UNIVERSITY OF GENOVA

MASTER'S DEGREE PROGRAM IN ROBOTICS ENGINEERING

Augmented Reality Guided Robotic Manipulation

by

Abdelouadoud Guelmami

Thesis submitted for the degree of Master of Science in Robotics Engineering

March 2025

Carmine Recchiuto — Supervisor

Dibris

Department of Informatics, Bioengineering, Robotics and Systems Engineering

I would like to dedicate this thesis, first and foremost, to my family. Without them, I wouldn't be where I am today. Specifically, I dedicate it to my mother, who was the first person to know about my decision and supported me without hesitation. To my father, who has stood by me since day one, as well as my aunt Sellam Fadhila and finally to my beautiful little sister Abir. Also, I dedicate this thesis to my two close friends, Akram Kalkoul and Mouadh Boutaghane as an appreciation for them and for their friendship.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Wadoud Guelmami
March 2025

Acknowledgements

The completion of this thesis marks the culmination of a journey filled with challenges, growth, and support. During this inspiring chapter of my life, I have been fortunate to receive the unwavering encouragement of my family, the guidance of my professors, and the companionship of my close friends. For this, I extend my deepest gratitude, so thank you everyone.

My deepest appreciations go to my supervisor Prof. Carmine Recchiuto, whose support and guidance were invaluable in completing this work. I am also thankful to Dr. Francesco Wanderlingh and Dr. Omotoye for their assistance.

If my career flourishes in the future, the major credit goes to Italy, both its people and government that facilitated for me the opportunities and funding. I would never forget what Italy has done for me. In return, I will seek to give back in some way, even if only a small fraction as a sign of appreciation. My warmest regards to this unique country.

I am extremely grateful to my family for standing by my side and making me believe in myself and constantly reminding me that the end is near and all the efforts will be worth it.

I consider this thesis as a token of the golden days throughout my master's degree, cherishing the friendship I made, the remarkable moments I lived, the lessons I learned and the trips I enjoyed across different 25 countries I visited. This thesis for me is not only just an academic work, but also a reminder of the beautiful days I had and an encouragement to be better and improve continuously.

Abstract

This thesis presents the development and evaluation of an Augmented Reality (AR)-based interface that allows the user to freely interact and control the Franka Emika Panda robotic arm using Hololens2, aiming to enhance usability and human-robot interaction. By integrating AR interface with a robotic control framework, the system bridges the gap between virtual interaction and physical execution, enabling intuitive and precise robot manipulation.

The system employs Unity3D and Hololens2 to provide an interactive AR interface, featuring sliders to adjust the desired Cartesian coordination point in 3D space. This point is visually represented as a shpere which can be freely draggable and movable in space serving as a target location of the robot's end effector. Moreover, the possibility to close and open the gripper is available. The interface contains also buttons to plan the motion of the holographic robot and the option to execute the movement to the Real Robot.

The communication between Unity and ROS is established using ROS-TCP connection which enables real time data exchange between the AR interface and the control system, this connection facilitates the transfer of the commands and the feedback. The movement of end effector is accomplished thanks to ROS and MoveIt that are utilized for motion planning and control, where MoveIt computes the optimal trajectory of the robotic arm and ROS processes with the execution of the commands.

Additionally, UDP connection plays the main part of sending the data in order to execute the motion to the real robot by forwarding the desired joint-states to a separate computer connected directly to Franka Panda Robot, this step ensure the motion can be executed in real-time and mimic the same motion of the virtual robot.

User trials demonstrated the system's effectiveness in performing a task such as grasping and placing objects using Hololens2. Usability evaluations using the System Usability Scale (SUS) and NASA Task Load Index (NASA-TLX) highlighted high user satisfaction, low workload, and reliable task execution. This work underscores the potential of AR-based interfaces in robotics, offering novel methodologies for intuitive, efficient, and precise robotic control.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background

Robotic systems have become an essential part of modern industries and daily life, enhancing automation, precision, and efficiency in various tasks. To this end, a major area of development in this field is the integration of human-robot interaction (HRI) strategies that foster intuitive ways for humans to control and collaborate with robots. Several technologies contribute to the advancement of HRI, among which augmented reality (AR) holds a strategic position, as it has the potential to bring humans and machines closer through an interactive and visual interface.

Augmented reality (AR) is a technology that enables users to visualize digital information overlaid onto the real world, providing effective solutions for instantly controlling robots while performing tasks. AR is particularly beneficial for tasks such as robotic manipulation, assembly, or surgery, where precision and adaptability are essential.

This thesis presents an AR-based control system for the Franka Emika Panda robotic arm. The proposed system leverages the Unity3D engine along with HoloLens 2 to provide an immersive AR interface, allowing users to handle tasks such as grasping and object manipulation more seamlessly. The interface includes virtual sliders that enable precise control of the robot's motion, buttons for performing specific operations, and real-time feedback that gently guides the user during interactions. The robotic system also utilizes ROS for robotic control and MoveIt for path planning, ensuring smooth and efficient task execution.

By enhancing usability, precision, and feedback mechanisms, this project addresses key challenges in human-robot collaboration. The goal is to demonstrate how AR can simplify

robotics control and provide the operational capabilities of complex systems, such as the Franka Panda robot, to users who may not be deeply versed in the technology.

1.2 Motivation

This research is motivated by the need for more intuitive and efficient ways to interact with robotic systems, especially via complex dynamic tasks. Conventional robotic control approaches have high training and technical barriers, resulting in not being widely used in non-specialist environments. If we integrate AR with such robots a possible application may produce a huge bond between human and machine with more realistic experience of controlling and analyzing in real-time.

Our research aims at tackling the challenge of providing a homogeneous interaction between AR technology and coordination with robotics. The human-friendly robot Franka Emika Panda is particularly well suited for exploring how AR can be used to support precision, adaptability, and usability in tasks involving object manipulation and assembly. Unity3D, HoloLens 2, and ROS provide different aspects of the solution, such that when combined, they can yield an AR-based control system that can address the challenges presented above by offering real-time feedback, intuitive interaction with the control mechanism, and precise motion control at a higher level.

Improving the integration of AR with robotics, this has the potential to lower the steep learning curve associated with using complex robotic systems, which significantly broadens the range of users who can apply them to domains like manufacturing, healthcare, and research. This study adds to the emerging domain of human-robot interaction by showcasing the potential of AR to transform the way robots can be controlled and operated in both simulated and practical applications.

1.3 Thesis Outline

This thesis is composed by the following chapters:

Chapter 2: This chapter discusses the state of the art in terms of existing research, strategies, and technologies in human-robot interaction (HRI). The article starts with a thorough look at HRI strategies in detail, such as collaborating robots, teleoperation, and more. It then explores critical application areas in which these strategies are applied, including manufacturing, healthcare and research. Finally, it provides usage cases that demonstrate the current strengths and weaknesses of HRI technologies, thus laying the basis for the proposed research.

Chapter 3: In this chapter, we give an overview of the technological stack used in this thesis. It explains all the hardware components and sensors, which include the Franka Emika Panda robotic arm, HoloLens 2, and computing devices. The software configuration part is also explained in this chapter, mentioning the Unity3D, ROS and MoveIt tools for motion planning and the ROS-TCP and UDP protocols for the communication between the different parts of the system.

Chapter 4: This chapter provides the methodology used for the development of AR-based robotic control system, helps to understand the overall system architecture, which encompasses the interplay of AR, robotic command functionality, and real-time feedback modalities. At the end of this chapter and beyond, the concepts related to designing virtual sliders, buttons, and elements in this exam, as well as the path planning built in MoveIt, will be mentioned. This realization will allow the development of an intuitive and efficient method to control the robotic arm in a dynamic context.

Chapter 5: In this chapter the results from user experiments used to validate the dissolve effectiveness and the usability of the developed AR-based control system are presented. Ten participants were recruited to complete a pick-and-place/tactile placement task (placing a sponge at 3 locations). Following task completion, participants filled out the System Usability Scale (SUS) and NASA-TLX surveys, using Google Forms. Finally, this chapter discusses the experimental outcomes, focusing on the time of task completion, accuracy, recorded feedback, and system weaknesses.

Chapter 2

State of the Art

This section summarizes the current state of the art in regard to Human-Robot Interaction (HRI) methods, specifically directed towards manipulators and the combination of Augmented Reality (AR) in industrial applications. It also presents several contributions such as collaborative robotics, teleoperation, interaction techniques, during which AR is highlighted as an effective tool for HRI by supporting intuitive interfaces over tasks, e.g., pick-and-place type of operations. Thanks to AR, situational awareness and decision making are enhanced, leading to increased productivity and safety. Nonetheless, yet HRI systems are expected to be fully effective in industrial environments, problems similar to intuitiveness of interfaces, real-time feedback and barriers of communication must be solved.

2.1 HRI Strategies

2.1.1 Colabortive Robotics

The principles of Human Robot Collaboration (HRC) have found applications in different ways, considering varying levels of engagement of human operator and robot. In particular, a detailed taxonomy was proposed in (3), where HRC was classified in eleven categories, including task type, robot morphology, interaction roles, time, and space. However, this might sound inappropriate for the time being given the most recent advances in the state of the art.

More recently, the distinction between safety, coexistence and collaboration between human and robot has been pointed out in (4). According to their framework, HRC spans from sharing only the physical workspace, but not the task, to shar- ing tasks, with cognitive engagement. In any condition, a safe behaviour must be inherently guaranteed and

accomplished. Thus, they have proposed a nested framework consisting of three levels of interaction between a human and a robot, where any greater engagement requires that the features of lower levels of interaction are guaranteed, as summarized in Fig. 2.1. Specifically, to achieve safety in a scenario of HRC, where cages and barriers are inappropriate, several internal and external mechanical, sensory and control safety features can be merged. In this regard, in general collisions should be prevented, but if they accidentally occur, the robot should be able to react reducing forces at the impact, by using appropriate control laws (5) or using lightweight robots with compliant joints (6). A further step into HRC according to (4) can be achieved by implementing coexistence.

This approach considers that a robot and a human operator safely share the workspace and might also work on the same object, but without any mutual contact or coordination of actions and intentions. Beyond coexistence, collaboration approaches allow the robot and the human operator to perform a complex task together, that is with direct interaction and coordination (4).

This can be achieved intentionally establishing a physical contact with exchanges of forces between the two agents, or without contact, for example by the use of gesture or voice commands. Within the premises of such a framework, a control architecture that integrates collision avoidance, detection, and reaction capabilities, as well as collaboration between a human and a robot, has been proposed in (4).

This distinction somehow recalls the one in (7) where HRC is differentiated from human–robot interaction (HRI) based on the principle that in HRC the human and the robot work together aiming at reaching a common goal. On the other hand, in HRI they interact not necessarily with a common goal, thus falling in the definition of coexistence of (4). Also in (8) the distinction is between safe coexistence, which pertains to safe (physical) HRI, and collaboration. In this context, a main challenge is to distinguish between accidental collisions and intentional contacts, which are associated to the human intention to start a physical collaboration phase(8).

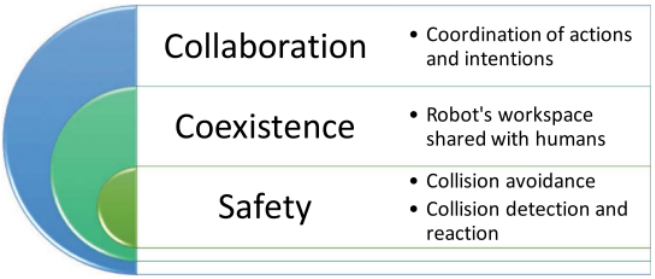


Figure 2.1 Nested levels for HRC, as proposed in the framework (1)

2.1.2 Teleoperation

Teleoperation refers to the remote control of robots or manipulators, allowing users to operate them from a distance. This technology is widely used in various fields, such as industrial automation, healthcare, and exploration, where direct human presence is difficult or dangerous. Teleoperation systems often include interfaces that provide real-time feedback, such as video feeds or sensor data, to help users control robots effectively. Additionally, advanced techniques like augmented reality (AR) can enhance teleoperation by overlaying helpful information, such as safe zones or robot trajectories, onto the user's view. These systems aim to make remote control intuitive, precise, and safe. Further details about teleoperation, including its design and applications, will be discussed in Section 2.2.

