scalability for continuous collaboration.
- **Speed and Separation Monitoring (SSM)** The robot adapts its velocity and trajectory based on the real-time distance to the human. As the operator approaches, the robot progressively reduces speed, stopping entirely if the operator crosses a critical separation threshold. This dynamic behaviour enables productive coexistence while ensuring safety but requires accurate tracking of human position and reliable sensors.
- **Power and Force Limiting (PFL)** Here, the robot is designed to be inherently safe by limiting the forces, torques, and contact pressures it can exert. In case of accidental contact, the physical design (rounded edges, compliant joints) and control algorithms ensure that injury risk is minimised. While this enables close collaboration, payload and speed are constrained by these force limits.

These modes establish technical thresholds for motion, force, and proximity, but they do not prescribe *how* transparency or usability should be ensured. In practice, operators may remain uncertain about *why* the robot slowed down or stopped. Augmented Reality provides a complementary layer by visualising invisible safety envelopes, displaying robot intentions, and thereby reducing hesitation and improving trust in collaborative operations [SZH24].

2.3 Human–Robot Interaction

To accurately situate the form of human–robot interaction addressed in this dissertation, it is important to distinguish it from related concepts such as coexistence, cooperation, and collaboration. Ambiguity in these terms often leads to inconsistent communication, especially in industrial and safety-critical contexts [Vic20]. This section clarifies the differences between these concepts, outlines frameworks for classifying interaction intensity, and defines key terms that will recur throughout this dissertation.

Foundational Concepts

HRI is an umbrella term encompassing all exchanges between humans and robots [Bor14]. Within this broad field, interaction is commonly described across five levels [WMB⁺16, MMZ⁺19]:

- **Cell:** Human and robot are physically segregated, often by a safety fence, and do not cooperate directly [WMB⁺16].
- **Coexistence:** Both share the same broader environment but operate in separate workspaces, performing independent tasks [MMZ⁺19].
- **Synchronised:** Both share the same workspace but act at different times, avoiding temporal overlap [MMZ⁺19].
- **Cooperation:** Human and robot work in the same workspace at the same time on distinct yet complementary tasks [MMZ⁺19].
- **Collaboration:** Highest level of interaction, where both execute the same task simultaneously, requiring mutual adjustment and coordination [MMZ⁺19].

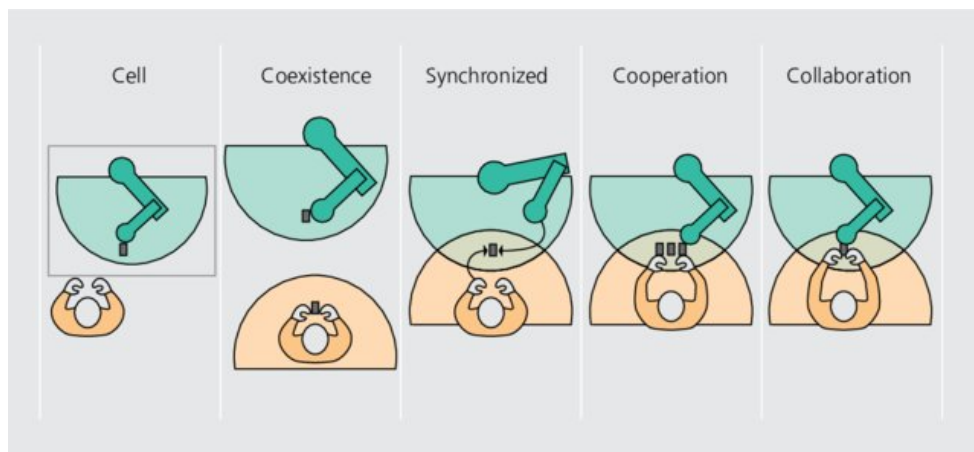


Figure 2.1: Canonical levels of human–robot cooperation/collaboration [WMB⁺16].

A *collaborative workspace* refers to the spatial region in which human and robot may be present and act, potentially simultaneously, as shown in Figure 2.2. This is the physical space where safety, efficiency, and communication become critical design factors.

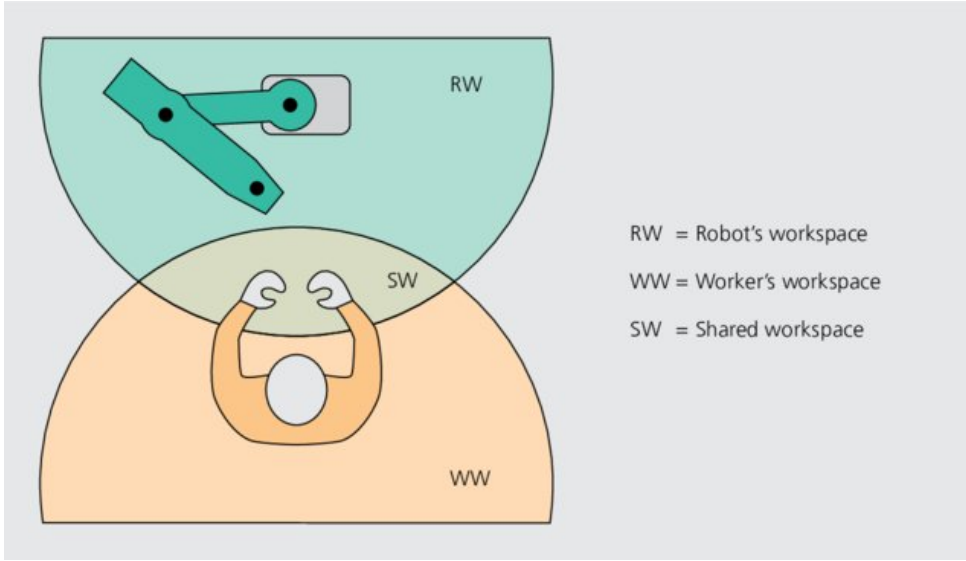


Figure 2.2: Workers’ and robots’ workspaces [WMB⁺16].

2.4 Mixed Reality and Augmented Reality

Augmented Reality and Mixed Reality (MR) are central to this dissertation, as they provide the medium through which transparency, safety, and usability are achieved. Azuma’s foundational definition describes an AR system as one that combines real and virtual content, operates interactively in real time, and is registered in three dimensions [Azu97]. This means that, for AR to be effective in human–robot collaboration, holograms must be spatially aligned with the robot’s real structure and must react without noticeable delay. If either requirement fails, overlays become misleading or distracting, undermining safety.

While Virtual Reality (VR) immerses users in fully synthetic environments, AR anchors information directly in the operator’s field of view of the real world. This characteristic is essential for safety in HRC, because it allows workers to remain situationally aware of the real robot while simultaneously accessing predictive cues. The Reality-Virtuality continuum proposed by Milgram et al. [MK94] positions AR close to the “real” end of the spectrum, making it uniquely suited for industrial environments where awareness of surroundings cannot be compromised.

Recent surveys [SZH24, CC⁺24] highlight AR’s role in fostering trust by making robot intentions explicit. Without visual support, operators often cannot anticipate whether a robot will stop, continue, or change trajectory. AR bridges this gap by overlaying trajectory previews, safety envelopes, and contextual warnings. Suzuki et al. [SYH22] further propose taxonomies for designing such AR interfaces, categorising them into guidance, visualization, and safety functions—three categories that align directly with the contributions of this work.

Recent work also demonstrates that head-mounted AR can scale to multi-user HRC settings, supporting collaborative coordination and evaluation of shared tasks [YAK⁺25]. This illustrates

the potential of AR not only for single-user safety and transparency but also for future multi-operator collaborative environments.

2.4.1 AR Interfaces and Display Modalities

AR content can be delivered through multiple modalities, each with distinct trade-offs in industrial contexts [CPM22]:

- **Head-Mounted Displays** such as Microsoft HoloLens 2 enable hands-free operation and direct spatial alignment of holograms with the robot. This is the modality adopted in this dissertation, as it best supports interactive trajectory previews and real-time safety cues. However, challenges remain in field of view, ergonomics, and tracking accuracy [SBSPM21, CPMM24].
- **Hand-Held Devices (HHDs)** such as tablets or smartphones provide lower entry barriers but occupy the operator’s hands, which reduces practicality in collaborative work.
- **Spatial Displays** (e.g., projectors) can highlight safety zones directly in the workspace, supporting shared awareness among multiple users. Yet, these are highly sensitive to occlusion and ambient lighting.
- **Fixed Screens** offer global overviews but force operators to shift their attention away from the robot, introducing cognitive and safety costs.

Recent research pushes AR interfaces beyond static visualization. Subramanian et al. [SZH25] demonstrated that combining AR with *eye-gaze communication* enables smoother collaboration, as robots can infer user attention and adapt trajectories. Similarly, the ARDIE framework [AFS23] integrates natural dialogue and gaze, showing how AR can evolve into a true communication layer rather than merely a display. These findings underline why this dissertation focuses on AR not just as an output device but as a medium for mutual understanding between human and robot.

2.5 Tracking Methods in AR for HRC

Accurate tracking is the foundation that allows AR content to align correctly with the physical robot. If the tracking system drifts or lags, virtual safety zones and trajectory previews no longer match the real robot, creating the risk of misleading the operator. In HRC, this problem is not only technical but also safety-critical.

2.5.1 Marker-Based Tracking

Marker-based methods (ArUco, AprilTag, ChArUco) provide absolute references with high accuracy and robustness at low computational cost. In this dissertation, fiducial markers are used to calibrate the coordinate alignment between the UR30 robot and the HoloLens 2, ensuring that the ghost-arm previews and safety overlays appear in the correct place [CPMM24].

2.5.2 Markerless Tracking

Markerless methods, such as Visual-Inertial Odometry (VIO) and SLAM, allow the HMD to estimate its pose without external markers. These methods provide flexibility, but their accuracy varies with texture, lighting, and motion [SBSPM21]. In industrial spaces with repetitive patterns and low visual features, markerless tracking alone may introduce unacceptable drift.

2.5.3 Hybrid Approaches

Hybrid approaches combine the stability of fiducial markers with the flexibility of markerless tracking. For example, markers can provide initial calibration and re-localisation, while SLAM maintains continuous tracking. This strategy is particularly relevant for the system presented in this dissertation, where QR-based calibration anchors the robot’s base in AR space, and markerless tracking allows fluid user interaction during operation.

2.5.4 Multimodal and Future Directions

Beyond visual tracking, recent work integrates gaze, speech, and gesture inputs to complement pose estimation [KLF23]. These multimodal strategies are promising for future collaborative systems, as they can cross-validate user position and intent, reducing the probability of misinterpretation. While not planned for this dissertation, they represent a natural extension of the presented AR framework.

In summary, accurate and reliable tracking is not an optional detail but a prerequisite for safe and transparent AR-based collaboration. Without it, the visual layer could erode rather than enhance trust. This dissertation addresses this by combining fiducial calibration with real-time markerless tracking, ensuring that AR overlays remain stable and informative during human–robot interaction.

2.6 Digital Twins and Simulation

The concept of a digital twin refers to a continuously updated virtual replica of a physical asset or system. Unlike a static simulation, a digital twin synchronises with the physical system through sensor data and bidirectional communication, creating a “living” model that can be used for prediction, monitoring, and optimisation [dt23].

In the context of Human–Robot Collaboration, digital twins play several critical roles:

1. **Transparency:** By mirroring the robot’s state and planned trajectories, the digital twin allows operators to anticipate actions before they occur in the physical workspace. This predictive capability is especially valuable when overlaid in AR, enabling users to see ghost-arms or future motion paths.

2. **Safety:** Simulated trajectories can be validated against virtual representations of the human operator or workspace constraints before execution. This reduces the risk of collisions and builds operator confidence that the robot will behave safely.
3. **Training and validation:** Digital twins provide a safe environment to test different strategies or configurations without putting the operator at risk. New safety parameters, trajectory planners, or AR cues can be trialled virtually before deployment.
4. **Integration with AR:** When coupled with HMDs, digital twins cease to be abstract models and instead become immersive, interactive interfaces. Operators can preview robot behaviour in situ, interact with holographic waypoints, and understand the effects of their actions in real time [[adl24](#), [caa24](#)].

The system presented in this dissertation embodies a lightweight form of digital twin: the UR30 robot state (joint positions, TCP pose) is mirrored in Unity and updated continuously. The AR interface overlays this twin in the user's field of view, enabling trajectory previews (ghost-arm) and real-time safety visualisation. Although simplified compared to industrial-scale digital twins, this implementation demonstrates how simulation and AR together can provide both predictive power and practical usability.

2.7 Human Factors in HRC

Technical safety mechanisms alone are insufficient if operators do not perceive the system as safe or usable. Human factors such as trust, usability, workload, and acceptance are decisive for whether collaborative robotics systems succeed in practice [[Vic20](#)]. In fact, a system that is technically safe but cognitively demanding may still be rejected by its users.

Key human factors relevant to this dissertation include:

- **Trust in automation:** Operators must believe that the robot will act predictably and respect safety boundaries. Lack of trust often leads to hesitation, overcautious behaviour, or even refusal to collaborate [[SZH24](#)]. AR helps establish trust by making invisible intentions, such as trajectories or safety zones-explicit and easy to interpret.
- **Usability:** Interfaces must be intuitive and reduce the need for training. Complex menus or unclear feedback increase the risk of errors. This dissertation evaluates usability through the System Usability Scale (SUS) [[Bro96](#)], providing a standardised benchmark against widely studied systems.
- **Workload:** Even if a system is functionally correct, excessive mental or physical demand can lead to fatigue or errors. The NASA-TLX workload assessment [[HS88](#)] is used in this work to quantify mental, physical, and temporal demands, effort, and frustration during AR-supported HRC tasks.

- **Perception of safety:** Beyond measured collisions or forces, what matters is whether users feel safe standing near a robot. AR contributes by reducing uncertainty and increasing situational awareness, which in turn fosters a subjective sense of safety.

The experiments presented later in this dissertation explicitly integrate these human factors. By combining SUS and NASA-TLX questionnaires with direct behavioural observation, the evaluation bridges technical performance with human experience. This is essential to demonstrate that the system is not only efficient but also accepted and trusted by its intended users-fulfilling the Industry 5.0 principle of human-centred design.

2.8 Networking and Real-Time Communication

Human-robot collaboration in shared workspaces imposes strict requirements on communication latency and reliability. Unlike traditional offline programming, where trajectories are planned in advance and executed without modification, dynamic safety demands continuous bidirectional data exchange between robot and interface. If communication is delayed or unstable, AR overlays may become desynchronised from the physical robot, creating confusion or even unsafe situations.

2.8.1 Robot Communication Interfaces

Universal Robots manipulators expose several TCP/IP ports [Uni19] for real-time control and monitoring. These include:

- **30001** – Primary interface for scripted control and program execution.
- **30003** – Real-time client interface, enabling high-frequency streaming commands (e.g., `speedl`).
- **30004** – Secondary real-time interface, often used for additional clients or monitoring.
- **30013** – State feedback channel, streaming joint positions, TCP pose, and system status.

These interfaces allow external systems-such as AR applications or ROS2 frameworks-to both read the robot's state and send commands in real time. In this dissertation, a lightweight C# middleware handles connections to these ports, providing both a stream of robot state data and continuous `speedl` commands to achieve smooth motion control.

2.8.2 Latency Requirements

For AR overlays to remain trustworthy, updates must occur at interactive frame rates. Literature highlights that delays above 50 ms become perceptible and can lead to misalignment between holograms and physical robots, undermining user trust [YLS25]. Low-latency communication is therefore not simply a performance optimisation but a prerequisite for safety and usability.

2.8.3 Middleware and Synchronisation

Frameworks such as ROS2 offer abstractions for message passing with configurable Quality of Service (QoS), making them suitable for multi-robot and multi-sensor environments. However, direct socket communication with the UR controller was chosen in this dissertation to minimise overhead and maintain deterministic update rates. The Unity implementation acts as a synchronisation hub:

1. Reading robot state packets from port 30013.
2. Aligning this state with the AR coordinate system (via QR-code calibration).
3. Streaming `speedl` commands through port 30003 to adapt robot motion continuously.

This architecture ensures that holographic previews (ghost-arm) and physical trajectories remain tightly coupled, a fundamental requirement for dynamic avoidance.

2.8.4 Relation to Dynamic Safety

Dynamic safety depends not only on perception of the human but also on communication responsiveness. When the operator’s hand enters the workspace, the system must:

- Detect the intrusion (via tracking).
- Transmit this information to the Unity middleware.
- Halt or adapt robot motion with minimal delay.

Any lag in this chain compromises safety. The networking design presented here was therefore central to enabling the dynamic avoidance strategy validated experimentally in Chapter 6.

In summary, real-time networking is the backbone of AR-supported HRC. By leveraging UR’s streaming interfaces and ensuring sub-50 ms communication latency, the implementation guarantees that AR overlays remain aligned with robot behaviour, thereby supporting both transparency and operator trust.

2.9 Conclusions

Taken together, these foundational concepts highlight the interdisciplinary nature of AR-supported human–robot collaboration. Industry 5.0 principles demand transparency, adaptability, and human-centred design; collaborative robotics introduces safety standards; AR provides an intuitive communication layer; tracking ensures spatial accuracy; digital twins extend predictive power; and human factors remind us that usability is as important as technical performance. By integrating these elements, the subsequent chapters build towards an AR–HRC system that is both technically robust and human-acceptable.

Foundational Concepts

Chapter 3

Literature Review

This chapter explores related works using Head-Mounted Displays in Human–Robot Collaboration, focusing on trajectory prediction and dynamic obstacle avoidance, in order to identify research gaps and assess the current state of the art of safety systems in HRCs.

3.1 Safety Approaches in Human–Robot Collaboration

3.1.1 Static Safety Mechanisms

The work by Soares et al. marked one of the earliest attempts within INESC TEC to implement AR-based programming and static safety zones for collaborative robots [SPM21]. Their system allowed operators without programming expertise to guide a robot through AR demonstration, effectively lowering the entry barrier for robot programming.

In parallel, a safety system was integrated to monitor operator position and trigger warnings whenever a pre-defined workspace boundary was crossed. Figure 3.1 illustrates the spherical exclusion zones designed for a UR5 mobile manipulator and an ABB IRB 2600. These zones provided a straightforward means of ensuring operator safety, yet they were entirely static and heavily dependent on robot geometry.

Such rigidity meant that operator motion or task variation could not be accommodated dynamically, resulting in conservative safety envelopes that reduced efficiency. Moreover, while the implementation raised awareness of the need for integrated safety in AR-based programming, it did not explicitly align with ISO standards nor provide adaptive behaviour. This underlines the limitations of static exclusion zones and motivates the adoption of more flexible methods that account for operator trajectory and intent.