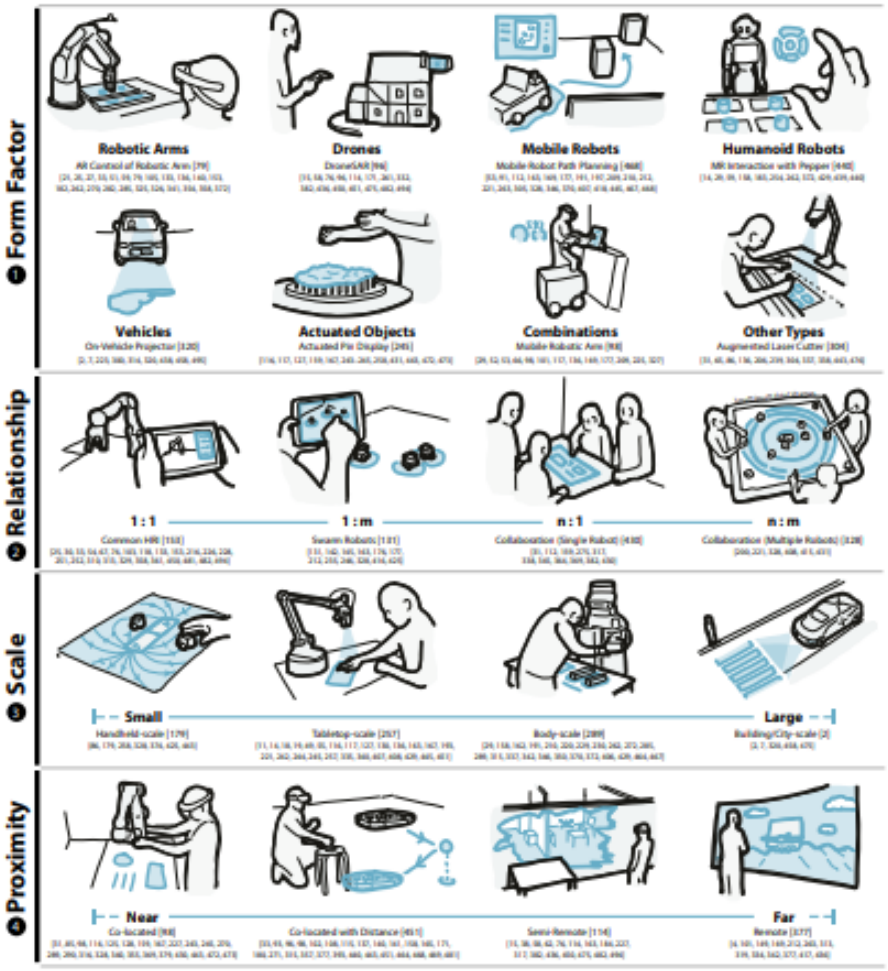


Figure 2.2 Characteristics of Augmented Robots
(2)

2.1.3 Interactions

In this section, interactions in AR and robotics research are analyzed, categorized along two key dimensions: the level of interactivity, which defines how users interact with AR systems and robots (Figure 2.3), and interaction modalities, which describe the different methods through which users engage with the system (Figure 2.4).

Level of Interactivity

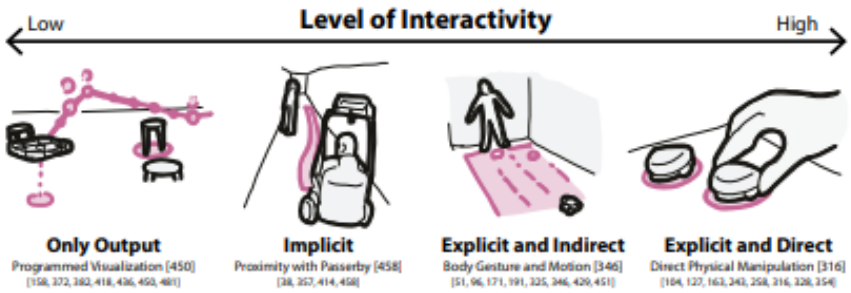


Figure 2.3 Level of interactivity (2)

No interaction (only output) occurs when AR is used solely for visual representation without user input, such as visualizing a robot's motion or capabilities (9). Implicit interaction, on the other hand, relies on user motion, such as position or proximity, as input (10). In this case, the system detects user behavior and responds accordingly, even if the user is not consciously aware of the interaction, as in scenarios where a robot adjusts behavior when someone approaches.

Explicit interaction can be divided into two main types. Indirect manipulation occurs remotely, without physical contact, using pointing gestures (11), object selection, drawing, or virtual menu navigation (12). Direct physical manipulation involves tangible engagement with the robot through actions such as grasping (13), object deformation (14), or direct touch inputs (15).

Interaction Modalities

Tangible interaction allows users to manipulate robots by physically deforming objects (14; 16), moving tangible components (13; 17), or directly handling the robot (18). Touch interaction typically involves touchscreen-based engagement, using gestures such as dragging (19; 20), pointing (13), and virtual menu navigation (21), allowing for precise input (22; 23).

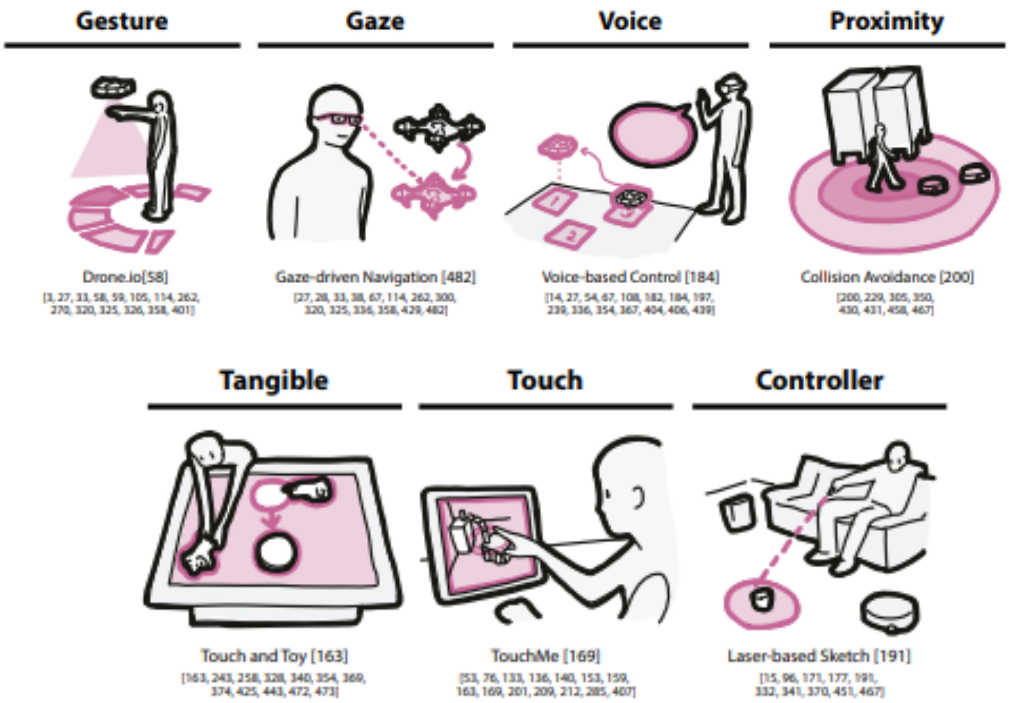


Figure 2.4 Interaction modalities and techniques (2)

Pointer and controller interaction enables users to manipulate robots through external controllers, often incorporating tactile feedback (24). This method also facilitates implicit interaction, such as designing virtual objects (25). Spatial gesture interaction is commonly used with Head-Mounted Displays (HMDs), allowing users to control robots through waypoint adjustments (11; 26) or hand gestures (12; 27).

Gaze-based interaction relies on eye movements for robot control, sometimes combined with gestures or functioning independently (28). Voice interaction allows hands-free control using spoken commands, particularly in co-located environments (29). Finally, proximity-based interaction utilizes user position as an implicit input method (30), dynamically updating AR trajectories when a user approaches the robot (10) or adapting content based on spatial proximity (31).

2.2 Teleoperation

Teleoperation represents a critical domain within Human-Robot Interaction (HRI), focusing on the remote control and supervision of robots by human operators. Unlike collaborative robotics, where humans and robots share physical or cognitive tasks in close proximity, teleoperation typically involves spatial separation between the operator and the robot. This separation necessitates robust communication channels, real-time feedback mechanisms, and intuitive interfaces to ensure effective task execution (32).

The evolution of teleoperation systems has been driven by advancements in robotics, augmented reality (AR), and communication technologies. Early teleoperation systems were primarily designed for hazardous environments, such as nuclear facilities and space exploration, where direct human presence was impractical or dangerous (32). Modern teleoperation systems, however, leverage sophisticated sensory feedback, AR overlays, and haptic interfaces to enhance operator situational awareness and control precision (2).

2.2.1 Contributions

Existing research has explored AR in robotics from various perspectives. Prior reviews have focused on specific application areas, such as HRI collaboration (33) and robotic surgery (34). This paper extends previous work by systematically reviewing literature in Human-Computer Interaction (HCI) and Human-Robot Interaction (HRI), particularly regarding robotic, actuated, and shape-changing interfaces. While AR's role in HCI is expanding (35), its integration with robotics remains underexplored.

Beyond reviewing existing work, this study highlights open research questions and opportunities for future investigation. Additionally, it introduces an interactive website that compiles a comprehensive research corpus. Inspired by similar efforts in personal fabrication (36; 37), data physicalization (3; 38), and material-based shape-changing interactions (29; 39), this resource offers an up-to-date and extensible platform for researchers and practitioners.

2.2.2 Approaches

Augmented reality in robotics is categorized along two dimensions: placement of AR hardware and the target location of visual augmentation. Hardware placement is classified into three categories: on-body (e.g., HMDs, mobile AR interfaces), on-environment (e.g., projectors, see-through displays), and on-robot (e.g., robot-attached projectors or displays). The target location of augmentation is categorized as either augmenting robots (overlaying

information on robots) or augmenting surroundings (enhancing the environment around robots).

On-Body Augmentation

On-body augmentation leverages AR devices worn or carried by users. HMDs like VRoom (40) overlay remote users onto telepresence robots, while mobile AR systems such as Young et al. (41) add expressive animated faces to robotic platforms.

On-Environment Augmentation

In environment-anchored augmentation, AR interfaces are embedded within the surroundings. Projection mapping techniques, as seen in DroneSAR (42), enhance drone capabilities, while see-through displays augment shape-changing walls and handheld interfaces (31; 43).

On-Robot Augmentation

Attaching AR devices directly to robots facilitates in-situ visualization. Furhat (44) employs a back-projected head for expressive interaction, while robot-attached displays enable robots to function as dynamic information screens (45).

2.2.3 Interaction Modalities in Teleoperation

A critical aspect of teleoperation is the design of interaction modalities. These include traditional methods such as joysticks and keyboards, as well as emerging techniques like gesture-based controls, voice commands, and haptic feedback devices. Recent studies have highlighted the potential of AR and VR interfaces in providing immersive teleoperation experiences. For example, AR overlays can project virtual waypoints or instructions onto the operator's field of view, enabling intuitive robot navigation and manipulation (11). Similarly, VR-based teleoperation systems allow operators to immerse themselves in the robot's environment, enhancing spatial understanding and task performance (12).

On-body augmentation leverages AR devices worn or carried by users. HMDs like VRoom (40) overlay remote users onto telepresence robots, while mobile AR systems such as Young et al. (41) add expressive animated faces to robotic platforms. On-environment augmentation integrates AR into the surroundings, with projection mapping highlighting objects or regions (46), and surface displays providing additional information for manipulation (13). Meanwhile, on-robot augmentation allows robots to modify their own appearance

or surroundings to enhance interaction. Robot-attached projectors display information on the robot's body or environment (47)(48), while robot-attached displays present interactive content directly on the robot (49)(45).

2.2.4 Methodology

This section outlines the methodology used to collect and analyze a representative set of AR and robotics papers (39).

Dataset and Inclusion Criteria

A systematic search was conducted across ACM Digital Library, IEEE Xplore, MDPI, Springer, and Elsevier. The search terms combined "augmented reality" AND "robot" in the title and/or author keywords since 2000, along with synonyms such as "mixed reality" (MR), "AR," "robotic," "actuated," and "shape-changing." This process initially identified 925 papers after removing duplicates.

Each paper was reviewed independently by four authors to exclude those focusing solely on AR-based tracking or conceptual papers. This led to 396 relevant papers. Additionally, 64 papers were included based on the authors' expertise in HCI, HRI, and robotic interfaces, resulting in a final corpus of 460 papers. While not exhaustive, this dataset serves as a representative subset of the field. Given the subjectivity of inclusion criteria, the dataset and coding are made open-source for future refinement and expansion (2).

Analysis and Synthesis

The dataset underwent a structured multi-step analysis. Open-coding was first conducted on a small subset to establish initial categories within the design space. This classification was iteratively refined through discussions among all authors to ensure consistency. Systematic coding followed, with three co-authors independently tagging the complete dataset. Finally, discrepancies were resolved to obtain the final coding results.

To enhance accessibility, results are presented using color-coded text and figures, with key citations embedded within figures for quick reference. Additionally, the appendix includes tables compiling all citations, categorizing papers by design space dimensions, facilitating easy retrieval of relevant works, such as those using AR for improving safety with robots, augmenting surroundings, or visualizing robot trajectories.

2.2.5 Purposes and Benefits

Visual augmentation enhances human-robot interaction, primarily aiding programming and control, as well as improving interpretation and communication. AR-based programming tools provide intuitive methods for simulating programmed behaviors. For example, GhostAR (50) visualizes robotic trajectories to aid industrial automation, while direct-mapping interfaces align AR overlays with real-world contexts.

Improved Safety

AR enhances teleoperation safety by visualizing safe zones and virtual barriers that help prevent collisions (51)(52). For instance, AR overlays can highlight obstacles in the robot's path, enabling operators to adjust trajectories in real-time.

Enhanced Communication

AR improves communication between operators and robots by conveying intent through spatial information. For example, motion and behavior plans can be visualized in the operator's field of view, providing clear instructions for task execution (9)(53).

Increased Expressiveness

AR enables robots to communicate more expressively by adding virtual arms, facial expressions, and other elements that improve user interaction (54)(41). This is particularly useful in scenarios where robots interact with humans in social or collaborative settings.

2.2.6 Applications of Teleoperation

Teleoperation has found applications in diverse fields, including healthcare, disaster response, and space exploration. In healthcare, teleoperated robotic systems enable surgeons to perform minimally invasive procedures with precision and accuracy (34). For example, the da Vinci Surgical System uses teleoperation to provide surgeons with enhanced dexterity and visualization during complex surgeries (55). In disaster response, teleoperated robots are deployed to assess hazardous environments, such as collapsed buildings or nuclear reactors, minimizing human risk (56). Space exploration relies heavily on teleoperation, with robotic systems like NASA's Mars rovers being controlled remotely from Earth (57).

2.2.7 Challenges in Teleoperation

Despite these advancements, several challenges remain in teleoperation research and application. Latency in communication channels can degrade system performance, particularly in applications requiring real-time responsiveness, such as remote surgery or disaster response (5). Additionally, the lack of tactile feedback in many teleoperation systems can hinder fine-grained manipulation tasks. Researchers are addressing these issues through innovations in haptic interfaces, predictive modeling, and AI-driven assistance systems (8).

Another key challenge is the trade-off between autonomy and human control. Fully autonomous systems may struggle with complex or unstructured environments, while fully manual systems can overwhelm operators with excessive workload. To address this, semi-autonomous teleoperation systems have emerged, where robots perform low-level tasks autonomously while relying on human operators for high-level decision-making (58). For example, AR-enhanced teleoperation allows operators to visualize robot actions and environmental data in real-time, improving task efficiency and reducing cognitive load (2).

2.2.8 Future Directions in Teleoperation Research

Future research in teleoperation is likely to focus on improving latency resilience, enhancing sensory feedback, and developing adaptive interfaces that cater to diverse user needs and application contexts. Innovations in haptic interfaces, for instance, could enable operators to "feel" the environment through force feedback, improving task precision and safety (8). AI-driven assistance systems could further enhance teleoperation by predicting operator intent and automating repetitive tasks (58). Additionally, the integration of AR and VR technologies could lead to more immersive and intuitive teleoperation experiences, bridging the gap between humans and robots in remote scenarios.

2.3 Interaction with Manipulators-Based AR

This section explores how Augmented Reality (AR) enhances interaction in manipulation-based robotics, focusing on design strategies, information presentation, and user interaction techniques.

Design Components for Manipulation

AR interfaces for manipulation-based robotics rely on specific design components to facilitate interaction. UIs and widgets, for instance, enable users to interact with robots through various elements. Menus allow for option selection and gestural control (59), while information panels display robot status, such as altitude or task progress (42). Controls and handles facilitate manipulation through virtual handles or adjustable control values (22)(17). Monitors and displays provide real-time feedback, including camera feeds and 3D reconstructions, aiding manipulation tasks (60).

Spatial references and visualizations further enhance manipulation by overlaying data onto the physical environment. Points and locations highlight landmarks or target areas (23)(61)(62), while paths and trajectories illustrate expected robot behaviors (52)(19). Additionally, areas and boundaries define safe zones or regions of interest (63)(52).

Embedded visual effects contribute to the expressiveness of AR-enhanced manipulation. Anthropomorphic effects render human-inspired graphics, such as arms or facial expressions, to improve interaction (54). Virtual replicas display 3D renderings of objects or environments, simulating manipulation behaviors (19), while texture mapping overlays interactive content onto physical objects to enhance interfaces (28).

Information Presentation for Manipulation

AR interfaces play a crucial role in presenting information essential for manipulation tasks. Robot internal information includes its status and condition, whether for emotional interaction (54)(41) or hardware maintenance (64)(65). Functionality and capability indicators, such as reachable regions and safe zones, support effective control (51).

External environmental data is equally significant. Sensor data, including camera feeds and 3D reconstructions, aids remote manipulation (24)(66). Obstacle visualization helps highlight objects or boundaries for safer interaction (67). Future robot behaviors are also conveyed through AR, with motion and behavior plans depicting trajectories (9)(68) and task progress indicators providing real-time updates (69)(61).

Interaction Techniques for Manipulation

AR enhances interaction techniques in manipulation-based robotics through on-body, on-environment, and on-robot augmentations. On-body augmentation involves AR devices like HMDs and mobile interfaces, enabling overlay of remote users or trajectory visualization (70)(9). Mobile AR further enhances expressiveness by adding animated faces to robots (41).

On-environment augmentation integrates AR into the surroundings, with projection mapping highlighting objects or regions (46), and surface displays providing additional information for manipulation (13). Meanwhile, on-robot augmentation allows robots to modify their own appearance or surroundings to enhance interaction. Robot-attached projectors display information on the robot's body or environment (47)(48), while robot-attached displays present interactive content directly on the robot (49)(45).

Benefits of AR for Manipulation

The application of AR in manipulation-based robotics brings numerous benefits. Improved safety is achieved through the visualization of safe zones and virtual barriers that help prevent collisions (51)(52). Enhanced communication results from the ability to convey robot intent via spatial information (9)(53). Finally, increased expressiveness allows for the addition of virtual arms, facial expressions, and other elements that improve user interaction (54)(41).

Chapter 3

Hardware and Software Configuration and Utilization

This section details the elements of hardware and software employed for this project together with their characteristics. The main objective is to present the method of connecting the Hololens 2 device with the Franka Panda robot to build an efficient system. This might help others understand the goal of each component and the role it serves to facilitate recreating the project in the future.

The project relies on two essential hardware units. The first one is the Hololens2 that let the user interacts with virtual objects which combine real-world and virtual elements. The device enables users to track hand movements while mapping spatial areas surrounding them and adds digital information onto actual physical elements. The Franka Panda robot serves as a precise robotic arm along with high flexibility which provides safe operation with human workers.

From the software implementation side various tools were used. The Mixed Reality Toolkit from Microsoft serves as a tool for developing interactive AR applications for Hololens implementation. The robot gets its control functions from MoveIt which provides strong capabilities for robot movement planning and self-collision avoidance. ROS-TCP enables the exchange of data between the interface and ROS. A fast communication through UDP enables quick command transmission as well as immediate robot response feedback. The next section helps to understand how the system operates, here we dive to each component characteristics.

3.1 Hardware Component

3.1.1 HoloLens 2

The Hololens2 is an AR headset device developed by Microsoft in 2019. It was created as an augmented reality solution that helps developers build various projects. Hololens2 two comes with 52° of vision with resoltion of 2K per eye, supporting hand and eye tracking thanks to its eye-tracking cameras and depth sensor. It supports also Voice Commands recognition for hands-free control.

One of the most important technology that comes with Hololens2 is the Spatial mapping that played the main role calculating the the cartesian coordination point in space, which served to end effector movement to the desired target. Different programming languages such as C#, C+, and Python can be used for development giving developers several options. Thanks to Holographic Remote Player the project could be executed and tested directly in real-time without the need to export the project to the headset each time a single edit is made. The detailed characteristics figure could be found in the next page.

Figure 3.1 HoloLens 2

Hololens2 in this project helps to see the virtual environment strating from the interface to the point placed in space represting the target point to the virtual Franka Panda Robot that is perfectly aligned on the real one. The headset is used to visualize the Franka Panda robot's state and planned motion, allowing users to see real-time updates on its position and trajectory. It enables intuitive control by sending movement commands through hand gestures. Additionally, it receives real-time feedback on the robot's status, ensuring smooth operation and immediate response to user inputs. The following figure contains a several characteristics details of the Hololens2 that I was taken from Microsoft's website (71):

Technical Specifications

Display	Optics	See-through holographic lenses (waveguides)
	Resolution	2k 3:2 light engines
	Holographic Density	>2.5k radiants (light points per radian)
	Eye-based Rendering	Display optimization for 3D eye position
Sensors & Audio	Depth	Azure Kinect sensor
	IMU	Accelerometer, Gyroscope, Magnetometer
	Camera	8MP stills, 1080p30 video
	Microphone Array	5 channels
	Speakers	Built-in, Spatial Audio
Human Understanding	Hand Tracking	Two-handed fully articulated model, direct manipulation
	Eye Tracking	Real-time tracking
	Voice	Command and control on-device, Natural Language with internet connectivity
Environmental Understanding	6DoF Tracking	World-scale positional tracking
	Spatial Mapping	Real-time environment mesh
	Mixed Reality Capture	Mixed hologram and physical environment photos and videos
Compute & Connectivity	SoC	Qualcomm Snapdragon 850 Compute Platform
	HPU	2nd Generation Custom-built Holographic Processing Unit
	WiFi	802.11ac 2x2
	Bluetooth	5.0
	USB	USB Type-C
Power	Battery Life	2-3 hours of active use
	Charging	USB Power Delivery – Fast Charging
	Thermals	Passively Cooled
Fit	Single size	Fits over glasses, size using adjustment dial
Software	Windows Holographic OS	
	Edge	
	Remote Assist	
	Layout	
	Guides	
	3D Viewer	
	OneDrive for Business	

Figure 3.2 Technical Specification

3.1.2 Franka Panda Robot

The Franka Emika Panda is a robotic arm designed for research, industry, and education. It was chosen for this project due to the ability to support the inetgration with ROS, which is suitable for advanced control, simulation in Gazebo, and even the use of artificial intelligence to improve its movements, it also is known for being precise, flexible, and easy to use. Weighing about 26 kg, it has 7 joints (or degrees of freedom) that can lift up an object that weights up to 3kg, making it great for delicate and complex tasks. Thanks to its torque sensors, it can feel forces in real time and react accordingly, making it safe to use around people.

The Panda robot can be programmed in C++, Python, and MATLAB, giving various options in how it could be controlled. in our case we will be using C++ to control movement of real robot that mimics the virtual one, the next chapter will explain how this could be done. It comes with Franka Control Interface (FCI) and Franka Desk, which make it easy to operate, Where the option to lock, unlock the joints is available.

One of the advantages this robot serves is the "Reflexive Mode" where it gets activated when the robot hits an object or a certain wrong pressure applied on it, this ensures the safety of the users as well as the robot itself with the possibility of shutting down the robot using special switches. In this project, the Panda robot is used together with Hololens 2 to create an interactive control system, which later will be used to achieve pick-and-place tasks which the rest of the details could be found in chapter 5,

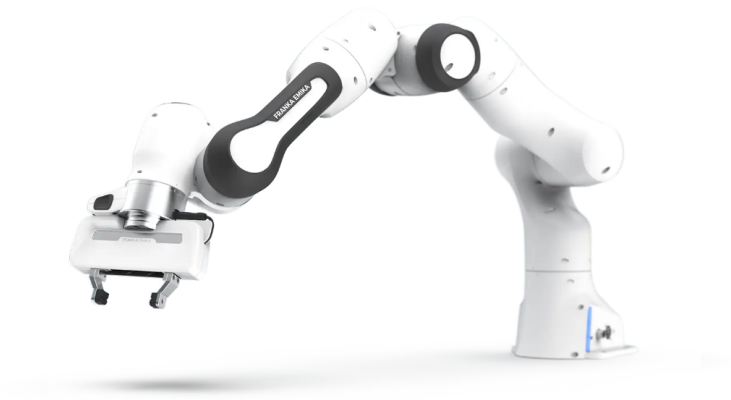


Figure 3.3 Franka Panda Robot

3.2 Software Tools

3.2.1 Unity3D

Unity 3D is a powerful development platform widely used for creating applications in gaming, simulations, and AR/VR. It supports over 25 platforms, including Windows, macOS, Android, ect, allowing developers to build and deploy projects. Unity uses C# as its main scripting language and provides real-time deploying with advanced visual effects. The Unity Editor offers tools for animation, physics, and lighting, making development more efficient.