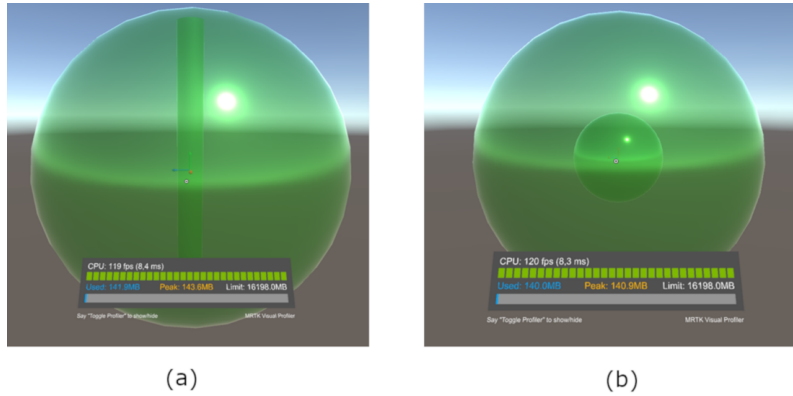


Figure 3.1: Projected workspace: (A) Universal Robots UR5. (B) ABB IRB 2600 [SPM21].

3.1.2 Dynamic Safety Mechanisms

To overcome the rigidity of static zones, Karagiannis et al. introduced a method of dynamic speed and separation monitoring [KKM⁺22]. Their approach divided the collaborative space into adaptive zones, each corresponding to different robot behaviours: full speed when the human is distant, reduced speed in intermediate zones, and a complete stop in critical proximity. Figures 3.2 and 3.3 show how these adaptive boundaries evolve in real time.

The advantage of this method lies in its ability to balance safety and productivity: the robot can remain active as long as the operator maintains sufficient distance, unlike static stop/resume behaviour. However, transitions between zones were relatively abrupt, sometimes producing unnatural motion that could be perceived as unsafe by operators. Furthermore, while spatial adaptation was achieved, no predictive visualisation was provided to the user, leaving operators uncertain about why the robot slowed down or when it might stop.

Ali et al. proposed an alternative approach called Interactive Distance-field Mapping and Planning (IDMP) [AWM⁺24]. Instead of using discrete zones, their framework generated continuous distance and gradient fields from depth camera input, creating a fluid representation of free and occupied space (Figure 3.4). The key benefit was real-time responsiveness without high computational overhead, allowing the robot to replan its motion around dynamic obstacles with minimal delay.

Although IDMP demonstrated efficient avoidance, it did not address the communication of robot intent to the human operator. From the perspective of transparency and trust, the robot moved “safely” but without conveying its reasoning. Both Karagiannis and Ali advanced adaptive safety but highlight the missing link: integration with AR-based visualisation to explain robot behaviour in human-understandable terms.

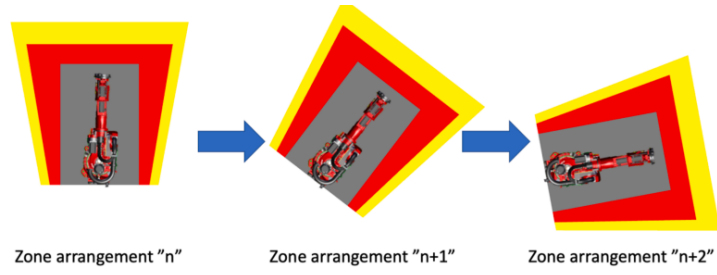


Figure 3.2: Examples of zone arrangements [KKM⁺22].

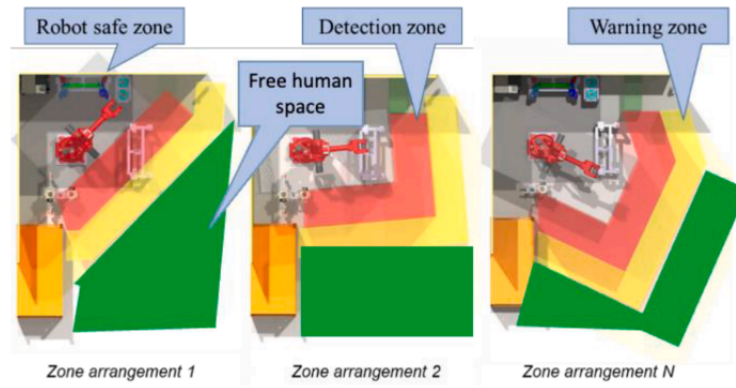


Figure 3.3: Dynamic safety zone arrangements and operator free space [KKM⁺22].

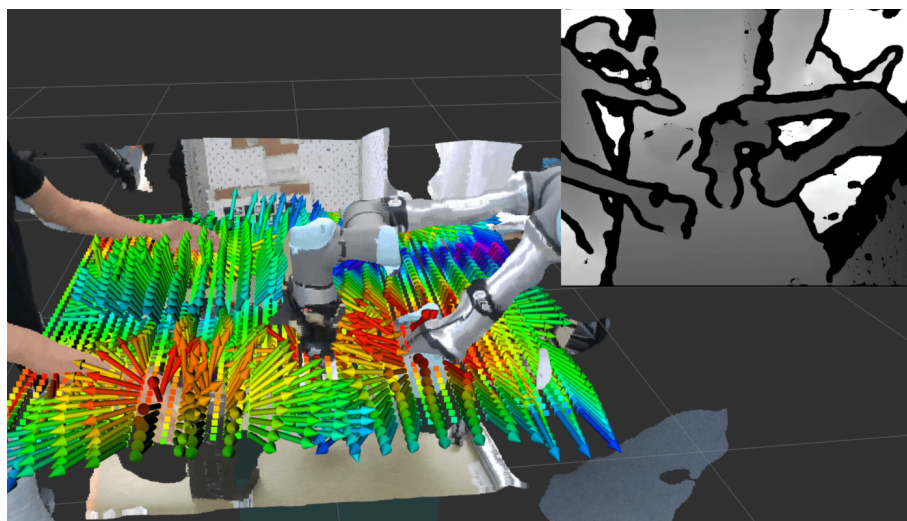


Figure 3.4: Interactive distance and gradient fields in an HRC setting [AWM⁺24].

3.2 Augmented Reality in HRC: Guidance, Visualization, and Safety

3.2.1 AR for Programming & Guidance

Martins et al. presented an AR framework for mobile manipulators designed with Industry 5.0 principles in mind [Mar25]. Their system introduced two notable “soft-safety” mechanisms. First, a red semi-transparent holographic sphere enveloped the robotic arm, triggering a slowdown to 10% speed when a user entered. Second, a multi-level safety strategy for the robot’s mobile base adjusted velocity from 100% down to 0% depending on human proximity.

By visualising these safety envelopes directly in AR, the framework improved operator awareness and reduced unnecessary protective stops. This was a meaningful step forward compared to purely static methods, because it allowed operators to understand when and why speed reductions occurred. Nevertheless, the system did not feature full trajectory adaptation: the robot slowed or stopped, but its path remained unchanged. This highlights AR’s value as a guidance and awareness tool, while also pointing to the need for coupling it with adaptive path planning and prediction, precisely the focus of this dissertation.

3.2.2 AR for Trajectory Visualization

Rybalskii et al. proposed the use of AR for trajectory previews, overlaying semi-transparent red holograms that showed the future tip path of a manipulator [RKSS24]. Inspired by earlier work by Hosseini et al. [HFB⁺23], their system sought to make robot motion more predictable to operators and aligned with European directives such as the Machinery Directive 2006/42/EC [dir06]. Figure 3.5 illustrates this preview concept.

The strength of this approach is early conflict identification: operators can immediately see whether the planned path intersects with their workspace, enabling them to react before a collision becomes imminent. However, two critical limitations remained. First, only the end-effector trajectory was displayed, while the intermediate robot links, which also pose collision risks, were ignored. Second, the system provided no proactive avoidance: the robot’s path was visualised but not adapted dynamically. This dissertation addresses both gaps by introducing full-arm previews (ghost-arm) and coupling them with real-time adaptive avoidance.

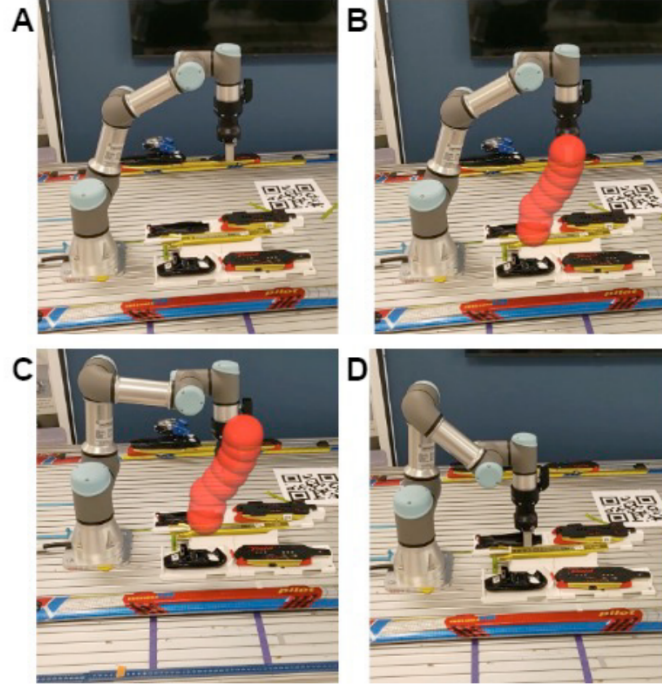


Figure 3.5: Visualization of future cobot tip trajectory [RKSS24].

3.3 Path Planning, Prediction & Simulation for HRC

3.3.1 Unity-Based Path Planning Platforms

Angelidis et al. developed an extended reality platform in Unity3D to experimentally evaluate path planning approaches for HRC [APVM24]. Their study compared Rapidly-Exploring Random Trees (RRT) with a Machine-Learned Path Planning (MLPP) strategy across 21 participants. Results indicated that RRT provided safer trajectories with fewer collisions, while MLPP offered faster completion times but sometimes at the expense of predictability.

A distinctive feature of their setup was the integration of visual and auditory cues to alert participants when restricted spaces were violated (Figure 3.6). This not only increased user awareness but also allowed evaluation of how interface feedback influenced perceived safety. Importantly, the platform was used as a simulation tool rather than a deployed AR interface. While participants interacted in XR, the robot remained virtual, and no immersive HMD was employed to overlay trajectories in real physical space.

Thus, Angelidis et al. demonstrate the potential of interactive simulation for testing planning strategies and safety cues, but their work stops short of translating these insights into AR-enhanced physical collaboration. This leaves an opportunity, seized in this dissertation, to combine predictive planning, real robot execution, and AR overlays in a single integrated framework.

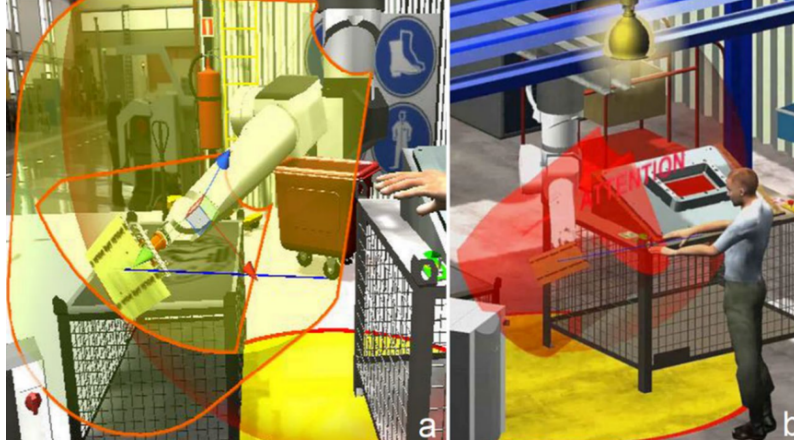


Figure 3.6: Virtual robot workspace with proactive safety cues [APVM24].

3.4 Conclusions of the Literature Review

To the best of the author’s knowledge, no system yet provides a fully integrated method for predicting and visualizing the *entire* manipulator trajectory in AR while also performing dynamic avoidance in real time, without the help of external sensors.

Soares et al. [SPM21] and Martins et al. [Mar25] advanced AR-based programming and safety visualization but remained limited to static or semi-dynamic zones. Rybalskii et al. [RKSS24] improved transparency with tip previews but ignored link-level collisions. Angelidis et al. [APVM24] demonstrated the role of simulation in testing planners, while Ali et al. [AWM⁺24] and Karagianis et al. [KKM⁺22] advanced adaptive avoidance without AR transparency.

This dissertation addresses this gap by combining full-arm predictive visualization (ghost-arm) with real-time dynamic avoidance and validating usability and safety through participant studies.

Table 3.1: Comparative summary of related works in HRC safety and AR visualization

Authors	Focus	Main Contribution	Limitations / Gaps
Soares et al. (2021) [SPM21]	AR-based programming with static safety zones	Enabled non-experts to program robots via AR demonstration; defined spherical exclusion zones for UR5 and ABB arms.	Zones were static and geometry-dependent; reduced efficiency; no dynamic adaptation; not aligned with ISO standards.
Karagiannis et al. (2022) [KKM ⁺ 22]	Dynamic speed and separation monitoring	Adaptive zones with speed reduction and stop behaviour depending on human proximity.	Abrupt zone transitions; limited flexibility in complex scenarios; no predictive visualisation of intent.
Ali et al. (2024) [AWM ⁺ 24]	Distance-field mapping and planning	Proposed IDMP with real-time gradient fields from depth cameras; enabled smooth dynamic obstacle avoidance.	Focused only on avoidance; no integration of AR visualisation; operator could not anticipate robot decisions.
Martins et al. (2025) [Mar25]	AR guidance for mobile manipulators (Industry 5.0)	Introduced AR soft-safety cues: translucent arm sphere and multi-level base slowdown; improved awareness and reduced stops.	Did not include full trajectory adaptation; still zone-based; reliance on slowing rather than avoidance.
Rybalskii et al. (2024) [RKSS24]	AR-based trajectory preview	Displayed semi-transparent holograms of end-effector path; improved operator awareness and early conflict identification.	Only tip trajectory shown; ignored link-level collisions; lacked proactive avoidance.
Angelidis et al. (2024) [APVM24]	XR-based path planning simulation	Compared RRT and MLPP with 21 participants; added visual/audio safety cues in Unity3D simulation.	Simulation-focused; no immersive AR with physical robots; lacked full predictive visualisation in HMD.
This work	AR predictive visualization + dynamic avoidance	Combines full-arm ghost trajectory previews with real-time speed1-based avoidance; validated through user studies.	Currently limited to one robotic platform (UR30) and lab-scale validation.

Literature Review

Chapter 4

System Design and Architecture

This chapter presents the design and architectural decisions behind the development of the system aimed at improving safety and efficiency in human-robot collaborative workspaces through the integration of AR. The proposed system leverages a digital twin of a UR manipulator within Unity to simulate, visualize, and evaluate robotic motion before it is physically executed. By previewing the robot’s movements with a “ghost arm”, the system enables the user to safely and intuitively set target positions and understand the implications of those movements in real time.

The architecture is designed to meet three core objectives:

- **Predictive Visualization:** Accurately simulate and display the robot’s movement path using a virtual ghost arm before execution.
- **User Interface Integration:** Develop an AR-based interface that enables users to interactively assign end-effector targets and initiate actions.
- **Dynamic User Avoidance:** Implement a real-time collision prediction and avoidance mechanism that accounts for the user’s position (head and hands) to minimize interruptions or halts in robotic operation.

The remainder of this chapter details each component of the architecture, including communication protocols, system flow, collision handling logic, and the decision-making logic that enables adaptive robotic behavior in a shared human-robot environment.

4.1 Motivation and Scenario

In collaborative workspaces, human operators and robotic manipulators share the same physical environment. While robots such as the Universal Robots UR30 are capable of executing precise and repeatable tasks, their integration alongside humans introduces critical safety and usability challenges. Operators must not only be protected from unintended collisions, but also be provided

with intuitive mechanisms to guide, preview, and validate robotic movements before they are executed.

Augmented Reality (AR) offers a promising solution to this problem. Through an AR head-mounted display such as the Microsoft HoloLens 2, operators can visualize the robot’s digital twin in real scale, interactively define trajectories, and receive real-time feedback about the robot’s state. This capability bridges the gap between human intention and robotic execution, making the process of programming and supervising robots more transparent and accessible to non-experts.

The system proposed in this work builds on these opportunities: it enables an operator to create waypoints in AR, preview the corresponding motion through a “ghost arm” representation, and finally execute the validated trajectory on the real robot. Safety is ensured through mechanisms that automatically pause execution when the operator’s hands or head approach the robot, and through dynamic detours generated in real time when the end-effector path intersects with the human workspace. In this way, the system combines efficiency, intuitiveness, and collaborative safety in a unified framework.

The proposed system operates in a collaborative workspace where both the human operator and the robot share the same environment. The main entities are:

Operator: The operator wears the Microsoft HoloLens 2, which provides an augmented reality interface for authoring, previewing, and executing trajectories. The operator interacts with the system through hand gestures, head movements, and visual feedback. Safety is also directly linked to the operator, as hand and head proximity are monitored in real time.

AR Application: Developed in Unity with MRTK2 and ARFoundation, the AR application is responsible for rendering the ghost arm, handling user input, generating waypoints, and monitoring safety conditions. It also manages the user interface, which includes the *Robot Menu*, the *Waypoint Editor*, and the tutorial. This application serves as the central coordinator between the operator and the robot.

Robotic Arm: The UR30 robot (or URSim) receives commands from the AR application through TCP/IP connections. It provides continuous state data (joint angles and Cartesian pose) and executes trajectories sent either as real-time velocity commands (`speed1`) or as batched linear motion programs (`move1`).

Environment: The physical workspace represents the shared environment in which the operator and the robot coexist. It contains obstacles that may require avoidance and serves as the reference space in which AR content is visualized. Spatial alignment between the AR application and the robot is achieved through QR code calibration, ensuring consistency between the virtual and physical representations.

Figure 4.1 illustrates the relationships between these entities. The operator interacts with the AR application via the HoloLens 2, the AR application communicates with the robot through URScript over TCP/IP, and both the operator and the robot share the same physical environment, which is synchronized with the AR coordinate frame via QR calibration.

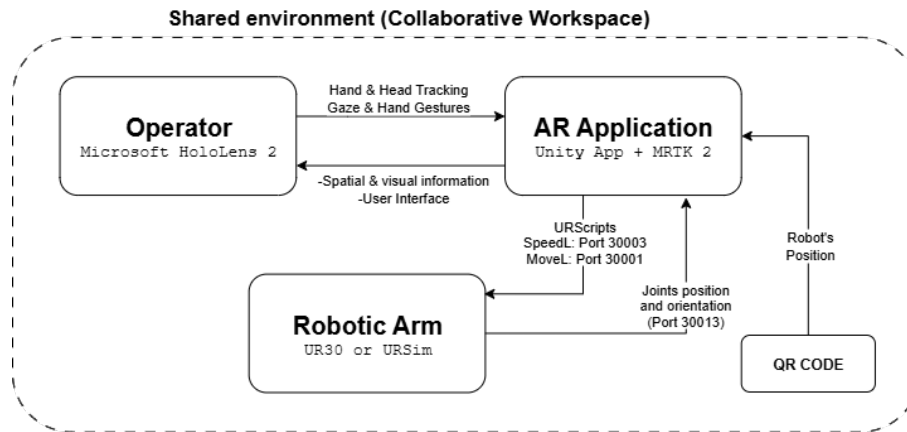


Figure 4.1: System context showing the interaction between operator, AR application, robotic arm, and QR-based calibration.

4.2 Design Requirements

From the scenario described above, a set of requirements was defined to guide the system design. These requirements ensure that the solution is not only functional, but also usable, safe, and compatible with the constraints of both the AR hardware and the UR robotic platform. They are organized into three categories: functional, non-functional, and system constraints.

4.2.1 Functional Requirements

The system must provide the following core capabilities:

- **Waypoint authoring:** allow the operator to create and edit waypoints by manipulating the tip of the robotic arm, as well as clear them directly within the AR environment using the “ghost arm”.
- **Trajectory preview:** enable visual inspection of the planned motion through a virtual execution of the ghost arm before sending commands to the robot.
- **Robot execution:** execute trajectories on both the URSim simulator and the physical UR30 robot, using either continuous streaming commands or batched programs.
- **Safety monitoring:** automatically pause and resume execution when the operator’s hands or head enter restricted safety zones, and generate real-time detours when the end-effector path intersects with human obstacles.
- **Diagnostics:** provide access to robot state information such as joint angles and Cartesian pose for monitoring purposes.
- **Tutorial and guidance:** support new users with a multimodal tutorial combining audio instructions and a visual ghost-hand demonstration.

4.2.2 Non-Functional Requirements

Beyond core functionality, the system must also fulfill the following non-functional requirements:

- **Low latency:** communication between AR device and robot must allow responsive updates suitable for real-time collaboration.
- **Spatial accuracy:** the alignment between the AR environment and the robot must be accurate enough to ensure that waypoints correspond to physically reachable positions.
- **Continuous motion:** the primary execution mode must avoid unnecessary stops between waypoints, ensuring fluid movements of the robot.
- **Usability and workload:** the user interface must support efficient interaction with minimal cognitive effort, measurable using standardized tools such as the System Usability Scale and NASA-TLX.

4.2.3 System Constraints

The design of the system is bounded by the following technical constraints:

- **Communication with the robot:** the system interfaces with a Universal Robots collaborative manipulator through TCP/IP socket connections, transmitting commands written in the native URScript language. Three ports are used: port 30013 for continuous state streaming (e.g., joint angles and TCP pose), port 30003 for real-time velocity commands using `speedl`, and port 30001 for sending small URScript programs (e.g., batched `move1` instructions).
- **Hardware compatibility:** the AR interface targets optical see-through head-mounted displays; a reference implementation was built for Microsoft HoloLens 2 with Unity and MRTK2.
- **Superposition of holograms:** alignment between the AR coordinate frame and the robot base frame is achieved through the detection of a QR code marker. This method was selected because it is simple to deploy, natively supported by AR Foundation, and provides sufficient accuracy for aligning waypoints in the experimental setup.

4.3 User Story and Workflow

To illustrate the intended operation of the system, this section presents the workflow from the perspective of an operator wearing the HoloLens 2. This “user story” highlights how the different features of the system are experienced in sequence, providing an intuitive bridge between the design requirements and the architectural choices that follow. The overall flow is summarized in Figure 4.2.

1. **Calibration via QR Code:** The workflow begins with spatial calibration. The operator scans a QR code marker placed at the robot base using the HoloLens 2 camera. This process aligns the AR coordinate system with the physical robot coordinate frame, ensuring that waypoints defined in AR correspond to physically accurate positions in the workspace.
2. **Connecting to the Robot:** After calibration, the operator opens the *Robot Menu*, where the system provides tools to establish a connection with either the physical UR30 robot or the URSim simulator. Diagnostics such as joint angles and Cartesian pose are displayed, and test movements using `speedl` commands can be executed to verify communication.
3. **Authoring Waypoints:** Once connected, the operator switches to the *Waypoint Editor*. Here, a “ghost arm” is displayed in AR. By using hand gestures, the operator grabs and places the tip of the ghost arm at the desired location, creating a new waypoint. After confirming the position, small spheres are generated, forming a visible trajectory in the shared workspace. Waypoints can also be cleared at any time to restart the authoring process.
4. **Status Feedback:** Throughout the workflow, a status bar provides immediate feedback on the system state, indicating whether it is ready to execute a trajectory or whether no valid waypoints are currently defined.
5. **Previewing the Trajectory:** Before execution, the operator can preview the planned motion. In this mode, the ghost arm animates through the sequence of waypoints at a safe speed, allowing the operator to visually confirm that the trajectory is correct and does not interfere with the environment. The preview is purely virtual and does not move the physical robot.
6. **Executing the Trajectory:** When satisfied with the preview, the operator initiates execution on the real robot. Two execution strategies are available: (a) continuous streaming with `speedl`, or (b) batched programs with small `movel` sequences. The first ensures continuous motion, while the second was used for experimental comparison.
7. **Safety Interaction:** During execution, the system continuously monitors the operator’s hands and head. If they approach the robot’s links, execution is paused automatically. If a hand approaches the trajectory of the end-effector, the system dynamically adjusts the path, generating a local arc detour in real time. Once the operator withdraws from the restricted area, execution resumes seamlessly.
8. **Tutorial Guidance:** For experimental studies, a built-in tutorial is available in the *Robot Menu*. It combines audio instructions with a visual ghost-hand indicator to guide participants step by step through the system’s functionality.

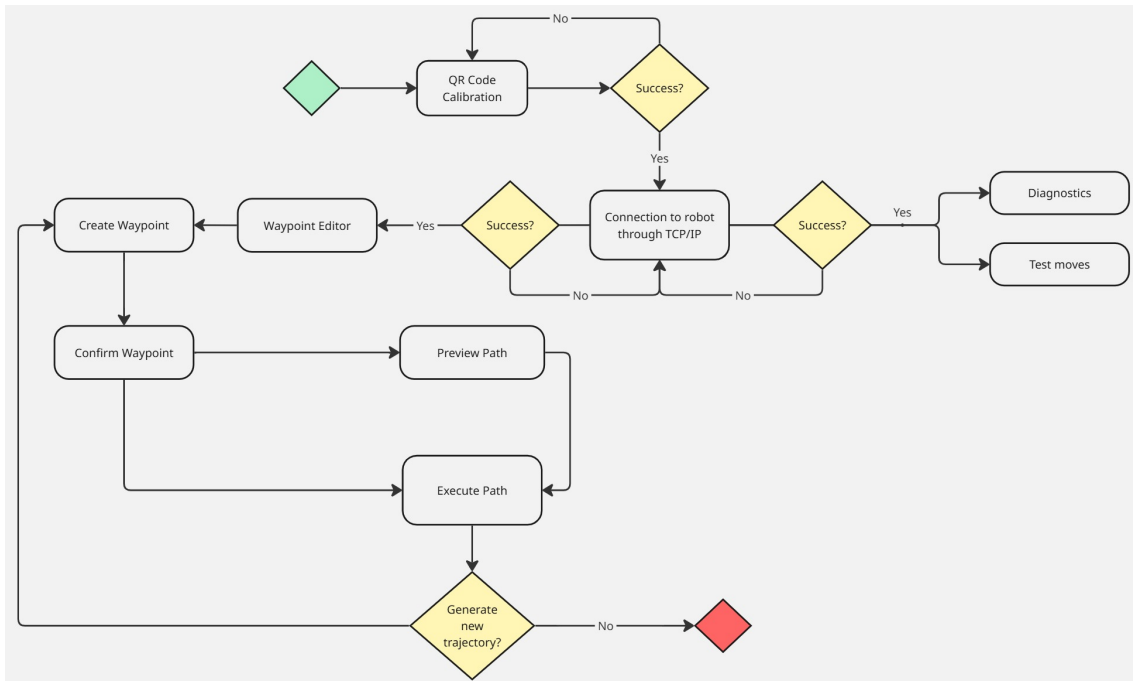


Figure 4.2: Workflow of the AR-based collaborative system from calibration to trajectory execution. The flowchart illustrates the operator's interaction sequence, including calibration, connection, waypoint authoring, preview and execution.

4.4 System Architecture

The system architecture defines how the different components interact to achieve the workflow outlined above. It is organized into four main modules, each responsible for a critical aspect of the system: robot communication, waypoint management, safety monitoring, and spatial calibration.

Robot Communication Layer: Communication with the robot is handled by a dedicated module adapted from the *Unity3D_Robotics_UR* project by Parák [Par]. This layer is responsible for establishing TCP/IP connections to the UR30 (or URSim) and managing the exchange of commands and state data. Three ports are used: port 30013 for state streaming (robot pose and joints), port 30003 for real-time velocity control using `speedl`, and port 30001 for sending small URScript programs such as batched `move1` instructions.

Waypoint Management: The *Waypoint Controller/Spawner* implements the waypoint authoring and execution logic. It generates waypoints based on the ghost arm's end-effector position, interpolates trajectories, and provides preview functionality. During execution, trajectories can be sent as continuous `speedl` streams or batched `move1` programs, depending on the selected mode.

Safety Monitoring: The *Safety Stop* module ensures operator protection by monitoring proximity between the user's hands or head and the robot. Capsule colliders are attached to the robot

links to detect approaches, pausing execution if necessary. In parallel, dynamic avoidance is implemented for the end-effector trajectory: when the user's hand intersects the path, the system generates a real-time arc detour that lifts the robot motion over the obstacle.

Spatial Calibration: Spatial alignment between the AR and physical environments is achieved through QR code detection. The *Scan and Position* module establishes the offset between the AR coordinate system and the robot base frame, ensuring that virtual waypoints correspond to physically reachable positions.

4.5 Summary

The architectural analysis highlights how the proposed system bridges the gap between intuitive human input and precise robotic execution. By embedding the robot's digital twin in AR, operators are no longer required to program abstract coordinates but can instead author trajectories in a spatially aligned, visual manner. The integration of real-time communication with the UR robot ensures that these trajectories are not only simulated but also faithfully executed.

A central outcome of the architecture is that safety is treated as an embedded property of the interaction, rather than an external constraint. The combination of proximity-based monitoring and dynamic end-effector detours enables collaboration without forcing the robot into unnecessary stops, preserving both efficiency and operator trust. Furthermore, the modular separation of calibration, waypoint management, communication, and safety ensures that each component can evolve independently, supporting scalability and adaptability to different collaborative scenarios.

In short, the system architecture demonstrates that it is possible to design a collaborative workspace where usability and safety are not competing objectives but mutually reinforcing. This foundation sets the stage for the following chapter, where the implementation of these architectural principles is described in detail.

Chapter 5

Implementation

5.1 Introduction

This chapter describes the implementation of the system architecture outlined in Chapter 4. While the previous chapter focused on the conceptual design, requirements, and structural decomposition of the system, the present chapter details how these ideas were realized in practice using Unity, MRTK2, the Microsoft HoloLens 2, and the Universal Robots platform. The chosen toolchain favors stability and reproducibility in an industrial setting, even if it limits the exploration of more experimental interaction toolkits.

The implementation followed a modular approach, mirroring the architecture, and was carried out iteratively. Each component was developed and tested individually before being integrated into the complete workflow. Special attention was given to achieving real-time performance, maintaining accurate spatial alignment between AR and the physical robot, and embedding safety mechanisms directly into the control logic.

The remainder of this chapter is structured as follows. Section 5.2 introduces the development environment, including hardware and software platforms. Section 5.3 presents the robot communication layer, detailing the TCP/IP interface with URScript. Section 5.4 describes the adaptation of the ghost arm model from an existing UR10 asset to the UR30 manipulator used in this work. Section 5.5 explains the waypoint management system. Section 5.6 illustrates the AR user interface. Section 5.7 describes the implementation of the safety systems. Finally, Section 5.8 covers the superposition of holograms via QR Markers procedure.

5.2 Development Environment

The implementation required a combination of hardware and software platforms to support augmented reality interaction, real-time robot control, and seamless integration between the two. The environment was selected to balance maturity and development speed against the risk of vendor lock-in, with an emphasis on reproducibility for user studies.

5.2.1 Hardware

- **Microsoft HoloLens 2:** an untethered mixed-reality headset providing hand tracking, gaze interaction, and spatial mapping, used as the operator’s primary interface. Its hands-free interaction is well suited to HRC, although the limited field of view constrains how much of the holographic trajectory is visible at once, which motivated concise visual cues.
- **Universal Robots UR30:** a collaborative manipulator with a payload of 30 kg and a reach of 1300 mm, chosen for its availability, suitability in industrial human–robot collaboration tasks. For testing purposes, the URSim simulator was also employed, providing a virtual instance of the robot controller that enabled repeatable experiments while acknowledging that network and controller latencies differ slightly from the physical system.
- **Development workstation:** a desktop computer equipped with an Intel i7 CPU, 32 GB RAM, and an NVIDIA RTX 4070 GPU, used for Unity development and communication with both the HoloLens 2 and the robot.

5.2.2 Software

- **Unity 2022.3.2f1:** the main development environment for building the AR application, chosen for its compatibility with HoloLens and mature AR toolchain. Long-term support reduced integration risk compared to adopting newer XR pipelines.
- **Mixed Reality Toolkit 2:** used for hand tracking, gesture recognition, and interaction components. MRTK2 was preferred for stability and documentation. Even though MRTK3 already existed during the development of this dissertation, it still lacked effective documentation and community discussion compared to MRTK2.
- **Visual Studio 2022:** used for scripting and deployment to the Microsoft HoloLens 2.
- **ARFoundation and OpenXR:** libraries enabling marker detection, spatial alignment, and deployment on HoloLens 2.
- **URSim:** the official simulator from Universal Robots, used to test and validate communication without requiring the physical robot.
- **Blender:** used to adapt the UR10 model to UR30 geometry, ensuring kinematic accuracy and correct sizing of the ghost arm visualization.
- **Programming languages:** C# for Unity scripting and URScript for robot motion control.

5.2.3 Networking Setup

Communication between the AR application and the robot was established through INESC TEC’s network using TCP/IP. Three ports were employed: port 30013 for robot state streaming (pose

and joint values), port 30003 for real-time velocity commands using `speedl`, and port 30001 for sending small URScript programs containing batched `move1` instructions. This configuration allowed both continuous streaming for smooth execution and discrete program-based control for experimental comparisons.

Port 30004, the secondary client interface, was deliberately not used, since it requires UR-Caps and additional setup on the robot controller. By relying solely on the primary and real-time interfaces, the implementation remained lightweight and could be fully developed within Unity without external extensions, which simplified deployment and reproducibility.

5.3 Robot Communication Layer

The communication between the AR application and the UR30/URSim controller is implemented over TCP/IP using URScript as the control interface. The Unity-side integration builds upon the open-source project by Parák, *Unity3D_Robotics_UR* [Par], which provides a robust starting point for socket management and basic message exchange. In the original implementation, robot state was mainly streamed for visualization and simple test movements were performed by sending `speedl` commands on port 30003. Building on this codebase accelerated delivery while retaining full visibility into network behavior, a property that would be harder to obtain through a heavier middleware.

In this work, the base project was substantially adapted to support additional communication modes and provide a more complete interface for collaborative operation:

- **State streaming on port 30013** was added to continuously monitor the robot’s joint values and Cartesian pose, enabling timely UI feedback and safety gating.
- **Program upload on port 30001** was integrated to allow the transmission of small URScript programs containing batched `move1` instructions, used for experimental comparisons where determinism of exact point passage was required.
- The existing **realtime client on port 30003** was repurposed to not only send test moves but also execute full user-defined trajectories via continuous `speedl` streaming, favoring continuity and responsiveness when a human is in the loop.

This adaptation allowed the system to combine smooth, realtime velocity-based control with the ability to fall back on program-based execution when needed, while simultaneously exposing a robust and low-latency state stream for diagnostics and safety monitoring. Moreover, the intention of this work was to implement the entire solution directly in Unity without relying on external middleware, which made a lightweight socket-based approach preferable for the study objectives.

5.3.1 Script Overview: `ur_data_processing`

The Unity component `ur_data_processing` orchestrates all robot input/output (I/O). It follows a simple state machine in `FixedUpdate`: upon *Connect* it launches two background threads,

one for state reading (`UR_Stream`) and one for command sending (`UR_Control`); upon *Disconnect*, it stops both. Separating read and write paths prevents head-of-line blocking and simplifies error handling during transient network faults.

State streaming on 30013 (reader thread): The `UR_Stream` thread establishes a TCP client with `NoDelay` and short I/O timeouts, then continuously reads packets from port 30013. Each packet begins with a 4-byte big-endian length header; the body is read exactly and parsed. The implementation follows UR’s binary layout by indexing known doubles from the tail of the packet (after a reverse step for convenience), extracting joint positions `J_Orientation[0..5]` (radians), TCP position `C_Position[0..2]` (meters), and TCP orientation `C_Orientation[0..2]` (axis-angle radians). After each successful parse, `pose_seq` is incremented atomically, allowing consumers to obtain consistent snapshots. Any socket error triggers a clean reconnect with a brief backoff. This technique is efficient but depends on firmware layout stability; offsets were versioned in code to guard against future changes.

Realtime control on 30003 (writer thread): The `UR_Control` thread manages the real-time client on port 30003. It reconnects on failure and, when a send is requested, writes the prepared command buffer to the stream and flushes it. In this project, high-level `speedl` streaming with lookahead, filtering, arrival detection, and safety-hold gating is performed in the `WaypointController`, while `ur_data_processing` ensures robust, threaded socket-level transmission. This division makes testing simpler and confines potential mapping errors to a single module.

Primary programs on 30001 (moveL batches): To enable comparison with velocity-based streaming, the script supports program uploads on the Primary interface. When `UR_Control_Data.execute_moveL` is set, the script builds and sends a small `URScript` program containing one or more `moveL` instructions. In the experimental setup, each program contained two `moveL` commands, allowing sequential execution with controlled pauses. This mode maximizes determinism of exact waypoint passage at the expense of continuity.

Connection management and diagnostics. A port status utility attempts a non-blocking connect to ports 30013/30003/30002/30001 and logs OPEN/CLOSED. State data are exposed to the Unity UI, showing joint values and TCP pose in the *Diagnostics* panel for transparency during experiments.

5.4 Holographic robotic arm

A key element of the implementation is the visualization of the robot’s future trajectory through a virtual “ghost arm.” This model enables the operator to preview motion before execution, improving both safety and intuitiveness. To ensure accuracy, the ghost arm was adapted to match the kinematics of the UR30 manipulator used in this work. The adaptation favored mesh simplicity for on-device performance while preserving link frames and pivot alignment, which was more impactful for trust in previews than photorealistic detail.

Implementation

Source Model: The ghost arm was initially based on an open-source UR10 model from the *Flange* project [Pre25]. However, since the UR30 differs in both link lengths and axis conventions, the model required adaptation.