Unity3d supports Microsoft Mixed Reality Toolkit which was the main key used thanks to its features that allowed designing the interface, adjust the sphere. It provides also essential tools for hand tracking, eye tracking, spatial mapping, and UI interactions in mixed reality applications. Another important feature is the Holographic Remote Player, which enables developers to test and run their AR projects in real-time without needing to deploy them manually to the headset every time, saving time in the development procedure.

Unity also played a key role in developing the AR interface for the Franka Panda robot, allowing users to visualize and interact with the robot's state and planned movements through the AR environment. Additionally, Unity supports ROS-TCP connection, which was crucial in this project, as it enabled real-time communication between Unity and ROS which allowed the transfer of different messages such as: (x y z) coordination, grasp release, move actions. The Articulation Body system in Unity was used to model and simulate the Panda robot's kinematics, ensuring accurate joint movement and realistic physics-based interactions.

These features made Unity a fundamental tool for integrating AR with robotics, providing an convenient experience for a realistic robotic control and visualization.

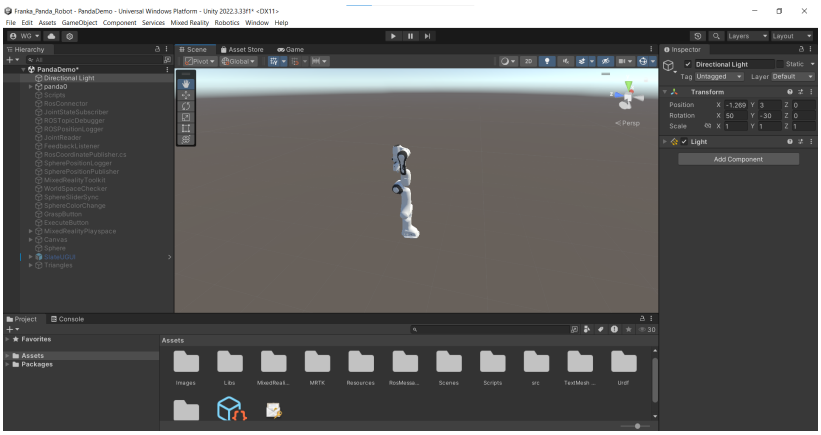


Figure 3.4 Unity3D footage

Here we will break down the most important libraries that were used, their functionalities and a certain role example they played in the structure:

Library	**Functionality**	**Usage Example**
UnityEngine, TMPro, UnityEngine.UI	UnityEngine provides essential game object handling, TMPro enables advanced text input fields, and UnityEngine.UI manages UI elements for coordinates and button positions.	Handling user button actions
Unity.Robotics.ROS TCPConnector, RosMessageTypes.Std	Unity.Robotics.ROSTCPConnector enables communication between Unity and ROS, allowing real-time message exchange, while RosMessageTypes.Std provides standard message types like 'BoolMsg' for signaling execution.	Sending execution signals to ROS
Microsoft.MixedReality .Toolkit.Input, Microsoft.MixedReality .Toolkit.UI	MRTK provides support for hand-tracking, gestures, and UI interactions in mixed reality applications. It enables pointer-based interactions for touch and click events.	Changing object color on touch interaction in Mixed Reality
RosMessageTypes.Geometry	Provides message types such as 'PointMsg', which is used to represent 3D coordinates in ROS.	Publishing the sphere's position in Unity relative to the Panda robot's base to ROS
System.Collections.Generic	System.Collections.Generic enables data structures like lists and enumerations for managing joint states.	Controlling and rotating robot joints using inverse kinematics in Unity

Table 3.1 Libraries used in the project

Franka Panda Robot Unity Model:

Articulation Body is a special type of physics component in Unity that helps simulate realistic movement for the connected parts. It is used to create joints that move and respond to forces like gravity. In this work, the Articulation Body was applied to the Franka Panda model to make it move in a way that feels real. This helped the virtual robot in HoloLens 2 behave like the actual robot, following the correct motion and physics. The specific configuration is listed in the table below:

Feature	Value / Description
Use of gravity	Activated
Automatic center of mass	Activated
Automatic Tensor	Activated
Linear Damping	0.05
Angular Damping	0.1
Joint Friction	0.2
Collision detection	Discreate
Force limit	3.402e+10

Table 3.2 Articulation body settings

3.2.2 MRTK3

The Microsoft Mixed Reality Toolkit (MRTK) is an open-source toolkit that helps developers create Mixed Reality (MR) applications more easily. It works with devices like Microsoft HoloLens, Windows Mixed Reality headsets, and other XR devices. MRTK provides many tools.

MRTK has several important features. It supports different devices, includes tools for hand tracking, eye tracking, and gestures, and has a user-friendly UI system with buttons, sliders, and menus designed for MR, the good part they are prebuilt in the system which made the process of finishing the interface faster. It also has spatial awareness, allowing digital objects to fit naturally into the real world. The input system supports hand gestures, voice commands, and controllers, because of its modular design, developers can customize and extend it as needed.

MRTK also includes spatial awareness, meaning it understands the real-world environment. It can scan surfaces, handle object occlusion, and adjust interactions based on room size. It is especially optimized for Microsoft HoloLens, making it the best choice for developing MR applications on this device. The following table represents the configuration used in the system (72).

Configuration Option	Value / Description
Platform	Universal Windows Platform
SDK	10.0.22621.1778
Target SDK version	3D3 Project
Architecture	AMD 64-bit
Hand tracking	Activated
Gaze tracking	Activated
Motion Controller Model	Activated
Sptial awarness	Disactivated*
Background	Transparent

Table 3.3 MRTK3 Configuration

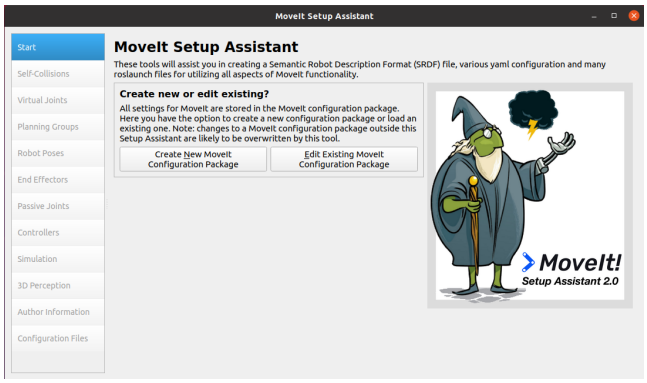
Hint*: The spatial awareness in this project needed to be deactivated, the virtual hologram needed to be precisely aligned with the real Robot, so it's meant to prevent affecting the virtual world in any wrong way.

3.2.3 MoveIt

MoveIt is an advanced motion planning framework for controlling robotic manipulators in ROS. Used to provide tools for motion planning, kinematics, collision detection, and trajectory execution, is one of the components that are critical when developing robotic arm applications that played the main role in motion planning execution in our developed structure. (73).

For handling kinematics, the system uses both forward kinematics (FK) and inverse kinematics (IK) solvers. To solve IK problems, it relies on tools like KDL, TracIK, or IKFast. These are written in C++. The choice of solver depends on the robot and task KDL is slower but flexible, while IKFast is faster if you predefine the robot model, the configuration used in the developed project will be detailed later.

The Self-Collision avoidance is built using the Flexible Collision Library (FCL) which is prebuilt in the program, it is useful to avoid such points where the robot can harm itself. In order to have Franka Panda Robot simulated on MoveIt, setting up the configuration is required using MoveIt assistant.



Self-Collison part is loaded automatically once loading the .xacro file of the Franka Panda Robot found on the official Franka Emika github (74) The rest of the MoveIt configuration is summarized in the following table:

Configuration Option	**Value / Description**
Virtual Joint	Child Link: panda_link0 | Parent Frame: world | Type: Fixed
Planning Groups	panda_arm: [Panda_joint1 ... Panda_joint7] - Revolute panda_hand: [Panda_finger_joint1Panda_finger_joint2] - Prismatic
End Effectors	Parent Link: panda_link7 "Imaginary link"*
Passive Joints	All Joints
Controllers	All Joints

Table 3.4 MoveIt Assitant Settings

Hint*: The reason for setting up the imaginary link is to define the end effector as a point located between the two fingers of the gripper, which in this case is a fixed link.

Parameter	**Value**
Kinematic Solver	`kdl_kinematics_plugin/KDLKinematicsPlugin`
Kin. Search Resolution	0.005
Kin. Search Timeout (sec)	0.005
Goal Joint Tolerance (m|rad)	0.0001
Goal Position Tolerance (m)	0.0001
Goal Orientation Tolerance (rad)	0.0010

Table 3.5 Planning Group Parameters for 'panda_arm'

3.2.4 ROS-TCP Connection

Connection oriented TCP (transmission control protocol) connection based link communication link between ROS tokens and external instruments or processes. Transmission Control Protocol: It provides reliable, ordered, and error-checked delivery of a stream of data packets between applications running on hosts communicating over an IP network. TCP layer in robotics2, etc.) TCP layer in software systems and commercial systems (hardware) TCP layer in ICT TCP Operational error detection Applied to high-reliability delivery Single service type so far Provide feedback for the system. This, in turn, is typically used in applications where both systems have to guarantee that all data being transmitted is coherent and valid, like remote control of robotic arms (75).

3.2.5 UDP Connection

ROS UDP(User Datagram Protocol), because UDP is connection less protocol, so you can get faster communication, where reliability and error correction is not to be considered. It also works at the transport layer, so using UDP allows for a connectionless protocol, so it is useful for real-time applications where there is no need to establish a connection, like streaming sensor data or broadcasting messages. In ROS, UDP is used frequently with large amounts of data, such as in visual odometry, when speed matters most, even above the idea that every packet was sent (76).

Chapter 4

Methodology

This chapter describes in detail the methodology underlying the presented system for the control of the Franka Emika robotic arm using augmented reality. The system integrates AR and robotics to enhance the speed and accuracy of manipulator control.

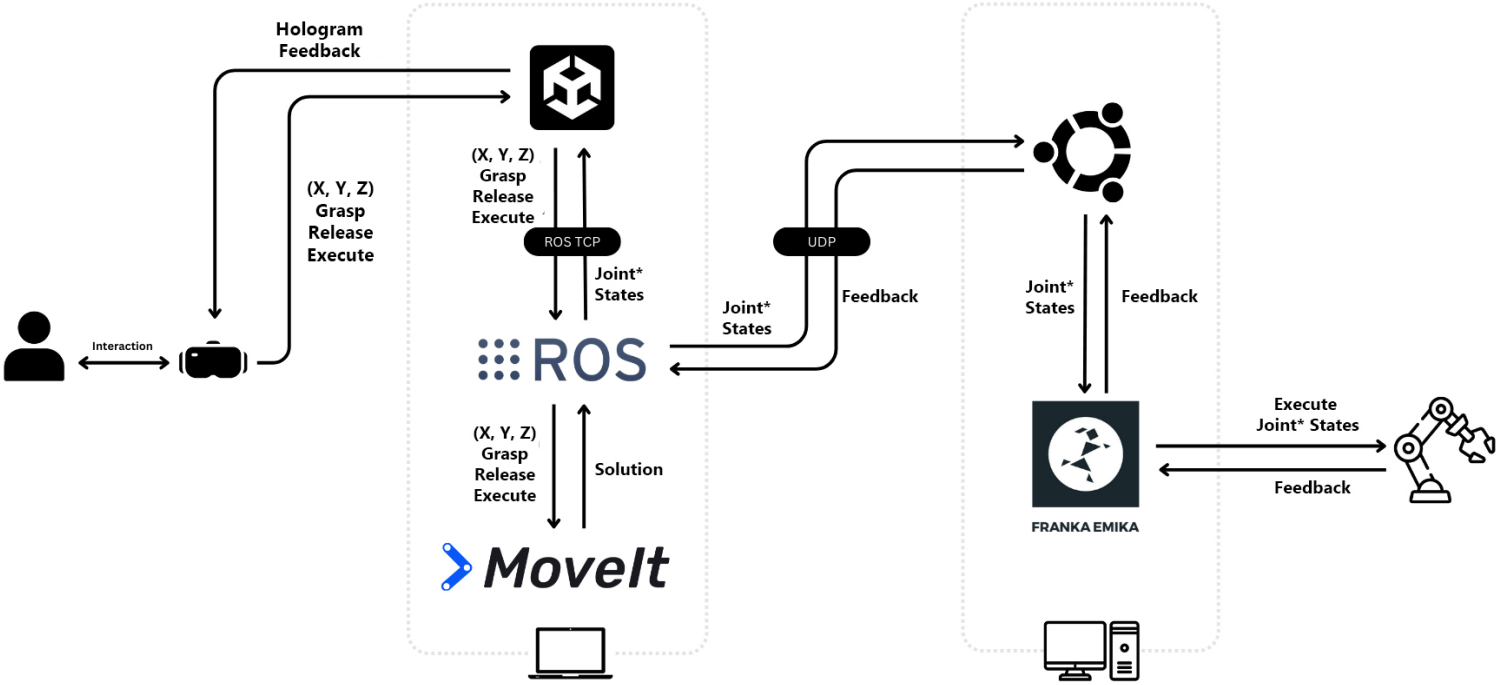


Figure 4.1 WorkFlow

The robotic arm is modeled in Unity as an articulation body since it provides a working mechanism for moving parts relative joint to joint. The articulation body determines the

movement and orientation of individual joints in the robot based on its kinematic constraints. Interaction data (coordinates and commands) is sent from HoloLens to Unity, which processes the inputs and connects to ROS through the ROS-TCP Connector. This protocol allows Unity to write or read from ROS, that is, Unity could send position and joint commands or commands related to individual grasps.

On the ROS side, the (X, Y, Z coordinates and command variables) received is used by a motion planning framework known as MoveIt. In this situation, MoveIt solves the inverse kinematics of the target pose to compute a collision-free trajectory. The solution is turned into joint states for the robot's joint positions, velocities, and efforts to reach the target pose. The joint states are subsequently passed along through the ROS-TCP interface back to Unity, allowing the holographic simulation to follow the planned motion of the real robot.

ROS also sends the joint states via UDP protocol to a separate computer that is connected to the real Franka Emika arm. UDPtransmission enables both real-time communication and low latency. The joint states are processed by the receiving computer, which then issues commands to the robot's controller to drive the motors for each joint in order to carry out the planned motion. The corresponding robot (physically, not as a hologram) will then be commanded to articulate to the location or grasp/release the object as requested.

At this point, as the robot performs this motion it reports its current joint states (e.g., position, velocity, effort) back upstream over the UDP connection to ROS. Classic Unity receives this feedback through ROS-TCP which communicates the holographic model updated in real time. This feedback loop allows the user to visualize the exact state and movement of the robot holographically on the HoloLens, keeping the virtual system and the physical systems in sync. Specifically, it incorporates Unity's articulation body system for rendering joint-based movement, ROS-TCP for sending positional information and directives, MoveIt for planning motions, and UDP for real-time interaction with the hardware robot. Our system includes the Holographic Remote a device that is responsible for streaming holograms over to the HoloLens and allows intuitive interaction and visualization of the robot's state and movements.

4.1 HoloLens2 & Unity

This is where we zoom into what the user actually experiences when interacting with the system. This refers to what the user will see, the arsenal of tools available, the working of the interface, and the prerequisites necessary to make this functionality a reality. Last but not least, when the user puts on the Hololens2, three main elements will appear in the augmented reality environment, the hologram, the interface and the sphere. In order to broadcast our Holographic space to the hololens2 an easy way is used which permits to play the unity scene directly in real time without the need to deploy each time. The tool is called "Holographic remote play mode". It needs just to insert the IP address shown in the **Holographic Remote** app on Hololens2 and unity as shown in the following figure (77):

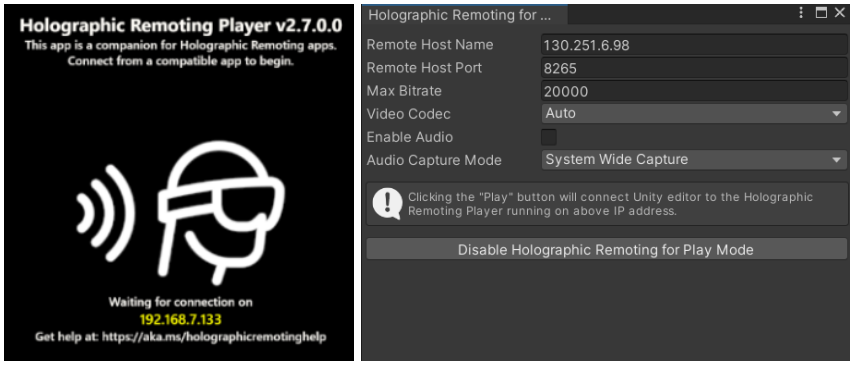


Figure 4.2 Holographic remote

The diagram 4.3 demonstrates the various functions being exchanged between Unity & ROS .

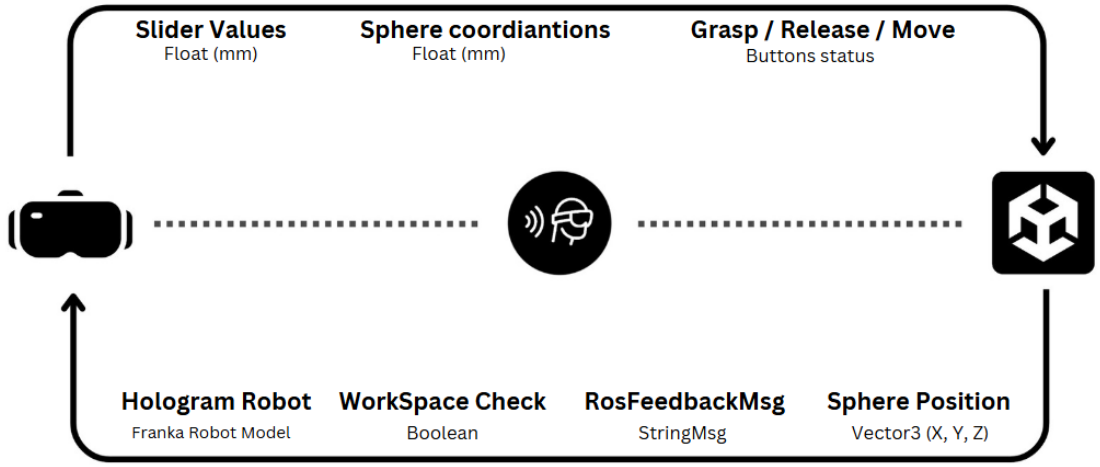


Figure 4.3 Hololens & Unity

4.1.1 The interface

The interface is the central element that enables the user to control the system effectively. It provides three key functionalities: moving the hologram, controlling the gripper (open/close), and executing the motion to the real robot. The interface is composed of three main components: action buttons, the slider, and the feedback box.

Action Buttons: The action buttons allow the user to execute specific commands directly through the Hololens2 interface. These buttons are responsible for:

- **Moving the hologram:** This button confirms the current sphere's position and sends the Cartesian coordinates (x, y, z) to control system of the robot where MoveIt is running for further processing. The system calculates the inverse kinematics (IK) and sends joint state solutions to the virtual robot.
- **Opening and closing the gripper:** The buttons send commands to open or close the robotic gripper, which is reflected in real-time feedback from the system.
- **Executing motion:** This will give the order to move the Real Robot

The Three Slider: The slider provides precise control over the Cartesian coordinates of the sphere, representing the target position the end effector aims to reach. Users can adjust the (x, y, z) coordinates using the slider, which synchronizes in real-time with the holographic sphere's position in the augmented reality space.

- **Bidirectional interaction:** When the user moves the slider, the sphere updates its position accordingly. Similarly, if the user directly moves the holographic sphere, the slider values change synchronously to reflect the updated coordinates.
- **Precision control:** The **+ -** allows precise adjustments of the slider which allow users to fine-tune the position of the target before sending the command to the robot. This ensures that the robot moves precisely to the desired location.

The Feedback Box: The feedback box is a gray panel located within the interface. Its primary role is to provide real-time updates to the user, ensuring that every action is tracked and confirmed.

- **Coordinate confirmation:** The feedback box displays whether the Cartesian coordinates have been successfully sent to ROS via the ROS TCP connection.
- **Gripper status:** The current state of the gripper (open or closed) is also shown here, providing clear feedback to the user.
- **Motion execution:** When the "Execute" button is pressed, the feedback box notifies the user if the motion has been successfully sent, along with a status update on the inverse kinematics solution calculated in MoveIt.

The interface structure was updated over time during the development process and while being tested by the pilot. It was modified until the size of different components became clear in terms of functionality and also usability, the size became suitable for use, and the overall structure achieved harmony which lead to the following final design:

Figure 4.4 The Interface

4.1.2 The sphere

The **Sphere** plays a central role in the holographic interface by acting as a visual and interactive representation of the target position the robot's end-effector needs to reach. It provides both intuitive feedback and precise control, allowing users to manipulate it seamlessly within the 3D holographic environment. The sphere's position is synchronized with the sliders on the interface, ensuring that any manual adjustment of its Cartesian coordinates using sliders instantly updates its position in the virtual space, and vice versa. This bidirectional synchronization is implemented using Unity's event-driven mechanism, where changes in either the sliders or the sphere trigger corresponding updates in real-time.

The sphere's position is calculated relative to the base of the Panda robot, known as 'panda_link0', ensuring consistency with the robot's frame of reference. Adjustments to its position are handled in Unity, where its coordinates are recalculated, transformed, and displayed in meters or centimeters. Additionally, users receive visual feedback through dynamically updated text, which follows the sphere's movement and shows its Cartesian coordinates for precise adjustments.

To enhance the sphere's interactivity, users can interact with it using MRTK controls on the Hololens2. For example, when the user touches the sphere, its color changes momentarily to red, providing tactile feedback. This is implemented by detecting pointer events and modifying the sphere's material color. The combination of these features ensures the sphere is both informative and engaging, bridging the gap between the virtual environment and the physical control of the robot.

This dynamic and interactive design of the sphere makes it a core component of the interface, enabling users to easily and intuitively set precise end-effector goals for the robot in augmented reality.

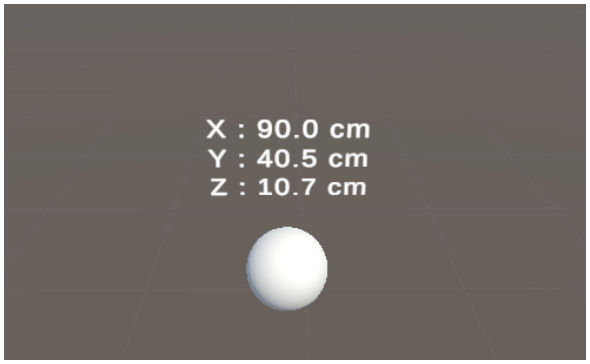


Figure 4.5 Sphere

4.1.3 The Robot Hologram

To support usability and help users have smoother interactions with the Virtual Franka Robot (78), the code implements a system to signal when the user places the sphere out of the workspace. Specifically, the sphere's position is checked against predefined workspace boundaries. If the sphere is placed outside the designated workspace, the virtual robot's appearance changes, and a warning panel is displayed.

The `WorkspaceChecker` script defines the workspace boundaries along the X, Y, and Z axes, and it continuously monitors the position of the sphere using the `SpherePositionHandler` script. The `CheckSpherePosition` method compares the sphere's current coordinates with the workspace boundaries. If the sphere is outside the workspace, an out-of-workspace warning panel `outOfWorkspacePanel` is shown, and the robot's appearance is modified to signal the user. This signaling effect is achieved by applying a pulsing red color to the robot's materials using the `PulseEffect` coroutine. The robot's color returns to its original state when the sphere is relocated within the workspace.

In addition to the visual signaling, the panel automatically closes after 10 seconds through the `AutoClosePanelAfterDelay` coroutine. If the sphere is brought back inside the workspace, the robot stops pulsing, the warning panel is hidden, and the color resets to its original state.

Overall, this interaction system guides the user, providing visual cues about where the end effector should be positioned to be within reach, helping to improve the experience and usability of the virtual robot(78).