UR30 Parameters: The adaptation was guided by the official Denavit–Hartenberg parameters of the UR30, shown in Table 5.1, which define the kinematic chain of the manipulator. These parameters served as the ground truth for configuring the virtual arm. A pre-flip in Blender aligned axis conventions so that the imported model matched the base frame in Unity without additional runtime transforms, reducing the risk of subtle sign errors.

Table 5.1: UR30 Denavit–Hartenberg parameters and dynamic properties [Rob23]

Kinematics	a [m]	d [m]	α [rad]	Dynamics	Mass [kg]	Center of Mass [m]	Inertia Matrix
Joint 1	0	0.2363	$\pi/2$	Link 0	16.343	$\begin{bmatrix} -0.0001 \\ -0.0600 \\ 0.0069 \end{bmatrix}$	$\begin{bmatrix} 0.0883 & -0.0001 & -0.0001 \\ -0.0001 & 0.0764 & 0.0076 \\ -0.0001 & 0.0076 & 0.0830 \end{bmatrix}$
Joint 2	-0.6370	0	0	Link 1	28.542	$\begin{bmatrix} 0.3894 \\ 0 \\ 0.2103 \end{bmatrix}$	$\begin{bmatrix} 0.1379 & 0.0001 & -0.0451 \\ 0.0001 & 2.5013 & -0.0000 \\ -0.0451 & -0.0000 & 2.4751 \end{bmatrix}$
Joint 3	-0.5037	0	0	Link 2	7.156	$\begin{bmatrix} 0.2257 \\ 0 \\ 0.0629 \end{bmatrix}$	$\begin{bmatrix} 0.0236 & 0.0000 & -0.0168 \\ 0.0009 & 0.3388 & 0.0001 \\ -0.0168 & 0.0001 & 0.3353 \end{bmatrix}$
Joint 4	0	0.2010	$\pi/2$	Link 3	3.054	$\begin{bmatrix} 0 \\ -0.0048 \\ 0.0353 \end{bmatrix}$	$\begin{bmatrix} 0.0056 & 0.0000 & 0.0006 \\ 0.0000 & 0.0051 & 0.0006 \\ 0.0000 & 0.0006 & 0.0043 \end{bmatrix}$
Joint 5	0	0.1593	$-\pi/2$	Link 4	3.126	$\begin{bmatrix} 0 \\ 0.0046 \\ 0.0341 \end{bmatrix}$	$\begin{bmatrix} 0.0060 & 0.0000 & 0.0000 \\ 0.0000 & 0.0056 & -0.0006 \\ 0.0000 & -0.0006 & 0.0046 \end{bmatrix}$
Joint 6	0	0.1543	0	Link 5	0.926	$\begin{bmatrix} 0 \\ 0 \\ -0.0293 \end{bmatrix}$	$\begin{bmatrix} 0.0009 & 0.0000 & 0.0000 \\ 0.0000 & 0.0009 & 0.0000 \\ 0.0000 & 0.0000 & 0.0012 \end{bmatrix}$

Implementation

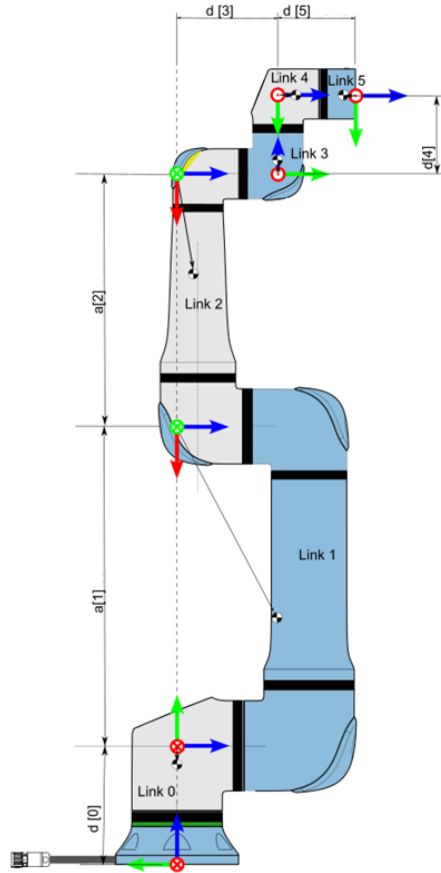


Figure 5.1: UR30 link structure and DH parameters [Rob23]

Model Conversion in Blender: Blender was used to change the UR10 mesh into a UR30 model downloaded through the Universal Robots website. Each link was positioned according to the DH parameters. During this process, special care was taken to align joint pivots with the UR30’s kinematic reference frames, just like in figure 5.1. This ensured that rotations applied in Unity would reproduce the correct physical motions of the real manipulator.

Axis Alignment: A further challenge arose from differing axis conventions: the UR robot controller, Blender, and Unity each use different axis orientations. To resolve this, the DH parameters were first translated into Blender’s coordinate system, and the model was pre-flipped before export. This way, once imported into Unity, the ghost arm was already aligned with the UR base frame without requiring additional corrections.

Final Model: Figure 5.2 shows the final adapted UR30 model in Blender, prepared for export and integration into Unity. This model was then imported into the Unity application, where it replaced the existing UR10. By ensuring that the ghost arm followed the exact kinematics of the UR30, the system provided reliable trajectory previews that could be trusted for both safety evaluation and operator guidance.

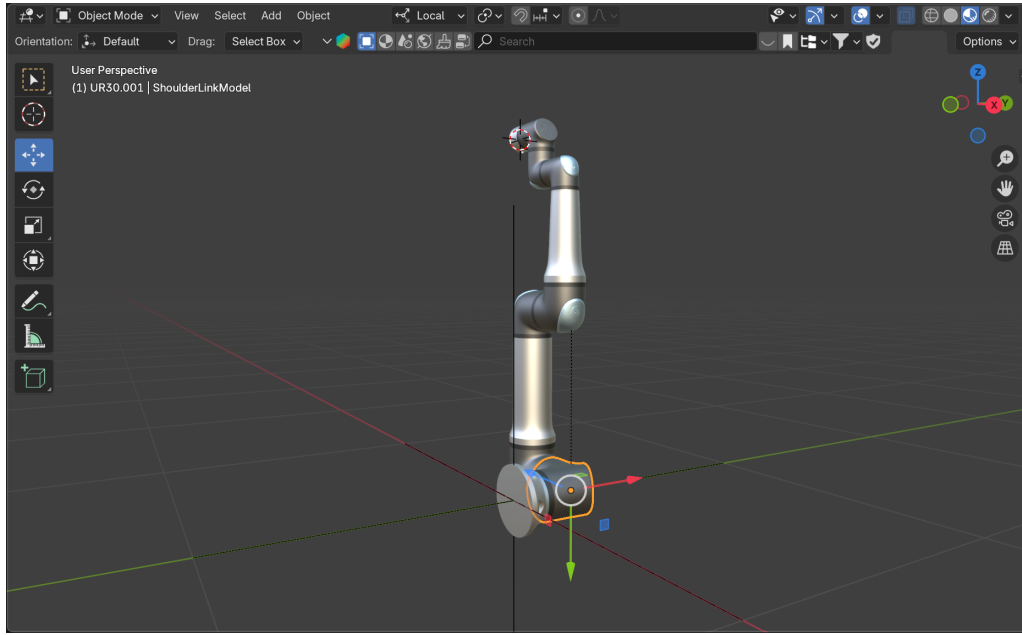


Figure 5.2: UR30 ghost arm adapted in Blender and pre-flipped for correct Unity import.

5.5 Waypoint Generation Algorithm

The waypoint generation algorithm was implemented to ensure that the robot could execute user-defined trajectories while maintaining safe distances from both static and dynamic obstacles in its workspace. When the operator places waypoints in the AR interface, the system cannot simply connect these points with straight lines, since such trajectories might intersect with the robot's base or the operator's body. To address this, the function `generateIntermediateWaypoints` refines the trajectory by inserting additional waypoints whenever a segment between two consecutive points would collide with an obstacle. Two complementary strategies were developed: one for the robot base and another for human obstacles, 5.3.

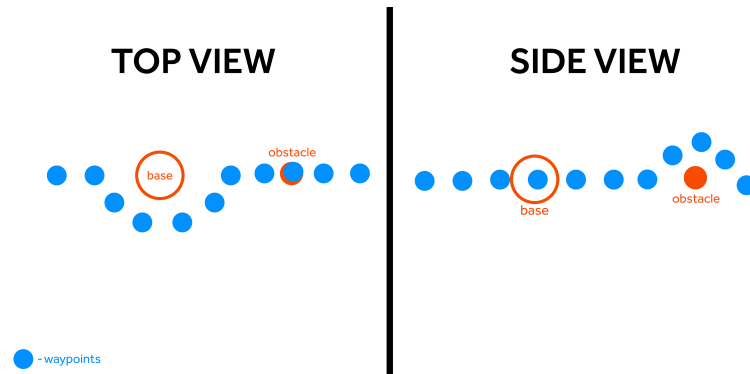


Figure 5.3: Exemplification of the waypoint generation algorithm

5.5.1 Base Obstacle (static arc detour)

The robot's base is modelled as a circular exclusion zone on the XZ plane, centred at the base origin (x_b, z_b) . Its radius r corresponds to the physical base radius extended by a safety clearance. A trajectory segment between two waypoints $P_1(x_1, y_1, z_1)$ and $P_2(x_2, y_2, z_2)$ is checked for intersection with this circle by solving:

$$L(t) = p_1 + t(p_2 - p_1), \quad t \in [0, 1] \quad (5.1)$$

where $p_1 = (x_1, z_1)$ and $p_2 = (x_2, z_2)$. The condition for intersection is:

$$\|L(t) - (x_b, z_b)\|^2 = r^2 \quad (5.2)$$

If this quadratic has a solution in $[0, 1]$, the line would cut through the base. To resolve this, the segment is replaced by an **arc** around the circle. The polar angles of p_1 and p_2 relative to the base are calculated as:

$$\theta_1 = \text{atan2}(z_1 - z_b, x_1 - x_b), \quad \theta_2 = \text{atan2}(z_2 - z_b, x_2 - x_b) \quad (5.3)$$

and the shorter angular direction is chosen. A sequence of intermediate points is then generated along the arc:

$$(x_k, z_k) = (x_b + r \cos \theta_k, z_b + r \sin \theta_k), \quad \theta_k \in [\theta_1, \theta_2] \quad (5.4)$$

while the y coordinate is interpolated linearly between y_1 and y_2 .

5.5.2 Human Obstacles (dynamic vertical lift)

Human joints (e.g. hand, head) detected via MRTK2 are modelled as spheres of radius r_h around the tracked position $H(x_h, y_h, z_h)$. For a segment P_1P_2 , the parametric line is:

$$L(t) = P_1 + t(P_2 - P_1), \quad t \in [0, 1] \quad (5.5)$$

and a collision occurs if:

$$\|L(t) - H\|^2 = r_h^2 \quad (5.6)$$

has a valid solution for $t \in [0, 1]$. If this condition holds, the system generates a vertical detour rather than a horizontal arc. The midpoint of the segment is computed as:

$$M = \frac{P_1 + P_2}{2} \quad (5.7)$$

and lifted above the obstacle:

$$M' = (M_x, y_h + r_h + \Delta y, M_z) \quad (5.8)$$

where Δy is an additional clearance margin. The original segment is then replaced by two sub-segments ($P_1 \rightarrow M'$) and ($M' \rightarrow P_2$). Unlike the base detour, these intermediate points are recomputed in real time at each update step, ensuring that the trajectory adapts continuously to the user's movements.

5.5.3 Rationale for the two approaches

The distinction between these methods was necessary because of the different nature of the obstacles. The robot base is a large, immovable structure at the centre of the workspace; without arc-based avoidance, many user-defined straight-line trajectories would unrealistically pass through it. The arc ensures global feasibility of the trajectory and can be computed once with no performance overhead. In contrast, human obstacles are unpredictable and dynamic. A permanent detour around them would make the robot's motion inefficient and unresponsive. Instead, a lightweight vertical lift, recalculated frame by frame, ensures safety while allowing the trajectory to remain as direct and fluid as possible.

In this way, the waypoint generation system combines static precomputed arcs for structural clearance with real-time adaptive lifts for human safety, producing trajectories that are both geometrically valid and responsive to collaborative interaction. This is illustrated in figure 5.4.

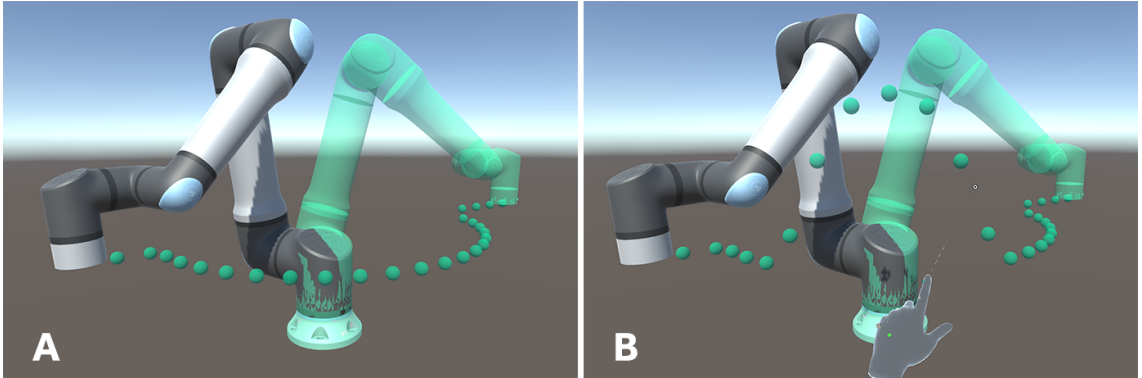


Figure 5.4: Dynamic avoidance example: A) Path with arc around the base; B) Path with arc around the base and avoidance on the MRTK2 hand.

5.6 User Interface

The user interface was designed to make the interaction with the system clear, efficient, and intuitive. By leveraging the capabilities of the Microsoft HoloLens 2 and the MRTK2, the operator is able to control the entire workflow without external devices, relying solely on hand gestures, gaze, and spatially anchored holograms. The interface is divided into two main modes: the *Robot Menu* and the *Waypoint Editor*, complemented by a status bar and a multimodal tutorial, both illustrated in figures 5.5 and 5.6.

The two menu modes are mutually exclusive and represent different stages of interaction.

Implementation

The **Robot Menu** provides tools for connecting to the robot, monitoring its state, and testing simple commands. This is the entry point for the operator once the system is calibrated.

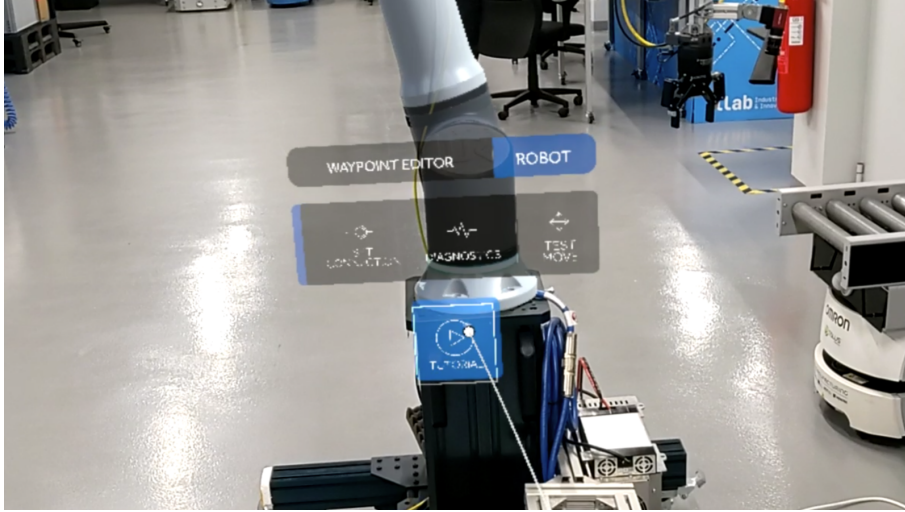


Figure 5.5: Robot menu during experimental test with Microsoft HoloLens 2

The **Waypoint Editor** enables the operator to author trajectories in AR space, preview them using the ghost arm, and execute them on the real robot.

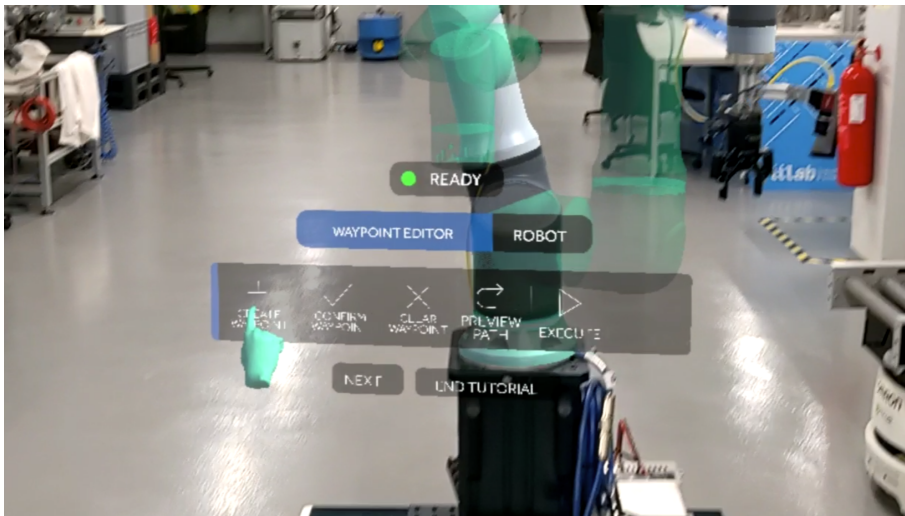


Figure 5.6: Waypoint editor during experimental test with Microsoft HoloLens 2

The operator can switch between these modes at any time, maintaining flexibility while keeping the interface uncluttered.

5.6.1 Robot Menu

The *Robot Menu* acts as the central hub for connection and diagnostics. It exposes the following features:

Implementation

The **Set Connection** option allows the operator to open the connection window and specify the IP address of the robot or simulator and establish a TCP/IP connection. Once the connection is successfully established, the interface provides clear visual confirmation by turning the status box green, ensuring that the user immediately understands that the robot is online and ready for interaction, as shown in figure 5.7 (B). Conversely, if the operator presses the disconnect button, the status box turns red, as illustrated in figure 5.7 (A), making it equally intuitive that the connection has been terminated. This color-coded feedback reduces ambiguity and provides a quick at-a-glance understanding of the system's connectivity state.