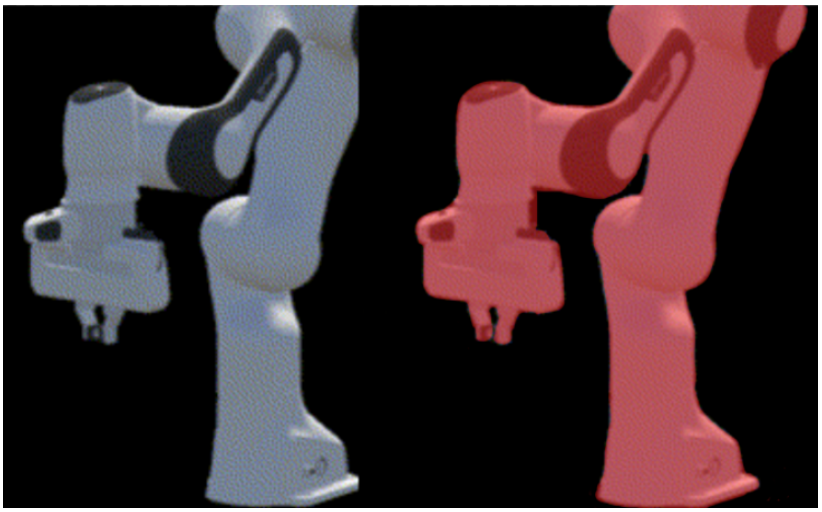


Figure 4.6 Before & After exiting Workspace

4.2 Unity & ROS

4.2.1 ROS TCP

The **Unity-ROS Integration** is a crucial component of the system, enabling real-time communication and data exchange between the Unity holographic interface and the ROS, which was possible due to ROS-TCP connections. This setup ensures that Unity acts as a user-friendly front-end interface for robot control, while ROS handles the back-end computation, planning, and execution. The diagram 4.7 highlights the exchange of the components between Unity and ROS.

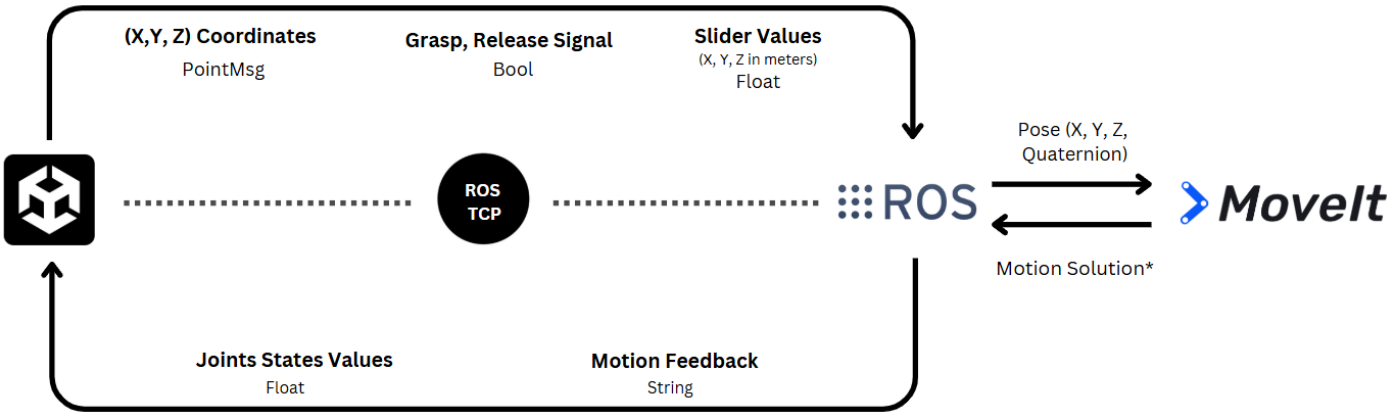


Figure 4.7 Unity & ROS

Unity's Role

Unity utilizes the **ROS TCP Connector**, a plugin that facilitates seamless communication with the ROS environment over TCP/IP. In the Unity project, the `RosConnector` script initializes and manages this connection.

1. ROSConnection Initialization: Unity creates a singleton instance of the `ROSConnection` class using `ROSConnection.GetOrCreateInstance()`. This instance acts as the bridge between Unity and ROS, enabling the two systems to send and receive messages.

2. Data Exchange: Unity uses the ROS TCP Connector to publish user-defined goals, like the sphere's position or other input parameters, to ROS topics. Similarly, it subscribes

to feedback topics from ROS to reflect the robot's state or task progress back in the Unity interface.

3. Modular Design:The design ensures flexibility and scalability, allowing new ROS topics, services, or actions to be added easily.

ROS's Role

In the ROS environment, the communication is set up using the `ros_tcp_endpoint` package, which acts as a server listening for Unity's messages. This is launched via the command:

```
roslaunch ros_tcp_endpoint endpoint.launch
```

1. **ROS TCP Endpoint**:
    - The `endpoint.launch` file starts the TCP server that receives data from Unity and routes it to the appropriate ROS topics, services, or actions. This bidirectional setup enables real-time feedback and control.
2. **Message Handling**:
    - Messages sent from Unity (e.g., target positions) are processed by ROS nodes responsible for planning or control.
    - Feedback, such as robot status or position updates, is sent back to Unity, ensuring synchronization between the holographic interface and the robot.

How It All Works Together

When the user interacts with the holographic interface in Unity, such as moving the sphere to define a target position, Unity publishes the target position to a specific ROS topic. ROS processes this target, plans the robot's motion, and executes it. The current robot state is then published back to a ROS topic, and Unity subscribes to this topic to provide visual and textual feedback to the user.

This integration of Unity and ROS creates a powerful and user-friendly system for controlling robots in real-time, combining the strengths of Unity's interactive 3D visualization with ROS's robust robotic framework.

4.2.2 Joint States subscriber

The Joint State Subscriber is an essential step in the Unity-ROS integration process, enabling Unity to receive and process joint state data from the ROS environment. This component facilitates real-time synchronization of robot joint positions, ensuring the Unity visualization accurately reflects the robot's physical state. In Unity, the `JointStateSubscriber` script handles the subscription to ROS joint state messages (`JointStateMsg`) and updates the corresponding `ArticulationBody` components.

The script defines an array of `ArticulationBody` objects, representing the robot's joints. The method `UpdateJointStates(JointStateMsg)` is triggered whenever a new joint state message is received. It performs the following tasks:

- Logs the number of received joints and compares it with the expected number of joints in the Unity model.
- Iterates through the received joint data, logging each joint's name and position.
- Updates the local position of each corresponding `ArticulationBody` based on the received joint positions.

Articulation Body Handling

To support joint movements, the system defines the `IKJoint` class, which encapsulates the properties and behavior of an individual joint. This class provides functionality to:

- Rotate joints based on user input or predefined goals.
- Set and adjust joint stiffness for control stability.
- Compute the current joint rotation and apply new target positions.

In the ROS environment, the joint state messages are published by relevant ROS nodes. Unity subscribes to these messages using the ROS TCP Connector, which ensures seamless communication over TCP/IP. The joint states are published in the standard ROS `JointState` message format, which includes:

- `name`: An array of joint names.
- `position`: An array of joint positions.

How It Works Together

When ROS publishes a new joint state message:

1. Unity's ROS TCP Connector receives the message.
2. The `UpdateJointStates` method processes the message, updating the Unity model's joints to reflect the new positions.
3. Any discrepancies in joint counts are logged, ensuring synchronization and error handling.

4.2.3 Feedback Sender & Receiver

This part provides detailed documentation for the **Feedback Sender Script**, a Python script designed to integrate a Franka Panda robotic arm with Unity via ROS. The script performs the following key tasks:

- Receives commands from Unity for Cartesian movement, grasping, and execution.
- Publishes feedback to Unity via the `/unity_feedback` topic.
- Controls the robotic arm and gripper using MoveIt.

The script interacts with the following ROS topics:

- `/unity_feedback`: Publishes feedback messages to Unity.
- `/unity_coordinates`: Subscribes to Cartesian coordinates for arm movement.
- `/unity_grasp_signal`: Subscribes to grasping commands (open/close gripper).
- `/unity_execute`: Subscribes to execution commands for the real robot.

The ROS node is initialized as `unity_feedback_publisher`.

Python Code

The Python feedback_sender.py script does as follows:
send_feedback: Publishes a message to the `/unity_feedback` topic.

- Input: String message.

- Example: `send_feedback("Goal reached successfully.")`

move_to_cartesian_point: Moves the robot to a specified Cartesian point using inverse kinematics (IK).

- Input: `Point` object with `x`, `y`, `z` coordinates.
- Feedback: Publishes success or failure to Unity.

move_fingers: Controls the gripper position.

- Input: Finger joint value (e.g., 0.035 m for open, 0.015m for closed).
- Feedback: Publishes `"Gripper opened"` or `"Grasping"`.

execute_real_robot_movement:Executes a planned trajectory on the real robot.

- Feedback: Publishes `"Moving real robot"` or `"Execution failed"`.

coordinate_callback: Handles incoming Cartesian coordinates from Unity and triggers `move_to_cartesian_point`.
grasp_signal_callback: Handles grasping commands (`True` for closing, `False` for opening) and triggers `move_fingers`.
execute_callback: Handles execution commands for the real robot and triggers real robot movement

4.2.4 ROS & MoveIt

This section provides a detailed explanation of the Python script used to integrate the **Franka Emika Panda robot** with ROS, MoveIt, and Unity. The script handles Cartesian movement, gripper actuation, and communication between Unity and the robot.

Imports

The script imports essential libraries for ROS, MoveIt, and transformations:

- **ROS Libraries:** `rospy` for ROS node initialization and communication, `geometry_msgs.msg` for handling Cartesian coordinates, and `std_msgs.msg` for Boolean and String messages.
- **MoveIt Libraries:** `moveit_commander` for motion planning and robot control.
- **Transformations:** `quaternion_from_euler` from `tf.transformations` to convert Euler angles to quaternions for orientation.

Global Variables

The script uses global variables to manage state and avoid redundant operations:

- `is_processing`: A Boolean flag to ensure the robot does not process multiple commands simultaneously.
- `last_coordinates`: Stores the last processed Cartesian coordinates to avoid duplicate processing.
- `cooldown_time`: A time delay (1.0 seconds) to prevent rapid retriggering of commands.
- `last_motion_time`: Tracks the timestamp of the last motion command.
- `finger_state`: Tracks the current state of the gripper (open or closed).
- `open_finger_value` and `close_finger_value`: Define the joint values for the gripper's open and closed states.

Functions

`move_to_cartesian_goal(x, y, z)`

This function moves the robot's end effector to the specified Cartesian coordinates:

- **Inputs:** `x`, `y`, and `z` coordinates received from Unity.
- **Process:**
  - Checks if the cooldown period is active using `last_motion_time` and `cooldown_time`.
  - Initializes MoveIt's `RobotCommander` and `MoveGroupCommander` for the `panda_arm` group.
  - Defines the target pose (`pose_goal`) using the input coordinates and a fixed downward orientation (quaternion derived from Euler angles).
  - Plans and executes the motion using `move_group.go()`.
  - Publishes feedback to Unity via the `unity_feedback` topic using `feedback_pub.publish()`.
  - Updates `last_motion_time` and resets `is_processing` after completion.

`move_fingers(position, feedback_msg)`

This function controls the gripper's fingers:

- **Inputs:** `position` (joint value for the fingers) and `feedback_msg` (status message to send to Unity).
- **Process:**
  - Initializes MoveIt's `MoveGroupCommander` for the `hand` group.
  - Sets the target joint values for the fingers and executes the motion.
  - Publishes feedback to Unity via `feedback_pub.publish()`.

`grasp_signal_callback(msg)`

This callback function handles gripper control signals from Unity:

- **Input:** `msg` (a `Bool` message from Unity indicating whether to open or close the gripper).
- **Process:**
  - Updates the global `finger_state` variable.
  - Calls `move_fingers()` with `close_finger_value` or `open_finger_value` based on the received signal.
  - Publishes feedback to Unity (e.g., "Gripper closed.").

`unity_coordinates_callback(data)`

This callback function processes Cartesian coordinates from Unity:

- **Input:** `data` (a `geometry_msgs.msg.Point` message containing `x`, `y`, and `z` coordinates).
- **Process:**
  - Compares the received coordinates with `last_coordinates` to avoid duplicate processing.
  - Checks if the robot is already processing a command using `is_processing`.

  - Updates `last_coordinates` and calls `move_to_cartesian_goal()` to move the robot to the new coordinates.
  - Publishes feedback to Unity (e.g., "Coordinates received...").

`listener()`

This function initializes the ROS node and sets up communication:

- Initializes the ROS node with `rospy.init_node()`.
- Sets up the publisher `feedback_pub` to send feedback messages to Unity via the `unity_feedback` topic.
- Subscribes to:
  - `unity_coordinates` (for Cartesian coordinates) using `geometry_msgs.msg.Point`.
  - `/grasp_signal` (for gripper control) using `Bool`.
- Starts the ROS event loop with `rospy.spin()`.

ROS Communication

The script uses ROS topics for communication between Unity and the robot:

- **Subscribers:**
  - `unity_coordinates`: Receives Cartesian coordinates (`x`, `y`, `z`) from Unity.
  - `/grasp_signal`: Receives Boolean signals to control the gripper.
- **Publisher:**
  - `unity_feedback`: Sends status updates (e.g., "IK solving...", "Goal reached successfully.") to Unity.

Example Interactions

- **Unity Sends Cartesian Coordinates:**
  - Unity publishes a `geometry_msgs.msg.Point` message to the `unity_coordinates` topic.

  - The `unity_coordinates_callback()` function processes the message, checks for duplicates, and calls `move_to_cartesian_goal()`.
  - The robot moves to the target coordinates, and feedback is sent to Unity via `unity_feedback`.

- **Unity Sends Gripper Signal:**
  - Unity publishes a `Bool` message to the `/grasp_signal` topic.
  - The `grasp_signal_callback()` function processes the message and calls `move_fingers()` to open or close the gripper.
  - Feedback (e.g., "Gripper closed.") is sent to Unity via `unity_feedback`.

Key Variables and Messages

- **Variables:**
  - `is_processing`: Ensures only one command is processed at a time.
  - `last_coordinates`: Stores the last processed coordinates to avoid duplicates.
  - `finger_state`: Tracks the gripper's current state.
  - `open_finger_value` and `close_finger_value`: Define joint values for the gripper.
- **Messages:**
  - `geometry_msgs.msg.Point`: Used for Cartesian coordinates.
  - `Bool`: Used for gripper control signals.
  - `String`: Used for feedback messages to Unity.

4.3 ROS & Franka Robot

4.3.1 UDP Sender

The provided code implements a ROS node designed to send joint state positions of a manipulator to a remote server via UDP, enabling the movement of the Real Panda Robot. It encapsulates UDP communication through the `UDPSenderSocket` class, which handles socket configuration and data transmission. The `Configure` method initializes the

UDP socket, sets the destination IP address and port, and includes error handling. The `Send` method transmits the robot's joint state positions to the remote server. Several global variables are defined, including `current_positions` and `previous_positions` for motion detection, `position_sent` to ensure data is only sent once when the manipulator stops, `execute_signal` to indicate when the execute command is received, and `position_threshold` to determine if the manipulator is in motion. Key functions include `isStopped()`, which compares current and previous positions to detect if the robot has stopped, and `jointStateCallback()`, which updates the position arrays when new joint state messages are received. The `executeSignalCallback()` function handles the execution signal by sending joint positions to the server once movement stops and providing feedback to Unity via the `unity_feedback` topic. The `main()` function initializes the ROS node, sets up publishers and subscribers, and continuously checks for joint state updates, resetting the `position_sent` flag when movement is detected. This setup ensures continuous communication and real-time control of the manipulator's movements.

4.3.2 UDP Receiver : Communication with Franka Robot

For clarification this code is implemted as mention in **figure 4.1** following C++ program establishes a real-time communication framework for controlling a Franka robot using UDP sockets. It utilizes multi-threading for efficient bidirectional data exchange between the robot and an external system (e.g., MATLAB). The program consists of `udpReceiver()` . The `udpReceiver()` function initializes a UDP socket to listen for incoming data, specifically joint reference positions for the robot. It parses the received data into a shared array `joints_ref` while ensuring thread safety using a mutex. Conversely, `udpSender()` sends the current joint positions of the robot, stored in `joints_pos`, to a specified UDP port. Both functions operate in infinite loops to maintain continuous communication. The main function handles the robot initialization and manages the multi-threaded execution of `udpReceiver()` and `udpSender()` using the `std::thread` library. It initializes the robot and, depending on the `realRobot` flag, executes movement commands based on the received joint references using Franka's motion control APIs. The robot's collision behavior and initial configuration are defined to ensure safe operation. Upon receiving valid references through UDP, the robot moves to the target positions, leveraging mutex locks to safely access shared variables. The program continuously monitors and updates the robot's state in a loop until manually terminated, demonstrating robust real-time communication and control for industrial applications.

Chapter 5

Experimental Results

5.1 Experiment Desciption

After the development is accomplished, we want to evaluate if the system is usable by a regular user and how easy it is to perform the task. We also want to measure the workload experienced by the user during the task. To do this, we set-up an experiment with 10 volunteers. Each user will first use the HoloLens with it, then uses the developed system for 15 mins to become familiar with the interface and the general functionality of the system. This familiarization phase is important to ensure that users understand how to interact with the system before starting the task.

The task involves starting with the robot in its home position, picking up a sponge, placing it in Area A, B and C in order. The sponge was chosen because it is soft, which prevents the robot's "Reflexive Mode" from being activated upon external impact. It is also easy to grab, even if the gripper is not perfectly aligned.

To evaluate the usability and workload, we will use two quantitative questionnaires: the System Usability Scale (SUS) to measure usability and the NASA-TLX to measure the workload. These metrics will help us understand how easy it is for a regular user to perform the task and how much effort is required. The experiment will be conducted under controlled conditions, with each user given the same instructions and time limits to ensure consistency. The time taken for completion of each task will be recorded throughout the experiment (the tasks will have to cover the expected period where applicable). Then two questionnaires will be sent to the user for completion.

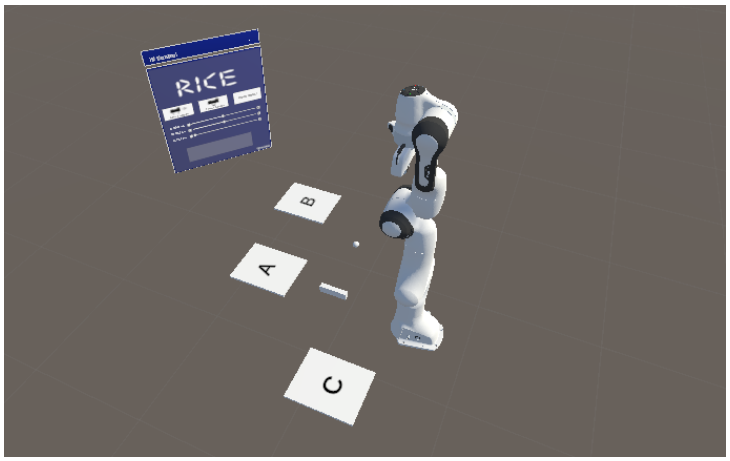


Figure 5.1 Thesis Experiment

5.1.1 What is SUS?

The System Usability Scale (SUS) is a simple way to check how easy a system, product, or interface is to use. It was created by John Brooke in 1986 and helps measure usability based on what users think. The SUS has 10 questions, and users answer them by choosing a number from 1 (strongly disagree) to 5 (strongly agree). It composed by 10 main questions.

The SUS Questionnaire

Users respond to the following 10 statements, rating their agreement on a 5-point scale:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think I would need the support of a technical person to use this system.
5. I found the various functions in this system well-integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.

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

Why SUS? : it has many advantages, it is quick and easy for users to complete, works for many types of systems, and gives a clear usability score that helps compare different products.

5.1.2 What is NASA TLX?

The **NASA Task Load Index (NASA-TLX)** is a widely used subjective workload assessment tool that evaluates perceived workload based on six dimensions listed in the figure 5.2 . It is commonly applied in human factors research, usability testing to measure the cognitive and physical demands of tasks (79)

NASA-TLX Dimensions

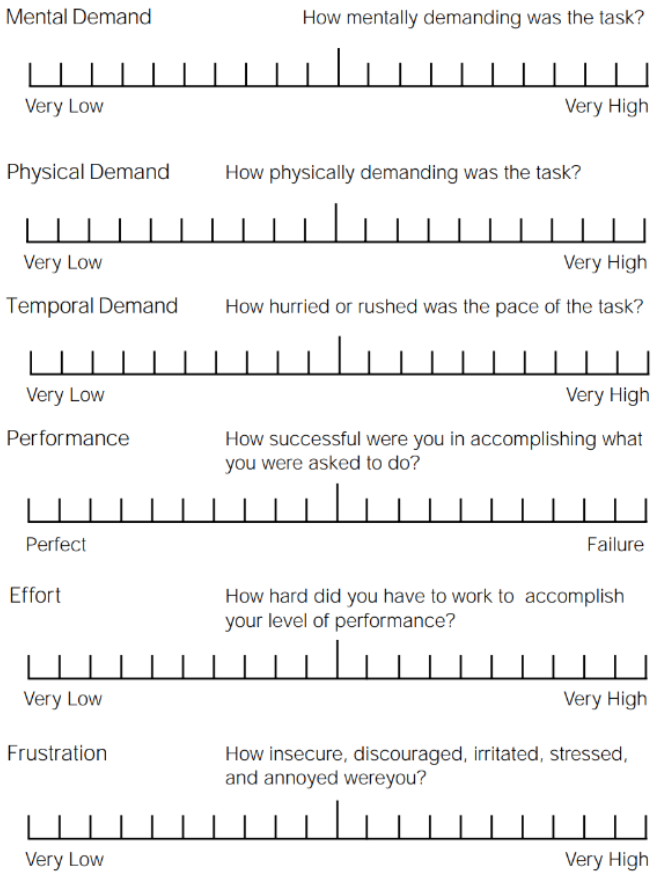


Figure 5.2 NASA-TLX Questionnaire
(80)

5.2 Results

5.2.1 Time

The following table contains the time taken by the users to finish each task:

User	A	B	C
1	1:50 min	1:37 min	59 sec
2	1:04 min	1:12 min	43 sec
3	2:05 min	1:16 min	52 sec
4	29 sec	27 sec	23 sec
5	1:14 min	1:45 min	1:12 min
6	1:42 min	1:01 min	43 sec
7	1:06 min	56 sec	49 sec
8	1:30 min	1:45 min	1:40 min
9	50 sec	55 sec	45 sec
10	1:32 min	1:10 min	1:03 min
avg	**1:29 min**	**1:12 min**	**55 sec**

Table 5.1 Time Results

5.2.2 Questionnaires

After finishing the tasks, the user was then presented with two questionnaires (NASA TLX and SUS) to assess their general perception of workload with the system, and the usability of the system. the results were as follows:

User	SUS Score	NASA TLX Score
1	95	30
2	85	38
3	90	40
4	83	35
5	88	31
6	82	28
7	90	32
8	85	39
9	93	34
10	89	41
Average	**88**	**34.8**

Table 5.2 SUS and NASA TLX Score

This graph shows the six factors separately, in the graph there are: Mental Demand, Physical Demand, Temporal Demand, Performance, Effort, and Frustration. Each factor was rated by 10 people, and the average score for each is shown below:

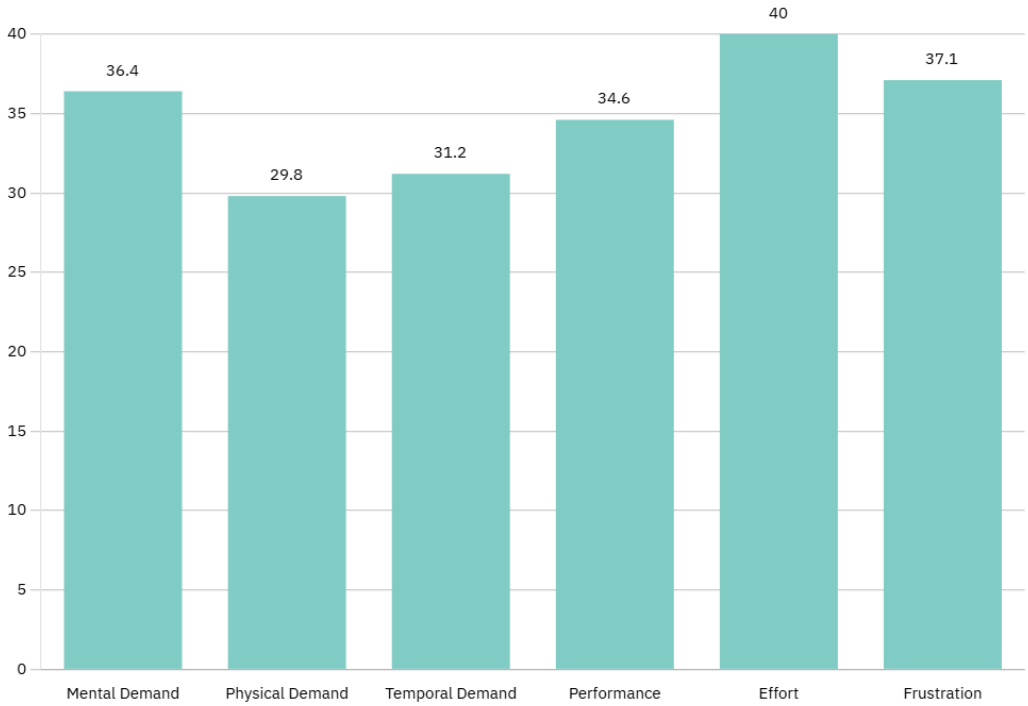


Figure 5.3 Average factors score

From the graph, we can see which factors had higher or lower scores. If a factor has a high score, it means the task was more difficult in that area. For example, Effort is high, users found it a bit tiring. Physical Demand is low, the task was not very demanding. From the graph, we see that Mental Demand is 36.4, meaning users had to think but not too much. In our case, Physical Demand is 29.8, showing the task was not very tiring. Temporal Demand is 31.2, meaning users did not feel too rushed. Performance is 34.6, which means users felt they did their task well, but there is room for improvement. Effort is 40.0, showing that users had to put in some energy. Frustration is 37.1, meaning users felt a bit annoyed but not too much.