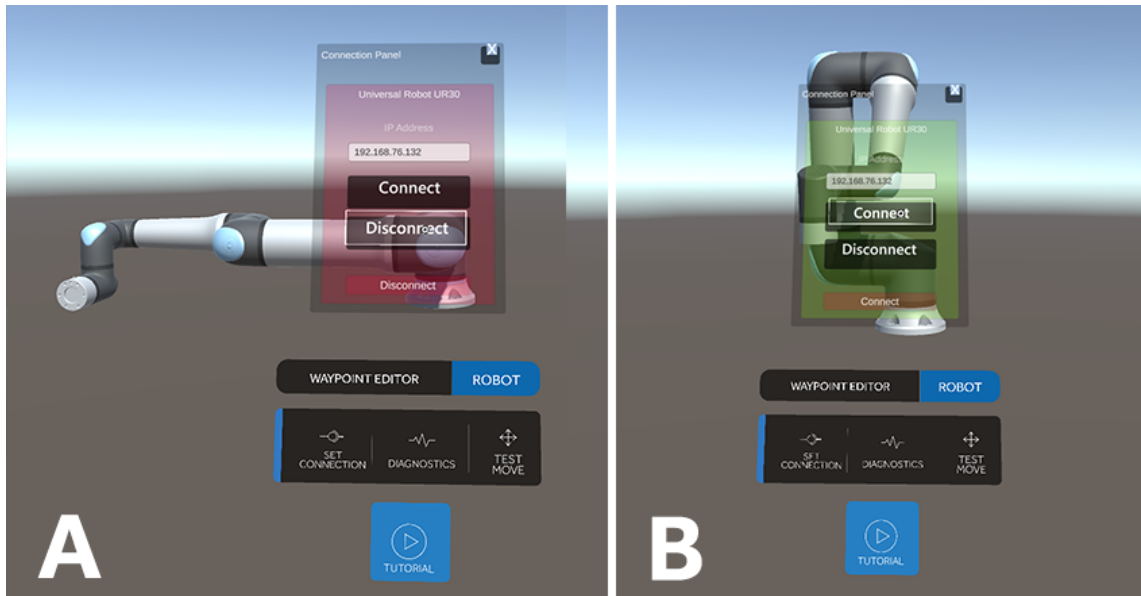


Figure 5.7: Connection panel of the AR interface. (A) Disconnected state, where the status box is shown in red to indicate loss of connection. (B) Connected state, where the status box turns green to confirm a successful TCP/IP connection with the UR30.

After connection, the **Diagnostics** view displays real-time information streamed from port 30013, including the joint angles of the six UR30 axes and the Cartesian pose of the TCP, figure 5.8. This transparency allows the operator to validate the health of the system and ensures confidence that the data exchange is functioning correctly. By monitoring these values, users can quickly detect abnormal conditions or confirm that the robot is accurately reporting its state.

Implementation



Figure 5.8: Diagnostics panel of the AR interface, showing real-time feedback from the UR30. The panel displays the Cartesian position and orientation of the TCP (X, Y, Z, RX, RY, RZ), as well as the six joint angles (J1–J6). This transparency allows the operator to validate the robot’s current state and verify that data streaming from port 30013 is functioning correctly.

The **Test Move** option enables the operator to issue simple `speedl` commands that move the robot by a small offset, figure 5.9. This functionality is particularly useful for verifying connectivity and responsiveness before committing to more complex trajectories, serving as a low-risk way to confirm that commands are executed properly.

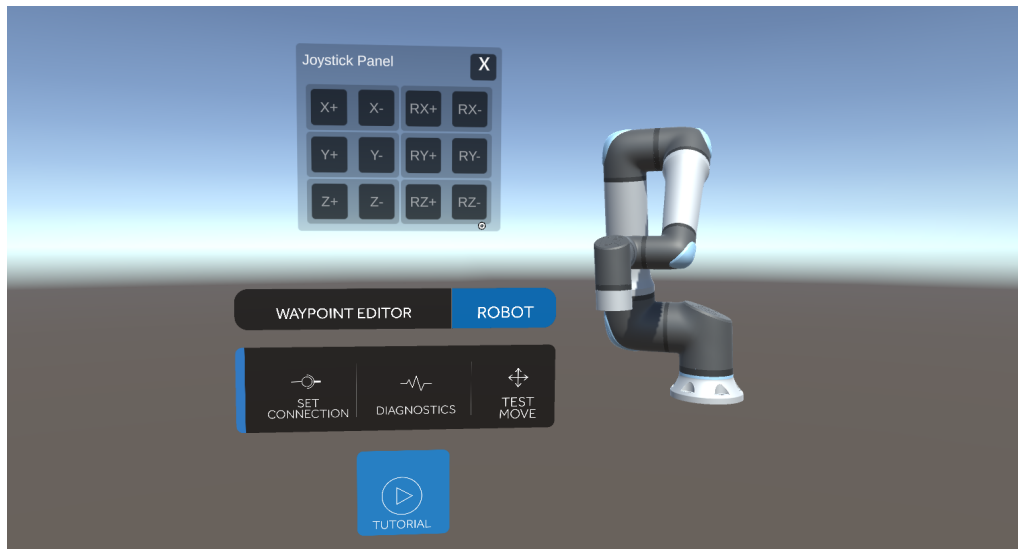


Figure 5.9: Test Move panel of the AR interface, implemented as a joystick-style control. The operator can issue incremental `speedl` commands along the Cartesian axes (X, Y, Z) and rotations (RX, RY, RZ). This feature provides a simple way to verify connectivity and responsiveness of the UR30 before executing complex trajectories.

Finally, the **Tutorial** function provides an accessible entry point for new users. A dedicated button launches the built-in tutorial, which combines audio instructions with a ghost-hand visual

guide. This feature was developed specifically for participant studies, making it possible for operators with no prior robotics knowledge to understand how to interact with the system effectively and safely.

5.6.2 Waypoint Editor

The *Waypoint Editor* provides the tools for creating, validating, and executing trajectories directly within the AR environment. It allows operators to design robot motions intuitively through hand interaction and holographic feedback, as illustrated in Figure 5.10. This interface makes it possible to define waypoints naturally in space, preview motions with the ghost arm, and execute them on the physical UR30 or the simulator, always with the option to stop execution immediately for safety.

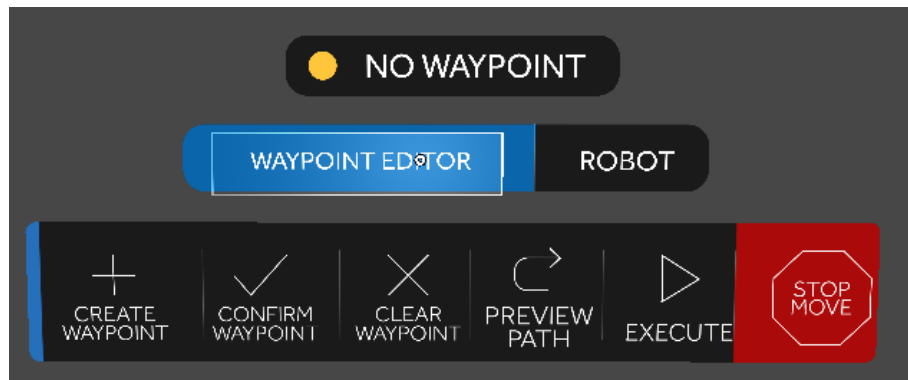


Figure 5.10: Waypoint Editor with Status Bar in Unity. The interface enables operators to create, preview, and execute trajectories using the ghost arm, with safety ensured through an immediate stop command.

The **Create Waypoint** function enables the operator to use hand gestures to grab and reposition the ghost arm’s tip in space. This provides an intuitive mechanism for defining target positions, as waypoints can be authored naturally in relation to the physical workspace.

Once positioned, the **Confirm Waypoint** option allows the operator to lock in the chosen location. The trajectory is then generated through the algorithm described in Section 5.5, which instantiates small spheres to form a path. This visual representation makes the robot’s intended motion explicit before execution.

At any moment, the **Clear Waypoints** command can be used to remove all existing spheres from the AR scene, allowing the operator to restart the authoring process with a clean workspace. This avoids visual clutter and ensures flexibility when redefining a path.

Before committing to execution, the **Preview Path** option animates the ghost arm along the defined sequence of waypoints. The motion is shown at a safe speed, enabling the operator to confirm the feasibility of the trajectory without moving the real robot.

Once the path has been validated, the **Execute** command initiates the trajectory on either the physical UR30 or the URSim simulator. Depending on the chosen mode, execution can be

performed through continuous `speedl` streaming, which provides smooth and responsive motion, or through batched `move1` programs, which were used for experimental comparison.

Finally, the **Stop Move** function ensures that the operator can immediately halt execution in case of unexpected conditions. This safety mechanism interrupts any ongoing trajectory, guaranteeing that the robot can be brought to a stop without delay.

5.6.3 Status Bar

To support situational awareness, a status bar is always visible in AR, as shown in Figure 5.11. It communicates the current readiness of the system through a simple but effective color-coded message. When the bar displays **Ready**, it indicates that the system is connected, calibrated, and has a valid trajectory defined. In contrast, when the bar shows **No Waypoint**, the system is either idle or lacks a valid trajectory available for execution. This minimal but persistent feedback helps the operator quickly determine whether the system is ready for action, reducing uncertainty and reinforcing trust in the interface.



Figure 5.11: Status bar of the AR interface. It provides persistent feedback on system readiness, showing either a "Ready" state or a "No Waypoint" state depending on the current configuration.

5.6.4 Tutorial Guidance

The tutorial provides step-by-step instructions for new users, figure 5.12. It combines audio playback with a ghost-hand visual that demonstrates the required gesture (e.g., grabbing, placing, confirming). To further guide the participant, the “Next” button is outlined visually once each audio instruction finishes, figure 5.12 (3), reminding the user to proceed. This multimodal tutorial lowers the entry barrier for non-expert participants in experimental evaluations and ensures consistency in task instructions.

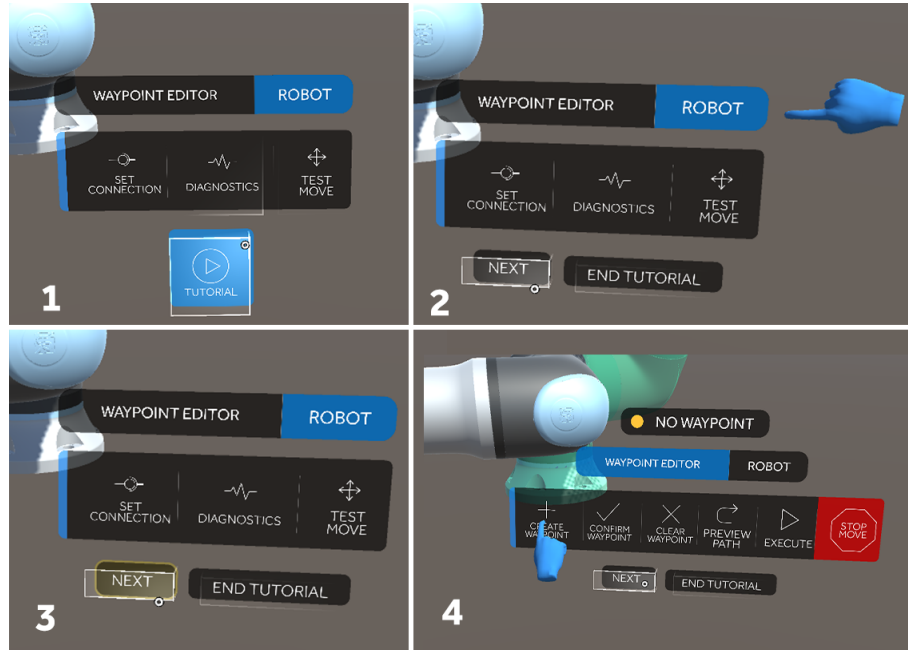


Figure 5.12: Tutorial example in sequential order

5.7 Safety System

Ensuring the safety of the human operator is a fundamental requirement of any collaborative robotic system. The approach adopted in this work combines two complementary mechanisms: local pausing of the robot when the operator’s body comes too close to its links, and dynamic avoidance of the operator’s hands at the end-effector level. Together, these mechanisms enable continuous operation with minimal disruption, while guaranteeing operator protection. An important design decision was to prioritize lightweight, Unity-only solutions instead of relying on external middleware (e.g., ROS 2 or URCaps) as mentioned before. While such alternatives offer modularity, they also introduce additional synchronization overhead and deployment complexity. By keeping the safety layer fully embedded in Unity, the system remained easier to maintain, reduced sources of nondeterminism, and aligned with the project’s emphasis on a self-contained AR interface.

5.7.1 Proximity Detection via Colliders

To ensure safe collaboration, the system continuously measures the distance between the operator’s tracked joints (hands and optionally head) and capsule colliders attached to the robot’s links, as illustrated in Figure 5.13. Each joint position is obtained in world coordinates through MRTK2, while each collider provides a geometric envelope of the corresponding link or tool. This choice of capsule geometry was deliberate: although more complex meshes could provide higher fidelity, capsules provide a robust trade-off between accuracy and computational cost, ensuring real-time performance.

Implementation

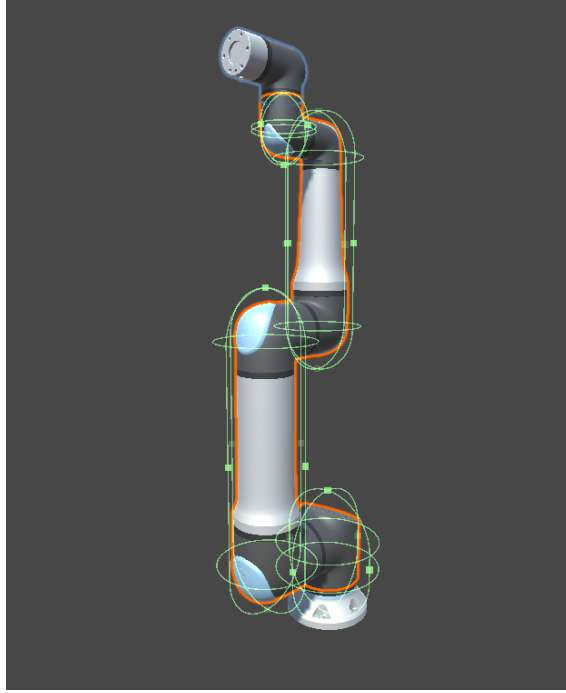


Figure 5.13: Capsule colliders placed on the robot's links for proximity monitoring.

Let the set of tracked joints be $\mathcal{J} = \{j_1, \dots, j_m\}$, with positions

$$\mathbf{h}_j \in R^3, \quad j \in \mathcal{J}. \quad (5.9)$$

Each capsule collider can be described as a line segment with endpoints $\mathbf{a}, \mathbf{b} \in R^3$ and a radius R . For a joint position $\mathbf{h}$, the projection onto the segment is

$$t = \text{clamp}\left(\frac{(\mathbf{h} - \mathbf{a}) \cdot (\mathbf{b} - \mathbf{a})}{\|\mathbf{b} - \mathbf{a}\|^2}, 0, 1\right), \quad (5.10)$$

$$\mathbf{q} = \mathbf{a} + t(\mathbf{b} - \mathbf{a}), \quad (5.11)$$

where $\mathbf{q}$ is the closest point on the capsule axis. The axis distance is then

$$d_{\text{axis}} = \|\mathbf{h} - \mathbf{q}\|, \quad (5.12)$$

and the distance to the capsule surface is

$$d_{\text{caps}} = \max(0, d_{\text{axis}} - R). \quad (5.13)$$

In practice, Unity's `Collider.ClosestPoint()` method is used to obtain the nearest surface point $\mathbf{p}$, from which the distance is simply

$$d = \|\mathbf{h} - \mathbf{p}\|. \quad (5.14)$$

Implementation

If $\mathbf{h}$ lies inside the collider, the function returns $\mathbf{p} = \mathbf{h}$, yielding $d = 0$.

To decide whether to pause execution, the minimum distance is evaluated across all joints and all colliders:

$$d_{\min} = \min_{j \in \mathcal{J}} \min_{c \in \mathcal{C}} d(\mathbf{h}_j, c). \quad (5.15)$$

A safety hold is triggered whenever $d_{\min}$ falls below a critical threshold d_{trigger} . Motion is resumed once the distance exceeds a larger release threshold d_{release} , combined with a short temporal delay τ to avoid oscillations. This hysteresis mechanism was chosen deliberately: although it adds implementation complexity, it prevents rapid stop–resume flickering and increases operator trust in the system’s responsiveness.

The corresponding Unity implementation uses a loop over joints and colliders. For each joint pose, the system queries the closest point on the collider and measures the Euclidean distance:

```
Vector3 p = col.ClosestPoint(h);  
float d = Vector3.Distance(h, p);
```

The minimum value is compared against the thresholds, and if necessary, the `WaypointController` is instructed to pause or resume execution. This decision to integrate pausing at the controller level, rather than through an external safety stop, reflects a trade-off between simplicity and robustness. It avoids forcing the robot into a protective stop (which would require manual recovery) and instead achieves a “soft stop” that can be reversed automatically. However, this choice also implies a dependency on Unity’s correct functioning, which would need to be mitigated by an additional hardware e-stop in a production environment.

This mechanism acts as a virtual protective stop: instead of aborting execution or resetting the program, the trajectory is suspended until the operator moves away from the restricted zone. As shown in Figure 5.14, when the operator’s hand enters the vicinity of a collider, the robot halts immediately and resumes once conditions are safe. By combining geometric distance computation with hysteresis, the system achieves robust and responsive proximity detection in real time, while acknowledging the limitation that it does not address untracked obstacles.



Figure 5.14: Protective stop triggered during system testing when a user places the hand close to a collider.

5.7.2 Dynamic End-Effector Detours

While link-based proximity detection prevents direct collisions, it does not address cases where the operator’s hand intersects the planned path of the end-effector. To handle these situations, the system implements a dynamic avoidance strategy: whenever the hand enters the end-effector’s trajectory, the system generates a local detour arc explained in Section 5.5 that redirects the motion above the hand in real time. This design decision was made to favor continuity of task execution, as simply halting the robot whenever the hand approached the TCP would have undermined the collaborative flow.

Initial Execution System: batched `move1` programs

The first implementation of end-effector motion relied on sending small URScript programs in batches of two waypoints through the Primary interface (port 30001). Each batch consisted of a short program containing two consecutive `move1` instructions, a `sync()` call, and a `textmsg()` marker to signal completion, followed by a `stop1(1.0)` for a controlled stop. This ensured that each pair of waypoints was executed exactly in sequence, but introduced frequent pauses and synchronisation overhead. Furthermore, as each batch was precomputed, it was not possible to adapt the path dynamically if an obstacle appeared. This approach guaranteed determinism, but at the cost of fluidity, showing the tension between strict reproducibility and responsiveness.

Improved Execution System: streaming with `speed1`

To overcome these limitations, the system was upgraded to use continuous velocity streaming with the `speed1` command on port 30003. Instead of sending precompiled batches, the script

Implementation

`WaypointController` updates the target pose in every `FixedUpdate` cycle. When an obstacle is detected, the current segment $P_1 \rightarrow P_2$ is replaced by a lifted segment via M' , and the new target is used to compute a Cartesian velocity vector.

The robot receives this vector in real time through the `speedl` command:

$$\text{speedl}([v_x, v_y, v_z, \omega_x, \omega_y, \omega_z], a, t)$$

where $[v_x, v_y, v_z]$ are the linear velocity components in m/s, $[\omega_x, \omega_y, \omega_z]$ are the angular velocity components in rad/s, a is the maximum allowed acceleration (m/s², rad/s²), and t is the watchdog timeout in seconds. These commands are streamed at 125 Hz, continuously replacing the previous setpoint and enabling immediate adjustments to the trajectory.

This architecture allows the robot's motion to adapt in real time: the base trajectory remains precomputed and stable, while local detours are injected and removed dynamically. By leveraging high-frequency streaming through `speedl`, the system ensures immediate responsiveness to human motion while preserving overall trajectory smoothness. From a critical standpoint, this choice privileges adaptability and operator comfort over strict reproducibility, a trade-off considered acceptable in collaborative scenarios but which would require further validation for safety-critical industrial deployments.

In practice, both execution systems were implemented and tested. The batch-based `move1` programs guaranteed strict waypoint execution, but suffered from frequent stops and an inability to re-plan. The streaming `speedl` approach reduced the number of stops, enabled smooth adaptation to obstacles, and proved better suited for collaborative scenarios where human presence is dynamic.

5.8 Superposition of holograms via QR Markers

Finally, the superposition of the holograms was made in order to align the real world with the experience. To align the AR coordinate system with the robot base frame, a marker-based calibration was implemented using Unity's `ARSession` and a custom `ARMarkerManager` (QR code detection). A lightweight component, `ScanAndPosition`, subscribes to marker events and updates the robot base transform whenever a valid marker pose is observed, while applying time-based hysteresis to avoid drift when the marker is lost.

Runtime flow

On enable, `ScanAndPosition` registers to `markersChanged`. For each callback, the script records or updates the last-seen timestamp of each marker ID in a dictionary, reads the current world pose of the marker, and recomputes the robot base transform using a fixed marker-to-base offset. Markers that have not been seen for more than 360 s are considered stale and are ignored until they are re-detected.

Pose composition

In robotics and augmented reality, the pose of an object is typically represented by a *homogeneous transformation matrix*. This is a 4×4 matrix belonging to the group $SE(3)$, the *Special Euclidean group in three dimensions*, which encodes both rotation and translation in a single structure. Formally, each element of $SE(3)$ has the form:

$${}^A T_B = \begin{bmatrix} R_{AB} & \mathbf{t}_{AB} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad (5.16)$$

where $R_{AB} \in SO(3)$ is a 3×3 rotation matrix describing the orientation of frame B with respect to frame A , and $\mathbf{t}_{AB} \in \mathbb{R}^3$ is the translation vector giving the position of the origin of B in frame A .

In our calibration process, ${}^W T_M$ denotes the transformation that places the detected QR marker frame M in the AR world frame W , as estimated by the tracking pipeline. A second transformation, ${}^M T_B$, is a fixed offset that maps the marker frame M to the robot base frame B , established during calibration. The pose of the robot base in the AR world is then obtained by matrix multiplication:

$${}^W T_B = {}^W T_M \cdot {}^M T_B. \quad (5.17)$$

Decomposing these matrices into their rotational and translational parts makes the computation explicit. Let $(R_{WM}, \mathbf{t}_{WM})$ be the rotation and translation of the marker in the world, provided by the `ARMarkerManager`. Let $(R_{MB}, \mathbf{t}_{MB})$ be the fixed offset that positions the robot base with respect to the marker. Then the composed base pose is

$$R_{WB} = R_{WM} R_{MB}, \quad (5.18)$$

$$\mathbf{t}_{WB} = \mathbf{t}_{WM} + R_{WM} \mathbf{t}_{MB}. \quad (5.19)$$

In Unity, this corresponds to reading the marker pose $(R_{WM}, \mathbf{t}_{WM})$, applying the fixed offset $(R_{MB}, \mathbf{t}_{MB})$ configured in the Inspector, and updating the base transform with $(R_{WB}, \mathbf{t}_{WB})$ as given by (5.18)–(5.19). The global matrix relation in (5.17) is therefore consistent with the component-wise definition of $SE(3)$ in (5.16).

Axis mapping and scale

Because the UR base frame differs from Unity’s left-handed world, a mapping mode is applied after the composition to keep consistency with motion execution (cf. `WaypointController`). In our implementation we used *Legacy_X_Z_Y*, where the robot’s (X_{UR}, Y_{UR}, Z_{UR}) are mapped from Unity as (X_U, Z_U, Y_U) , and a scale s (typically 1 if 1 Unity unit = 1 m). Formally,

$$\begin{bmatrix} \mathbf{t}_{WB}^{(UR)} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} P & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}}_{\text{axis remap}} \begin{bmatrix} sR_{WB} & \mathbf{t}_{WB} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}, \quad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \quad (5.20)$$

Implementation

The same permutation P is applied to the rotation (or to the equivalent quaternion) to maintain frame consistency.

Stability and timeout

To prevent jitter when the marker is intermittently occluded, `ScanAndPosition` maintains a *lastSeen* timestamp for each `TrackableId` and ignores poses whose marker has not been detected for more than 360s (`disableAfterSeconds`). It can also apply a small interpolation on pose updates, using position lerp and quaternion slerp, to smooth transitions and avoid visible jumps before committing the new ${}^W T_B$ to the Unity scene. This strategy provides a stable anchor for the UR base while still allowing re-alignment whenever the marker is observed again.

Integration with UI and preview

After calibration, the world-aligned base transform becomes the common reference for all components of the system. As illustrated in Figure 5.15, the AR glasses detect the QR code marker placed at the robot base, which provides the necessary reference to align the virtual and physical coordinate frames. This alignment ensures that the ghost arm preview is rendered in the correct position, that waypoints defined in world space correspond accurately to the robot's base frame, and that the communication layer can convert Unity poses into URScript commands with the proper frame and scale.



Figure 5.15: QR-based calibration as seen through the AR glasses. The system detects the pose of the marker in the world frame (${}^W T_M$) and combines it with a fixed offset to the robot base (${}^M T_B$), resulting in the robot base pose in world coordinates (${}^W T_B$).

Implementation

Chapter 6

Validation

6.1 Methodology

The validation of this work followed a two-way approach, combining an objective technical evaluation of execution performance with a subjective user study to assess usability, workload, and acceptance. This dual perspective was necessary to ensure that the proposed system is not only efficient from a robotic execution standpoint but also trusted and usable by human collaborators.

6.1.1 Methodological Alignment with Research Questions

The experimental design was structured to address the research questions introduced in Chapter 1. Two complementary strategies were adopted. The first focused on program efficiency through a controlled experiment in which a predefined trajectory with twenty waypoints was executed under three different system configurations: continuous streaming with `speed1`, batched execution using `move1` commands, and a fixed safety zone baseline. This design allowed a quantitative comparison of execution time, motion continuity, and safety, directly addressing the question of whether adaptive AR-based safety can reduce interruptions compared to static strategies.

The second strategy concentrated on user experience. A participant study combined a guided tutorial with a free exploration phase, encouraging users not only to interact with the system's interface but also to deliberately test its safety features. This design was chosen to investigate how AR visualization influences transparency, trust, usability, workload, and perceived safety. Standardized questionnaires (SUS and NASA-TLX) together with open-ended feedback provided quantitative and qualitative insights into participant perception.

6.1.2 Variables and Metrics

Two categories of variables were analysed. Objective measures from the program efficiency experiment included total task time, number of stops and pauses, and the occurrence of protective stops or collisions. Subjective measures from the participant study included usability scores from

the System Usability Scale, workload ratings from NASA-TLX, and qualitative observations of user behaviour.

6.1.3 Participants

A total of 15 participants took part in the validation study, including students, researchers, and professors at INESC TEC. The group represented a technically inclined but diverse population, with varied levels of experience with AR and 3D interfaces.

6.1.4 Equipment

The experiments were conducted in the Industry and Innovation Laboratory of INESC TEC. The setup included a UR30 robotic arm running with Polyscope, a Microsoft HoloLens 2 headset for AR-based interaction, and the Unity implementation of the collaborative interface deployed to the HMD. The implementation included the waypoint editor, ghost arm preview, and safety mechanisms.

6.2 Experimental Validation: Program Efficiency

The first validation experiment was designed to compare the efficiency, continuity, and safety of three different trajectory execution systems implemented in the collaborative robotic setup. The task consisted of executing a predefined trajectory with 20 waypoints, introduced once by the operator through a hand-guided interaction and an obstacle placed in the trajectory lasting approximately 8 seconds. After this initialization, the robot executed the trajectory under the three different system configurations:

1. **System 1:** Dynamic avoidance using continuous `speedL` streaming commands.
2. **System 2:** Dynamic avoidance using distinct `moveL` executions, in batches of two waypoints at a time.
3. **System 3:** Fixed safety collider without dynamic avoidance (forced stop/resume behavior).

6.2.1 Metrics and Data Collection

Three main metrics were defined and measured during the experiment. The first was **task time**, corresponding to the total duration required to complete the twenty-waypoint trajectory. The second metric was **continuity**, which quantified interruptions in motion. In this case, two types of pauses were observed. A *major pause* refers to a prolonged stop where the robot remains completely idle until the operator withdraws from the restricted zone; this reflects the safety logic holding execution for several seconds before resuming. In contrast, a *soft pause* corresponds to the short interruptions that naturally occur between successive `moveL` commands when the trajectory is executed in batched segments. These appear as small discontinuities but do not represent

safety-related events. Finally, the **safety** metric monitored whether protective hard stops were triggered (with the ideal outcome being zero) and whether any actual collisions occurred (also ideally zero).

The collected data are presented in Table 6.1.

Table 6.1: Experimental results for program efficiency validation on 20 waypoints

Metric	System 1 (speedL)	System 2 (moveL batch)	System 3 (fixed zone)
Task time (s)	15.15	24.27	20.12
Continuity (stops)	0	10 soft pauses	1 major pause
Protective hard stops	0	0	0
Collisions	0	0	0

6.2.2 Results and Discussion

System 1 achieved the fastest execution with an average completion time of 15.15 s. This corresponds to approximately 62% of the time required by System 2 (24.27 s) and 75% of the time required by System 3 (20.12 s). The result highlights the efficiency of continuous `speedL` streaming, which eliminates delays associated with program reloading. System 2 was the slowest, as expected, since it required multiple program transmissions (one per pair of waypoints). Each transmission introduced latency and synchronization delays. System 3 performed intermediately, faster than System 2 but slower than System 1, as no streaming was available but only one interruption occurred.

Regarding continuity, System 1 maintained uninterrupted execution, achieving perfect fluidity of motion. In contrast, System 2 presented ten micro-pauses, directly caused by the block-wise logic of `moveL` execution. This segmented behavior created a visibly less smooth trajectory. System 3 recorded a single major pause when the fixed safety zone was activated, which although less frequent, broke the flow of execution more noticeably.

In terms of safety, all three systems successfully avoided collisions and prevented protective hard stops, meaning that from a purely safety-critical perspective they performed equivalently. However, the mechanisms that ensured safety differed substantially. System 1 adapted in real time through continuous trajectory adjustments, maintaining safety without interruption. System 2 relied on frequent discrete pauses due to its batch-programming structure. System 3 instead depended on a binary stop/resume mechanism, which triggered a complete pause whenever the operator’s hand entered the fixed exclusion zone.

The superior performance of System 1 can be explained by its architectural advantages. Continuous `speedL` streaming allows the robot to adapt instantaneously to obstacles without halting the trajectory and without requiring the transmission of new programs. This ensures both high efficiency and natural, uninterrupted motion.

System 2, although capable of dynamic avoidance, is limited by its need to constantly stop and reload instructions, resulting in poor temporal performance and lack of smoothness. System 3,

while simple and effective in ensuring safety, does not adapt dynamically and only reacts by pausing when the safety zone is violated.

6.2.3 Conclusion of Experiment 1

From this first experiment, it can be concluded that:

1. System 1 (speedL streaming) demonstrated the best balance of efficiency, continuity, and safety. It represents the most suitable strategy for collaborative scenarios where human presence is frequent and unpredictable.
2. System 2 (moveL batch execution) introduced excessive pauses and delays, rendering it less practical despite maintaining safety.
3. System 3 (fixed zone) ensured safety but at the expense of adaptability and fluid collaboration.

Overall, System 1 proved to be the most effective approach, providing rapid task completion, uninterrupted execution, and robust safety simultaneously. These characteristics are fundamental for advancing human-robot collaboration within the principles of Industry 5.0.

6.3 Experimental Validation: Participant Study

The second validation experiment focused on assessing the subjective experience of participants when using the AR-based human–robot collaboration system. The procedure began with a guided tutorial in which participants were introduced to all functions of the system’s user interface. After completing the tutorial, they moved on to a free exploration phase, where they were encouraged to interact with the system at their own pace and to deliberately test its safety features by placing their hands near the trajectory or close to the robot links. An overview of the experimental setup and a sample of the participants involved is shown in Figure [6.1](#).

At the end of the session, all 15 participants filled two standardized questionnaires: the System Usability Scale and the NASA-TLX workload assessment, along with preliminary questions about their prior experience with Augmented Reality and 3D interfaces. These instruments allowed both a quantitative and qualitative assessment of the system’s usability, workload, and user acceptance.

Validation



Figure 6.1: Experimental validation: sample of participants and the environment in which the study was conducted.

6.3.1 System Usability Scale

The results of the SUS questionnaire revealed consistently high usability scores across all participants. The global score was 84.0, which implies that, in terms of usability scale, the project had an excellent result. In terms of willingness to reuse, the system obtained a mean score of 4.40 out of 5, suggesting that participants would welcome frequent interaction with the interface. Ease of use was also highly rated, with an average of 4.47, confirming that the interaction design was intuitive and required little effort to learn.

Regarding complexity, participants clearly perceived the system as simple and coherent: unnecessary complexity was rated at only 1.27 and inconsistency at 1.33, both on a scale of 5. This indicates that the workflow and interaction logic were easy to follow and well structured. Confidence while using the system received a mean score of 4.13, showing that users generally felt secure and in control during operation. Dependence on external support was very low, with a score of 2.00, implying that participants required little to no technical assistance. Finally, the integration of functions was rated positively at 4.33, reflecting that both the tutorial and the overall workflow were perceived as coherent and well integrated.

Taken together, these results validate the system as both highly usable and easily learnable, even for participants without prior experience with Augmented Reality.

6.3.2 NASA-TLX Workload Assessment

The NASA-TLX results indicated that participants perceived the system as minimally demanding across all workload dimensions. Mental demand was rated low, with a mean of 2.73 out of 10, showing that operating the interface required little cognitive effort. Physical demand was also minimal at 1.80, reflecting that interactions through the AR headset and hand gestures were not physically tiring. Temporal demand scored an average of 2.80, suggesting that participants did not feel pressured by time while using the system. Similarly, the overall effort required was rated low, at 2.13, and frustration was almost negligible, with a mean score of 1.40, indicating very low levels of stress or irritation during use.

The only dimension with notable variability was the Performance subscale, which presented a mean of 4.27. Some participants rated their performance very low (1–2), while others chose high values (8–10). This spread is best explained by a known issue with the NASA-TLX instrument: unlike the other subscales, where the left extreme corresponds to a “low” workload, the Performance dimension is inverted, with the left extreme representing perfect performance and the right extreme representing failure. Previous studies have reported similar inconsistencies and in some cases corrected the scale to align directionality with the other dimensions [HS06, SGR⁺20]. In this study no such correction was applied, and the variability observed is therefore interpreted as a misinterpretation of the inverted scale rather than a true reflection of differences in performance. Since all participants successfully completed both the tutorial and the subsequent tasks, actual performance can be considered high, and the apparent inconsistency in the Performance ratings is treated as an artifact of the measurement tool rather than a limitation of the system.

6.3.3 Impact of User AR Experience

Initial questions distinguished participants by their familiarity with AR and 3D interfaces. Analysis indicates that:

- Participants with little to no AR experience still rated the system highly in terms of usability (SUS > 4 across most items).
- Variability in NASA-TLX performance ratings appears correlated with AR background: experienced participants tended to rate their success higher, while less experienced participants were more critical of themselves despite completing tasks successfully.
- The structured tutorial effectively mitigated differences in prior AR knowledge, enabling all participants to operate the system confidently.

6.3.4 Observed Behavioral Patterns

During the exploration phase, several qualitative patterns emerged from participant behavior. The tutorial proved effective in onboarding new users, as it equalized skill levels across the group and

reduced the impact of prior experience with Augmented Reality. Once confident with the interface, participants deliberately tested the safety mechanisms by attempting to obstruct the robot's trajectory. These actions reinforced their trust in the system when they observed that motion was immediately paused or redirected in response to their interventions. In addition, many participants chose to extend their interaction beyond the required tasks, showing curiosity and engagement that suggested they found the system enjoyable to use.

6.3.5 Conclusions of Experiment 2

The participant study validates the system as a usable, intuitive, and safe interface for human-robot collaboration. Specifically:

1. **Technological Accessibility:** The tutorial design successfully lowered entry barriers, allowing novices to operate the system with ease. Prior AR experience did not prevent users from achieving positive results.
2. **Low workload:** NASA-TLX confirmed that interaction required little mental, physical, or temporal effort, supporting a natural and stress-free user experience.
3. **Confidence and safety:** Direct testing of collision avoidance and protective behavior fostered trust in the system's robustness, a critical factor in human-robot collaboration.
4. **Performance perception:** While raw NASA-TLX performance data appeared inconsistent, this was attributed to scale inversion. In practice, all participants completed tasks successfully, meaning actual performance was high.
5. **User acceptance:** High SUS ratings demonstrate that the system was perceived as consistent, well-integrated, and enjoyable to use, which is essential for future adoption in industrial and research contexts.

In addition to these quantitative and qualitative findings, participants provided valuable comments regarding possible improvements. One recurrent suggestion was to anchor the user interface menu to the hand rather than leaving it fixed in world space. This feature has already been added in subsequent iterations, and it effectively mitigates confusion by ensuring that users can freely roam the environment without losing sight of the menu; raising the hand instantly brings the menu into view. Another suggestion concerned adding adjustable safety parameters, such as configuring the maximum allowed height, defining the distance thresholds for soft pauses near the robot links, and other customizable safety margins. Such personalization would provide users with greater control over the system and was strongly endorsed for future development.

Taken together, these results indicate that the proposed AR-supported interface is both effective and engaging for diverse users. The combination of high usability, low workload, trust in safety mechanisms, and responsiveness to user feedback provides strong evidence that the system is suitable for integration into real-world collaborative workspaces, advancing the principles of Industry 5.0 where humans and robots share the same environment.

Validation

Chapter 7

Conclusions and Future Work

This dissertation addressed the development and validation of an Augmented Reality interface to support safe and efficient human–robot collaboration. The methodology was designed to be applicable to collaborative manipulators in general, requiring only minor adaptations to account for specific kinematic parameters or communication protocols. For demonstration purposes, the implementation was carried out with a Universal Robots UR30, within the context of Industry 5.0 principles, where humans and robots share the same workspace in a cooperative manner.

The work combined the design of a multimodal AR interface with real-time control and safety mechanisms for collaborative manipulation. The interface allowed users to create and manage waypoints through a holographic representation of the robot (ghost arm), preview and execute trajectories, and interact with safety features such as dynamic avoidance and protective stops. A tutorial was also included to guide users through the workflow.

Validation followed two complementary strategies. The first consisted of controlled system efficiency tests in which three control approaches, continuous streaming with `speedL`, batched execution with `moveL`, and a fixed safety-zone baseline, were compared in terms of execution time, continuity, and safety. The results showed that dynamic avoidance with `speedL` provided the best trade-off, combining fluidity of motion with safe operation. The second strategy was a user study involving fifteen participants, including students, researchers, and professors. After completing both the System Usability Scale and NASA-TLX questionnaires, participants rated the system as highly usable, intuitive, and safe, with a consistently low workload. Although some variability was observed in the NASA-TLX Performance scale, this was attributed to the known inversion issue in that subscale rather than to true differences in outcomes. All participants successfully completed their tasks, reinforcing the system’s effectiveness. Qualitative feedback further highlighted strong engagement with the interface and trust in the implemented safety mechanisms.

7.1 Conclusions

Overall, the objectives defined for this dissertation were successfully achieved. The research questions can now be answered as follows:

- **RQ1 – How can AR visualization improve transparency and trust in collaborative robotic systems?** The integration of holographic overlays, movement previews, and contextual alerts allowed operators to clearly understand the robot’s intentions and upcoming actions. The user study confirmed that these features increased predictability, reduced hesitation, and fostered trust in the robot’s behavior.
- **RQ2 – To what extent does an adaptive AR safety framework reduce unnecessary interruptions compared to static safety zones?** The efficiency experiments demonstrated that dynamic avoidance eliminated unnecessary stops while maintaining the same level of protection as static zones. This proved that adaptive AR-based safety can preserve safety standards while significantly improving workflow continuity and execution time.
- **RQ3 – What is the effect of AR-based interaction on usability, workload, and operator perception of safety?** SUS scores indicated high usability and consistency, while NASA-TLX revealed low mental, physical, and temporal workload. Participants expressed high confidence in the safety mechanisms and strong acceptance of the system. These results confirm that AR-based interaction is both usable and effective, reducing operator workload while enhancing perceived safety.

Thus, the system developed in this work enables intuitive human–robot interaction, validates adaptive safety mechanisms as efficient and reliable, and demonstrates that AR can support both trust and usability in collaborative robotics.

7.2 Main Contributions

The contributions of this work span both technical development and experimental validation. First, an Augmented Reality interface was created to support collaborative trajectory creation and execution with Universal Robots manipulators, providing operators with an intuitive means of interacting with the robot. Real-time safety mechanisms were implemented, combining dynamic avoidance, protective stops, and collider-based monitoring of the operator’s hands and head to ensure safe interaction.

A holographic visualization component, the “ghost arm”, was integrated into the system, serving both as a tool for waypoint editing and as a preview mechanism for upcoming robot motions. To make the interface accessible to non-expert operators, a user-friendly tutorial and workflow were designed, guiding participants step by step through the system’s functionality. Finally, the complete solution was validated experimentally through efficiency tests and a user study with 15 participants. These evaluations confirmed that the system delivers high usability, low workload, and strong user acceptance, thereby demonstrating its potential for practical deployment in collaborative industrial environments.

7.3 Future Work

While the system achieved its objectives, several directions remain open for further development. One area concerns the user interface, which could benefit from additional personalization options. Allowing users to configure safety parameters, such as the maximum allowed height, safety distances that trigger soft pauses, and other thresholds, would make the system more flexible and adaptable to different contexts. Building on the implemented feature of attaching menus to the user's hand, the interface could also evolve towards a more adaptive design, improving menu anchoring and overall intuitiveness of interaction.

Another important direction is extended validation. Larger-scale participant studies across different industrial scenarios, and involving users with diverse backgrounds, would strengthen the evidence of usability and robustness. Finally, future work could expand the scope of collaborative tasks. Moving beyond waypoint editing, the system could support shared task planning, adaptive role allocation, and even multi-robot collaboration. The addition of a mobile base to the robot would further broaden the range of possible applications and provide a richer environment for testing advanced human–robot interaction strategies. Finally, the system would benefit from implementing accessibility features to help operators with physical restrictions interact more easily. Adding options such as gaze and voice commands could significantly reduce these barriers, allowing users with limited mobility to operate the system independently.

7.4 Final Remarks

In summary, this work demonstrated the feasibility and benefits of an AR-supported interface for human-robot collaborative workspaces. The system met its objectives, answered the proposed research questions, and provided novel contributions in usability and safety integration. These outcomes contribute to the advancement of human-robot interaction within Industry 5.0, enabling safer, more intuitive, and more effective collaboration between humans and robots.

Conclusions and Future Work

References

- [adl24] Conception of a robotic digital shadow in augmented reality. *Procedia CIRP*, 2024. Cited on page [11](#).
- [AFS23] Mihai Arde, Julia Fink, and Ulrich Schreiber. Ardie: Ar dialogue and eye gaze policies for human–robot collaboration. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 221–231. IEEE, 2023. Cited on page [9](#).
- [APVM24] Antonios Angelidis, Emmanuel Plevritakis, George-Christopher Vosniakos, and Elias Matsas. An open extended reality platform supporting dynamic robot paths for studying human–robot collaboration in manufacturing. *The International Journal of Advanced Manufacturing Technology*, September 2024. Cited on pages [19](#), [20](#), and [21](#).
- [AWM⁺24] Usama Ali, Lan Wu, Adrian Mueller, Fouad Sukkar, Tobias Kaupp, and Teresa Vidal-Calleja. Interactive Distance Field Mapping and Planning to Enable Human-Robot Collaboration, October 2024. arXiv:2403.09988 [cs]. Cited on pages [2](#), [16](#), [17](#), [20](#), and [21](#).
- [Azu97] Ronald T. Azuma. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, August 1997. Cited on page [8](#).
- [BMAY⁺22] Achim Buerkle, Harveen Matharu, Ali Al-Yacoub, Niels Lohse, Thomas Bamber, and Pedro Ferreira. An adaptive human sensor framework for human–robot collaboration. *The International Journal of Advanced Manufacturing Technology*, 119(1):1233–1248, March 2022. Cited on page [3](#).
- [Bor14] D. Bortot. *Ergonomic Human-robot Coexistence in the Branch of Production*. Robotik und Automation. Verlag Dr. Hut, 2014. Cited on page [7](#).
- [Bro96] John Brooke. SUS – a quick and dirty usability scale. pages 189–194. January 1996. Cited on page [11](#).
- [caa24] Designing a dynamically configurable digital twin for human–robot collaboration tasks. In *CAADRIA 2024*, 2024. Cited on page [11](#).
- [CC⁺24] Yuan-Hao Chang, Jui-Chen Chen, et al. Augmented reality for human–robot collaboration: A survey. *Machines*, 12(8):540, 2024. Cited on page [8](#).
- [CPD20] Peter Chemweno, Liliane Pintelon, and Wilm Decre. Orienting safety assurance with outcomes of hazard analysis and risk assessment: A review of the ISO 15066 standard for collaborative robot systems. *Safety Science*, 129:104832, September 2020. Cited on page [1](#).

REFERENCES

- [CPM22] Gabriel de Moura Costa, Marcelo Roberto Petry, and António Paulo Moreira. Augmented Reality for Human–Robot Collaboration and Cooperation in Industrial Applications: A Systematic Literature Review. *Sensors*, 22(7):2725, January 2022. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute. Cited on pages 3 and 9.
- [CPMM24] Gabriel M. Costa, Marcelo R. Petry, João G. Martins, and António Paulo G. M. Moreira. Assessment of Multiple Fiducial Marker Trackers on Hololens 2. *IEEE Access*, 12:14211–14226, 2024. Cited on page 9.
- [dir06] Directive 2006/42/ec of the european parliament and of the council of 17 may 2006 on machinery, and amending directive 95/16/ec (recast). Official Journal of the European Union, L 157, 9 June 2006, p. 24–86, 2006. Cited on page 18.
- [dt23] Digital twin for human–robot collaboration enhancement in manufacturing systems: Review and outlook. *Computers & Industrial Engineering*, 2023. Cited on page 10.
- [DUR⁺16] Ana M. Djuric, R. J. Urbanic, J. L. Rickli, Ana M. Djuric, R. J. Urbanic, and J. L. Rickli. A Framework for Collaborative Robot (CoBot) Integration in Advanced Manufacturing Systems. *SAE International Journal of Materials and Manufacturing*, 9(2):457–464, April 2016. Publisher: SAE International. Cited on page 1.
- [HFB⁺23] Ali M. Hosseini, Clara Fischer, Mukund Bhole, Wolfgang Kastner, Thilo Sauter, and Sebastian Schlund. A Safety and Security Requirements Management Methodology in Reconfigurable Collaborative Human-Robot Application. In *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS)*, pages 1–8, April 2023. ISSN: 2835-8414. Cited on page 18.
- [hos24] *Real-Time Adaptive Industrial Robots: Improving Safety And Comfort In Human-Robot Collaboration*. September 2024. Cited on page 3.
- [HS88] Sandra G. Hart and Lowell E. Staveland. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In Peter A. Hancock and Najmedin Meshkati, editors, *Advances in Psychology*, volume 52 of *Human Mental Workload*, pages 139–183. North-Holland, January 1988. Cited on page 11.
- [HS06] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in Psychology*, volume 52, pages 139–183. Elsevier, 2006. Cited on page 58.
- [iso11] Robots and robotic devices — safety requirements for industrial robots — part 1 and 2, 2011. Defines safety requirements for the design, integration, and use of industrial robots. Cited on page 6.
- [iso16] Robots and robotic devices — collaborative robots, 2016. Technical specification providing safety requirements and guidance for collaborative robot operations. Cited on page 6.
- [KKM⁺22] Panagiotis Karagiannis, Niki Kousi, George Michalos, Konstantinos Dimoulas, Konstantinos Mparis, Dimosthenis Dimosthenopoulos, Önder Tokçalar, Toni Guasch, Gian Paolo Gerio, and Sotiris Makris. Adaptive speed and separation monitoring based on switching of safety zones for effective human robot collaboration.

REFERENCES

- Robotics and Computer-Integrated Manufacturing*, 77:102361, October 2022. Cited on pages [2](#), [16](#), [17](#), [20](#), and [21](#).
- [KLF23] Daehyun Kim, Xinyu Luo, and Kerstin Fischer. Multimodal interfaces for human–robot collaboration: Combining speech, gesture, and gaze in mixed reality. *IEEE Transactions on Human-Machine Systems*, 53(4):620–633, 2023. Cited on page [10](#).
- [Mar25] Costa Petry Martins, Moura. Ar framework for skill-based programming of industrial mobile manipulators. June 2025. Cited on pages [18](#), [20](#), and [21](#).
- [MCP⁺24] João G. Martins, Gabriel M. Costa, Marcelo R. Petry, Paulo Costa, and António Paulo Moreira. A Multi-User Multi-Robot Collaboration through Augmented Reality. In *2024 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 144–150, May 2024. ISSN: 2573-9387. Cited on page [2](#).
- [MK94] Paul Milgram and Fumio Kishino. A Taxonomy of Mixed Reality Visual Displays. *IEICE Trans. Information Systems*, vol. E77-D, no. 12:1321–1329, December 1994. Cited on page [8](#).
- [MMZ⁺19] Eloise Matheson, Riccardo Minto, Emanuele G. G. Zampieri, Maurizio Faccio, and Giulio Rosati. Human–Robot Collaboration in Manufacturing Applications: A Review. *Robotics*, 8(4):100, December 2019. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute. Cited on pages [6](#) and [7](#).
- [OS⁺23] Muhamad Zaimi Othman, Fahmi I. M. Salleh, et al. Human–robot collaboration in smart manufacturing: A systematic review. *Applied Sciences*, 13(11):6440, 2023. Cited on page [5](#).
- [Par] Roman Parák. rparak/Unity3D_robotics_ur: A digital-twin of the Universal Robots UR3 integrated into the Unity3D development platform. Cited on pages [28](#) and [33](#).
- [Pre25] Preliy. Preliy/Flange, September 2025. original-date: 2025-01-29T21:17:25Z. Cited on page [35](#).
- [RKSS24] Igor Rybalskii, Karl Kruusamäe, Arun Kumar Singh, and Sebastian Schlund. An Augmented Reality Interface for Safer Human-Robot Interaction in Manufacturing. *IFAC-PapersOnLine*, 58(19):581–585, January 2024. Cited on pages [18](#), [19](#), [20](#), and [21](#).
- [Rob23] Universal Robots. Dh parameters for calculations of kinematics and dynamics, 2023. Cited on pages [xiii](#), [xv](#), [35](#), and [36](#).
- [RSV⁺23] Raffik R, Rithish Ramamoorthy Sathya, Vaishali V, Balavedhaa S, and Jyothi Lakshmi N. Industry 5.0: Enhancing Human-Robot Collaboration through Collaborative Robots – A Review. In *2023 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, pages 1–6, June 2023. Cited on pages [1](#) and [3](#).
- [SBSPM21] Inês Soares, Ricardo B. Sousa, Marcelo Petry, and António Paulo Moreira. Accuracy and Repeatability Tests on HoloLens 2 and HTC Vive. *Multimodal Technologies and Interaction*, 5(8):47, August 2021. Publisher: Multidisciplinary Digital Publishing Institute. Cited on pages [9](#) and [10](#).

REFERENCES

- [SGR⁺20] Sadiq Said, Malgorzata Gozdzik, Tadzio Raoul Roche, Julia Braun, Julian Rössler, Alexander Kaserer, Donat R Spahn, Christoph B Nöthiger, and David Werner Tscholl. Validation of the raw national aeronautics and space administration task load index (nasa-tlx) questionnaire to assess perceived workload in patient monitoring tasks: Pooled analysis study using mixed models. *Journal of Medical Internet Research*, 22(9):e19472, 2020. Cited on page [58](#).
- [SPM21] Inês Soares, Marcelo Petry, and António Paulo Moreira. Programming Robots by Demonstration Using Augmented Reality. *Sensors*, 21(17):5976, January 2021. Number: 17 Publisher: Multidisciplinary Digital Publishing Institute. Cited on pages [2](#), [15](#), [16](#), [20](#), and [21](#).
- [SYH22] Ryota Suzuki, Go Yamamoto, and Satoshi Hashimoto. Taxonomy and survey of augmented reality enhanced human–robot interaction. *arXiv preprint arXiv:2203.03254*, 2022. Cited on page [8](#).
- [SZH24] Akshay Subramanian, Zeyu Zhang, and Bradley Hayes. Enhancing human–robot interaction through augmented reality: A comprehensive review and framework. *ACM Transactions on Human–Robot Interaction*, 2024. Cited on pages [2](#), [6](#), [8](#), and [11](#).
- [SZH25] Akshay Subramanian, Zeyu Zhang, and Bradley Hayes. Improving human–robot collaboration with augmented reality and eye gaze communication. In *Proceedings of the 2025 ACM/IEEE International Conference on Human–Robot Interaction*, pages 45–55. ACM/IEEE, 2025. Cited on page [9](#).
- [Uni19] Universal Robots Forum. Overview of used ports on local host, 2019. Cited on page [12](#).
- [Vic20] Federico Vicentini. Terminology in safety of collaborative robotics. *Robotics and Computer-Integrated Manufacturing*, 63:101921, June 2020. Cited on pages [6](#) and [11](#).
- [WMB⁺16] Bauer Wilhelm, Bender Manfred, Martin Braun, Peter Rally, and Oliver Scholtz. *Lightweight robots in manual assembly – best to start simply! Examining companies’ initial experiences with lightweight robots*. October 2016. Cited on pages [7](#) and [8](#).
- [YAK⁺25] Xiliu Yang, Felix Amtsberg, Benjamin Kaiser, et al. An implementation and evaluation of large-scale multi-user human–robot collaboration with head-mounted augmented reality. *Advanced Engineering Informatics*, 67:103475, 2025. Cited on page [8](#).
- [YLS25] Wei Yang, Zhen Li, and Meng Sun. Large-scale multi-user human–robot collaboration using augmented reality head-mounted displays. *Robotics and Computer-Integrated Manufacturing*, 87:102847, 2025. Cited on page [12](#).

Appendix A

Demonstration Video

A video demonstration of the AR-based human–robot collaboration system is available [here](#).



Figure A.1: Thumbnail of the demonstration video. Click the image to open the video online.

The montage showcases the complete workflow, including calibration through a QR code, connection with the UR30 robot, waypoint creation and preview with the ghost arm, execution of trajectories, and the safety mechanisms that pause or adapt motion in response to the operator's proximity.

Demonstration Video

Appendix B

System Usability Scale - Answers

I think that I would like to use this system frequently.

15 respostas

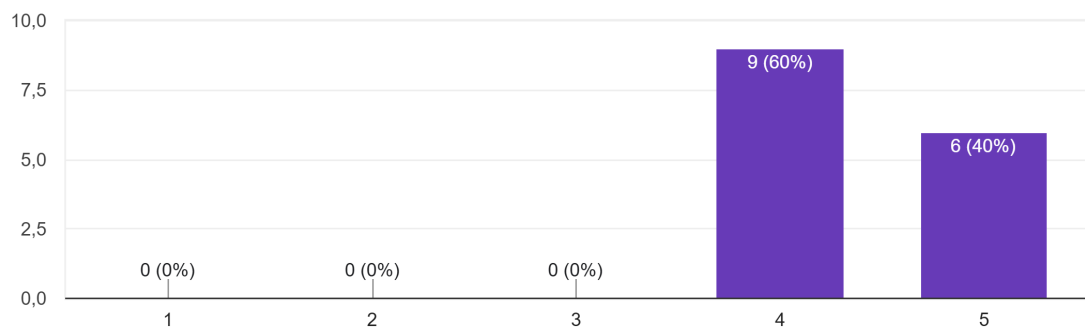


Figure B.1: SUS Question 1: I think that I would like to use this system frequently.

System Usability Scale - Answers

I found the system unnecessarily complex.

15 respostas

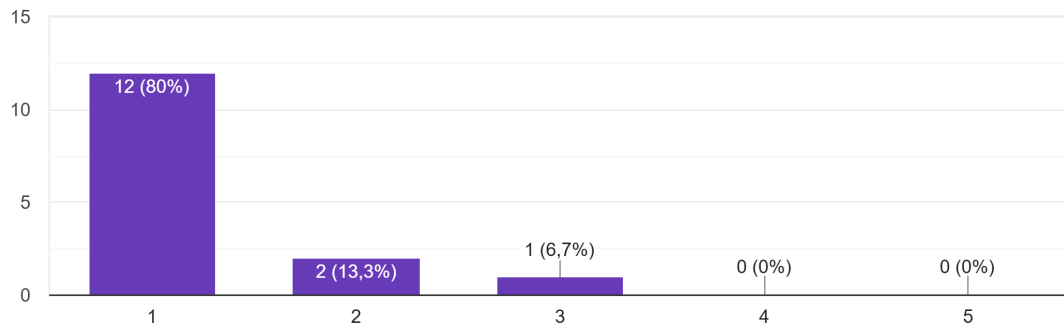


Figure B.2: SUS Question 2: I found the system unnecessarily complex.

I thought the system was easy to use.

15 respostas

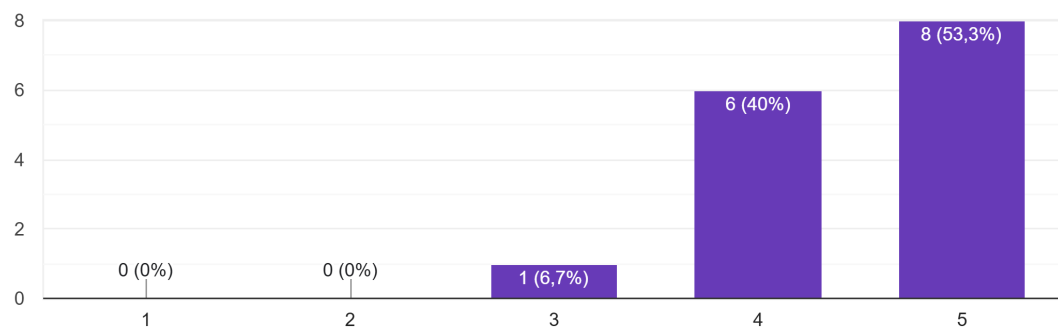


Figure B.3: SUS Question 3: I thought the system was easy to use.

System Usability Scale - Answers

I think that I would need the support of a technical person to be able to use this system.

15 respostas

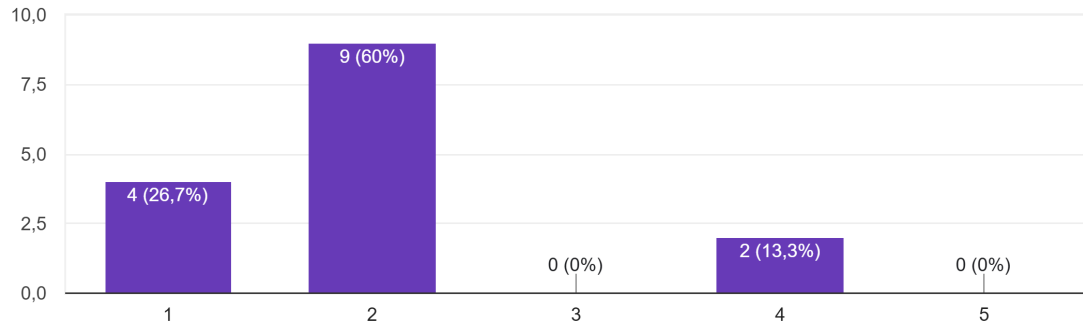


Figure B.4: SUS Question 4: I think that I would need support of a technical person to be able to use this system.

I found the various functions in this system were well integrated.

15 respostas

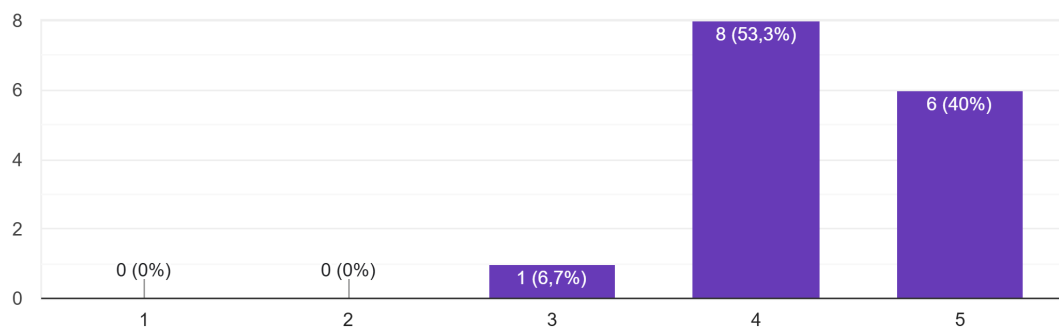


Figure B.5: SUS Question 5: I found the various functions in this system were well integrated.

System Usability Scale - Answers

I thought there was too much inconsistency in this system.

15 respostas

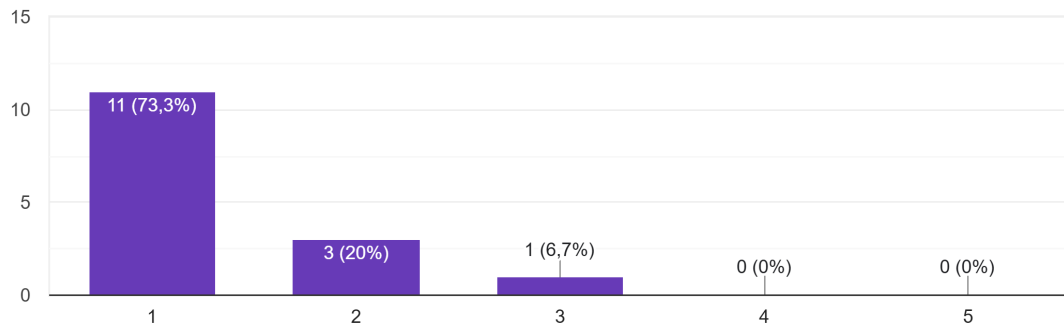


Figure B.6: SUS Question 6: I thought there was too much inconsistency in the system.

I would imagine that most people would learn to use this system very quickly.

15 respostas

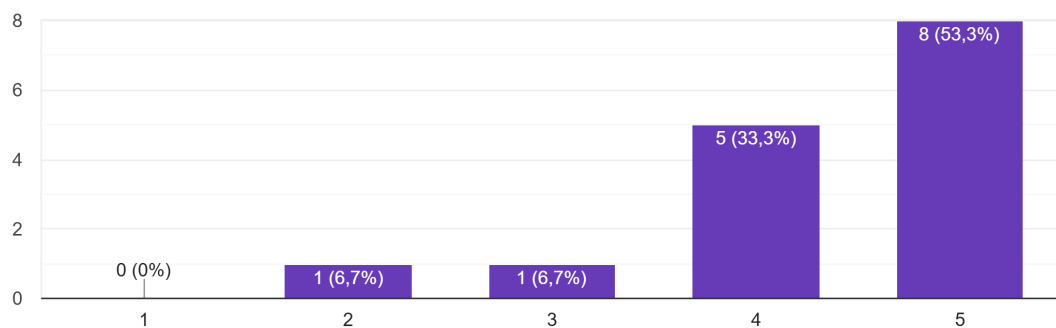


Figure B.7: SUS Question 7: I would imagine that most people would learn to use this system very quickly.

System Usability Scale - Answers

I found the system very cumbersome/awkward to use.

15 respostas

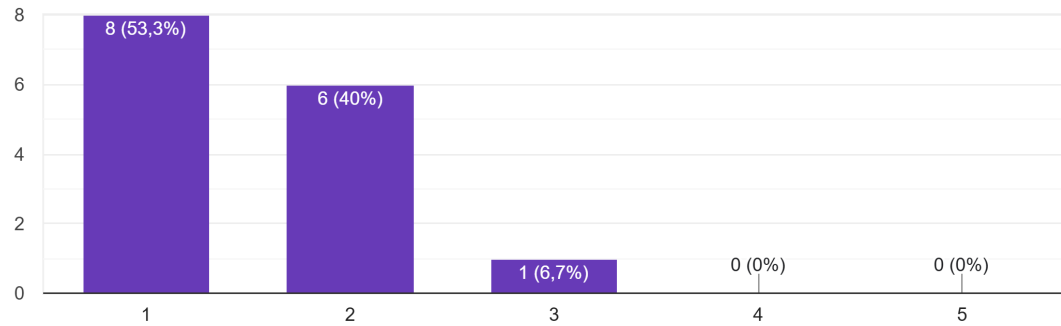


Figure B.8: SUS Question 8: I found the system very cumbersome/awkward to use.

I felt very confident using the system.

15 respostas

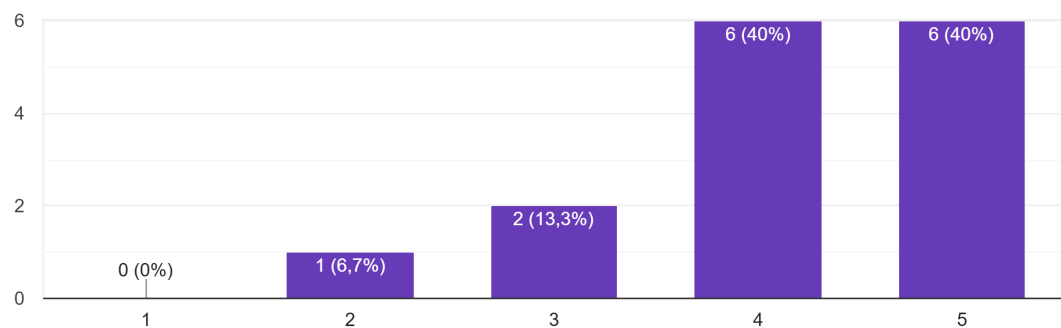


Figure B.9: SUS Question 9: I felt very confident using the system.

System Usability Scale - Answers

I needed to learn a lot of things before I could get going with this system.

15 respostas

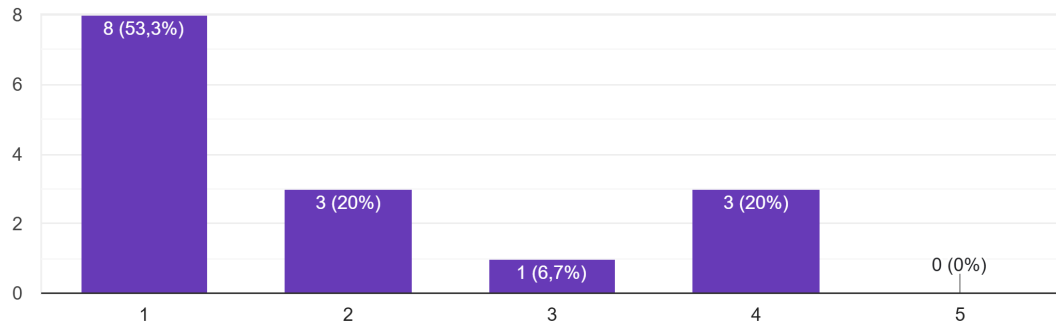


Figure B.10: SUS Question 10: I needed to learn a lot of things before I could get going with this system.

Appendix C

NASA-TLX - Answers

Mental Demand – How mentally demanding was the task?

15 respostas

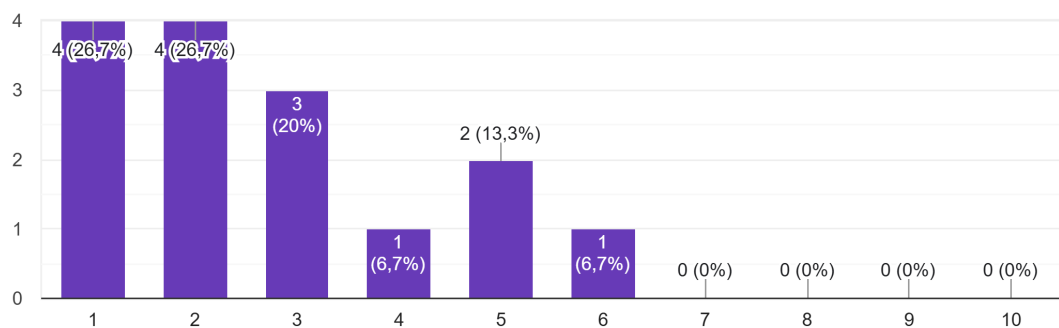


Figure C.1: NASA-TLX Question 1: Mental Demand - How mentally demanding was the task?

NASA-TLX - Answers

Physical Demand – How physically demanding was the task?

15 respostas

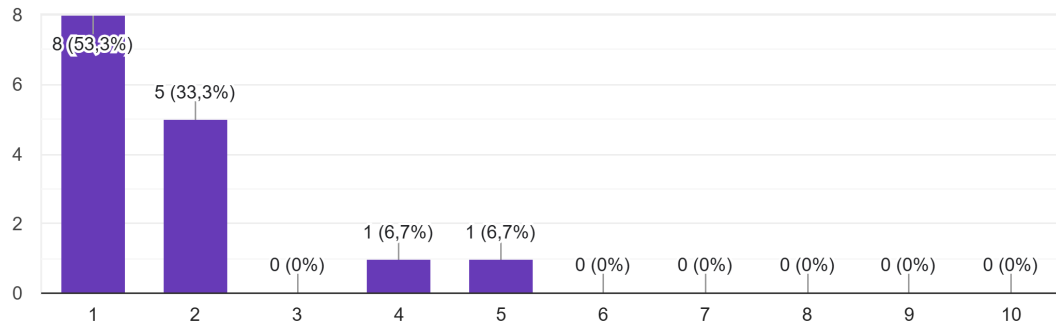


Figure C.2: NASA-TLX Question 2: Physical Demand - How physically demanding was the task?

Temporal Demand – How hurried or rushed was the pace of the task?

15 respostas

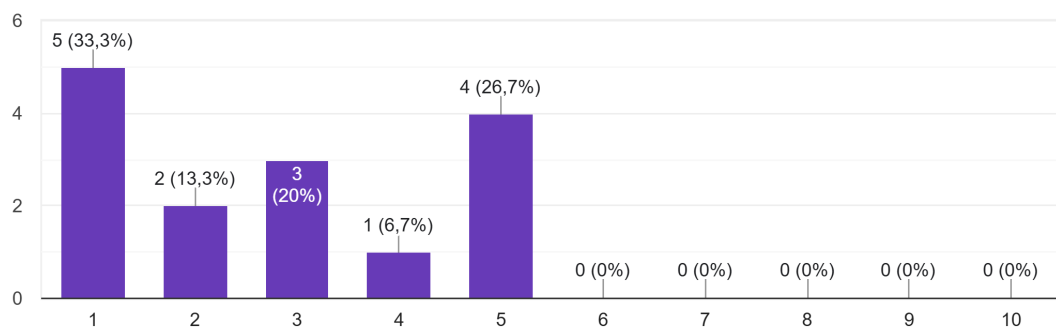


Figure C.3: NASA-TLX Question 3: Temporal Demand - How hurried or rushed was the pace of the task?

NASA-TLX - Answers

Performance – How successful were you in accomplishing what you were asked to do?

15 respostas

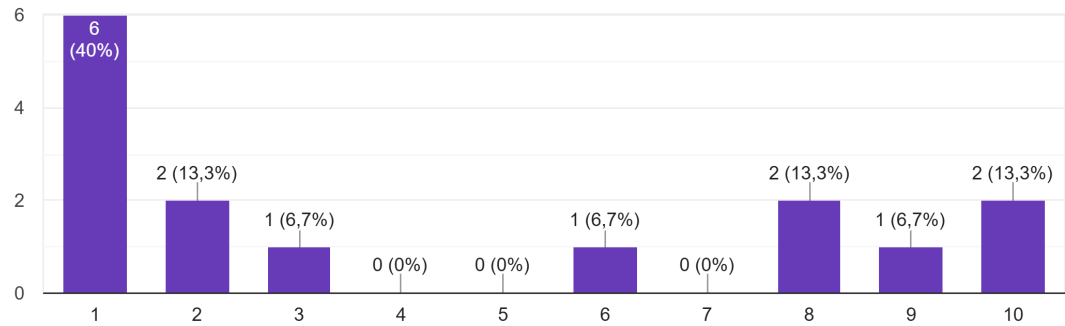


Figure C.4: NASA-TLX Question 4: Performance - How successful were you in accomplishing what you were asked to do?

Results in **'Performance'** are mixed due to scale inversion, however everyone fulfilled their task.

Effort – How hard did you have to work to accomplish your level of performance?

15 respostas

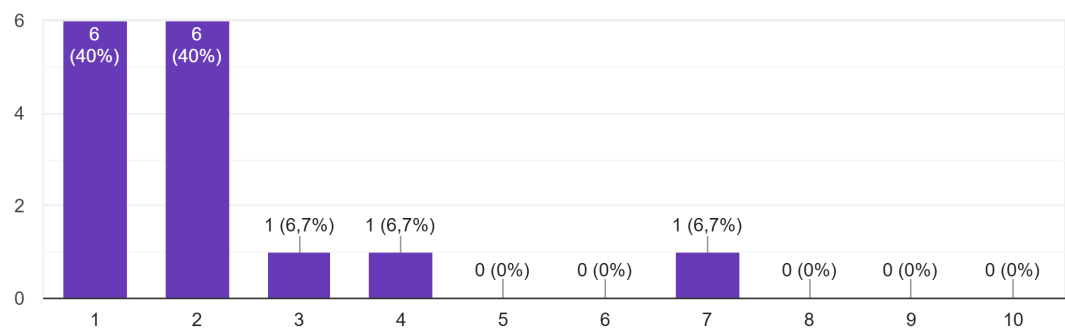


Figure C.5: NASA-TLX Question 5: Effort - How hard did you have to work to accomplish your level of performance?

NASA-TLX - Answers

Frustration – How insecure, discouraged, irritated, stressed, and annoyed were you?

15 respostas

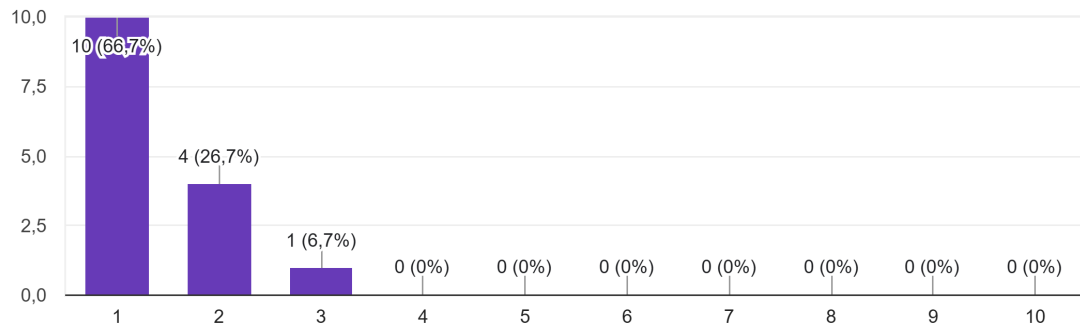


Figure C.6: NASA-TLX Question 6: Frustration - How insecure, discouraged, irritated, stressed, and annoyed were you?

Appendix D

Open feedback from participants

Open feedback

If you have any comment/suggestion, please leave it here. Thank you for your contribution!

5 respostas

Congrats Vitor Hugo!

:)

The system is incredibly interesting and intuitive to use, providing a good tutorial for it's functionalities.

.

Fixar o menu a um braço do utilizador;
Adicionar parametrização dentro do ambiente de AR;
Trocar o aviso sonoro constante da entrada na zona do robô por apenas um alerta sonoro, seguido de um indicador visual constante.

Figure D.1: Open feedback, comments or suggestions from participants of the experimental study.