5.3 Discussion of the results

In this part we discuss the results obtained by our volunteers from both perspective time and questionnaires.

5.3.1 Time

The results presented in Table 5.1 highlight a noticeable trend in task completion times across the three areas (A, B, and C). The average time for each step shows a clear decrease, with Area A taking the longest (**1:29 min**), followed by Area B (**1:12 min**), and finally Area C (**55 sec**). This downward trend suggests that as users progress through the experiment, they become more efficient in executing the pick-and-place tasks.

This improvement can be attributed to the **learnability** of the system, as users gain familiarity with the HoloLens interface and the robotic control mechanisms. Initially, in Area A, users likely spend more time understanding how to interact with the system, leading to longer execution times. By the time they reach Area B, their performance improves due to increased confidence and a better grasp of the system's functionalities.

Finally, in Area C, the fastest average time is recorded, indicating that users have effectively adapted to the workflow, requiring minimal cognitive effort to complete the task. Overall, these results validate the usability and learnability of the system, demonstrating that users can progressively enhance their efficiency with experience.

T-Test Analysis

To determine if there is a statistically significant difference in task completion times across the three areas (A, B, and C), we perform a paired t-test. This test compares the means of two related groups to check if their differences are statistically significant.

Defining Hypotheses

For each comparison, we set:

- **Null Hypothesis** (H_0): There is no significant difference between the task completion times of the two areas.
- **Alternative Hypothesis** (H_A): There is a significant difference between the task completion times.

Data Collection

The average task completion times are:

- **Area A**: 1:29 min (89 sec)
- **Area B**: 1:12 min (72 sec)
- **Area C**: 55 sec

Applying the Paired T-Test

We conduct t-tests for the following pairs:

- **A vs. B**
- **B vs. C**
- **A vs. C**

Results of the T-Test

The results are summarized as follows:

Comparison	t-value	p-value
A vs. B	0.997	0.345 (Not significant)
B vs. C	4.359	0.0018 (Significant)
A vs. C	2.922	0.017 (Significant)

Table 5.3 T-Test Results for Task Completion Times

So in the end we conclude thanks to T-Test that :

- There is **no significant difference** between Area A and Area B ($p = 0.345$).
- There is a **significant difference** between Area B and Area C ($p = 0.0018$) so it means that users performed in average significantly faster in Area C than in Area B.
- There is a **significant difference** between Area A and Area C ($p = 0.017$) it also confirms that users were significantly faster in Area C than in Area A.

5.3.2 SUS & NASA TLX score

Results from the System Usability Scale (SUS) and NASA Task Load Index (NASA TLX) scores are shown in Table 5.2. These scores help us understand how easy the system is to use and how much mental effort it requires. The average SUS score is 88, which means the system is rated as "Excellent" based on usability standards. This shows that users found it easy to interact with the system and understand how it works. This also matches the task completion times, where users became faster as they practiced.

For the NASA TLX score, the average is 34.8, which means the system caused a moderate mental workload. This means the system needed some thinking and effort to use, but it was not too hard. Looking at the individual NASA TLX factors: Mental Demand, Physical Demand, Temporal Demand, Performance, Effort, and Frustration, we can see how different parts of the task affected users.

Overall, the results show that the system is easy to learn and use. Users could control the HoloLens 2 and the robotic system without too much difficulty.

Chapter 6

Conclusion

Augmented reality is making it easier for people to control robots. Instead of using difficult computer commands or complex screens, AR lets users control robots in a simple and natural way. This is very helpful, especially for people who are not experts. This thesis focuses on using AR to control the Franka Emika Panda robotic arm. It uses Microsoft HoloLens 2, Unity3D, and the Robot Operating System to help people move and control the robot in real time.

Old robot control methods need special skills and can be hard to learn. Many systems use text commands or complicated software, which makes them difficult for beginners. AR makes this easier by allowing people to use their hands and eyes to control the robot. This improves accuracy, speed, and ease of use, making it better for work and research. In this project, the AR-based control system was carefully tested and developed. It lets users set points for the robot to move to, change positions.

Real-time feedback helps users make sure the robot moves correctly. It allows small adjustments to improve accuracy and avoid mistakes. This thesis introduces a system that connects Unity3D and ROS, creating a two-way connection between the HoloLens 2 and the robotic arm. It also includes a tool called MoveIt, which helps the robot to move. A strong communication system, called ROS-TCP Connector, links Unity and ROS, making data transfer fast and efficient.

The system was tested in both a computer simulation and real-life situations. It showed that users could control the robot with high accuracy. The system worked just as well as traditional methods but was easier to use. People found the AR interface simple to learn, and the robot responded quickly with very little delay. The system can also be improved in the future to allow multiple users and work with different robots, making it a flexible and useful tool for controlling robots with AR.

6.1 Limitation & Improvements

6.1.1 Limitations

During the development of our system, also after receiving the feedback from our volunteers it helped us conclude the limitations of our system and it could be stated as follows:

- **The tiny field of view:** The difficulty to see a wild field of view was one of the challenges where on Hololens2 you can only see **52**°
- **A delayed response solving time:** The user complained about the time **MoveIt** was taken to generate the solution in order to find the suitable movement of the robot.
- **Hologram Calibration:** What makes the system hard to start is the calibration of the Holographic Robot to make it perfectly aligned with the real robot, this very important step in order to make the real and the virtual robot meet at the exact same point.
- **Automatic execution:** This would be a problem if the user wants the motion to be executed directly to the real robot, the system was designed the execution to be manual in case the user wants to move the real robot, the holographic motion must be executed each time.

6.1.2 Improvements:

Beside the limitation that were stated in this project can be improved and modified to be more practical, useful and controllable if some consideration were taken into account:

- **Use Alternative Control System:** MoveIt was used as a convenient solution for motion planning. However, in a real-world implementation, a more dedicated and optimized control system could significantly reduce motion execution time and improve efficiency.
- **FK Implementation:** The implementation **Forward kinematics:** solution will help the movement of the robot to be more optional and suitable to the need of the user. Imagine having a slider besied each joint that permits you to rotate a desired joint into a certain value with out the need to modify any other joint.
- **Implement automatic mode:** This will help the user to have a faster experience to move the real robot directly with giving the permission each time.

- **Automatic Calibration:** The hard calibration problem could be solved using an Aruco marker place in the area of the real robot. then the distance difference will be needed in order to calibrate automatically the holographic robot.

6.2 Future direction

In the future, AR robot control can get better in many ways. One big problem is the small field of view in the HoloLens 2. Right now, people have to move their heads a lot to see the holograms. If the field of view gets bigger or if extra cameras are added, it will feel more natural and less tiring.

Hand and eye tracking also need to improve. Sometimes, the system doesn't understand movements correctly. If tracking becomes better, mistakes will happen less, and the robot will respond faster. Maybe in the future, people can use special gloves or feel vibrations when they touch something in AR. This would help people feel what the robot is doing, like if it is pushing against something.

Another thing that can be better is the way the system and the robot talk to each other. Right now, it depends a lot on local computers. If it uses cloud computing, it could be much faster, and robots could do more difficult tasks without needing strong computers nearby.

It would also be great if more than one person could control the same robot at the same time. This could help teams work together, even from far away. Maybe AI can also help by guessing what the user wants to do and making control easier, which could be useful in hospitals or factories.

Finally, the system should not only work with the Franka Emika Panda robot. It should work with many different robots so more people can use it. If this happens, AR robot control could become a normal thing in research, industry, and everyday life.

References

[1] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018. [Online]. Available: https://doi.org/10.1016/j.mechatronics.2018.09.002

[2] R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt, "Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–19. [Online]. Available: https://doi.org/10.1145/3491102.3517719

[3] A. Dey, M. Billinghurst, R. W. Lindeman, and J. Swan, "A systematic review of 10 years of augmented reality usability studies: 2005 to 2014," *Frontiers in Robotics and AI*, vol. 5, 2018. [Online]. Available: https://doi.org/10.3389/frobt.2018.00037

[4] B. Vanderborght, A. Albu-Schäffer, A. Bicchi, E. Burdet, D. Caldwell, R. Carloni, and et al., "Variable impedance actuators: a review," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1601–1614, 2013.

[5] G. Hirzinger, A. Albu-Schaffer, M. Hahnle, I. Schaefer, and N. Sporer, "On a new generation of torque controlled light-weight robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4. IEEE, 2001, pp. 3356–3363.

[6] M. Zinn, B. Roth, O. Khatib, and J. Salisbury, "A new actuation approach for human friendly robot design," *International Journal of Robotics Research*, vol. 23, no. 4-5, pp. 379–398, 2004.

[7] A. Bauer, D. Wollherr, and M. Buss, "Human–robot collaboration: a survey," *International Journal of Humanoid Robotics*, vol. 5, no. 01, pp. 47–66, 2008.

[8] M. Geravand, F. Flacco, and A. D. Luca, "Human–robot physical interaction and collaboration using an industrial robot with a closed control architecture," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 4000–4007.

[9] M. Walker, H. Hedayati, J. Lee, and D. Szafir, "Communicating robot motion intent with augmented reality," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 316–324. [Online]. Available: https://doi.org/10.1145/3171221.3171253

[10] A. Watanabe, T. Ikeda, Y. Morales, K. Shinozawa, T. Miyashita, and N. Hagita, "Communicating robotic navigational intentions," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5763–5769. [Online]. Available: https://doi.org/10.1109/IROS.2015.7354195

[11] M. Ostanin and A. Klimchik, "Interactive robot programming using mixed reality," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 50–55, 2018.

[12] J. R. Cauchard, A. Tamkin, C. Y. Wang, L. Vink, M. Park, T. Fang, and J. A. Landay, "Drone.io: A gestural and visual interface for human-drone interaction," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 153–162. [Online]. Available: https://doi.org/10.1109/HRI.2019.8673011

[13] C. Guo, J. E. Young, and E. Sharlin, "Touch and toys: New techniques for interaction with a remote group of robots," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 491–500. [Online]. Available: https://doi.org/10.1145/1518701.1518780

[14] D. Leithinger, S. Follmer, A. Olwal, S. Luescher, A. Hogge, J. Lee, and H. Ishii, "Sublimate: state-changing virtual and physical rendering to augment interaction with shape displays," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 1441–1450. [Online]. Available: https://doi.org/10.1145/2470654.2466191

[15] D. Robert and C. Breazeal, "Blended reality characters," in *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, 2012, pp. 359–366. [Online]. Available: https://doi.org/10.1145/2157689.2157810

[16] D. Lindlbauer, J. E. Grønbæk, M. Birk, K. Halskov, M. Alexa, and J. Müller, "Combining shape-changing interfaces and spatial augmented reality enables extended object appearance," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 791–802. [Online]. Available: https://doi.org/10.1145/2858036.2858457

[17] E. W. Pedersen and K. Hornbæk, "Tangible bots: interaction with active tangibles in tabletop interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 2975–2984.

[18] A. Özgür, S. Lemaignan, W. Johal, M. Beltran, M. Briod, L. Pereyre, F. Mondada, and P. Dillenbourg, "Cellulo: Versatile handheld robots for education," in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2017, pp. 119–127.

[19] L. Chen, K. Takashima, K. Fujita, and Y. Kitamura, "Pinpointfly: An egocentric position-control drone interface using mobile ar," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–13. [Online]. Available: https://doi.org/10.1145/3411764.3445110

[20] S. Kasahara, R. Niiyama, V. Heun, and H. Ishii, "extouch: spatially-aware embodied manipulation of actuated objects mediated by augmented reality," in *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*, 2013, pp. 223–228.

[21] R. Lingamaneni, T. Kubitza, and J. Scheible, "Dronecast: towards a programming toolkit for airborne multimedia display applications," in *Proceedings of the 19th International Conference on Human Computer Interaction with Mobile Devices and Services*, 2017, pp. 1–8. [Online]. Available: https://doi.org/10.1145/3098279.3122128

[22] S. Hashimoto, A. Ishida, M. Inami, and T. Igarashi, "Touchme: An augmented reality based remote robot manipulation," in *Proceedings of the 21st International Conference on Artificial Reality and Telexistence (ICAT 2011), Vol. 2*, 2011.

[23] M. L. Lupetti, G. Piumatti, C. Germak, and F. Lamberti, "Design and evaluation of a mixed-reality playground for child-robot games," *Multimodal Technologies and Interaction*, vol. 2, p. 69, 2018.

[24] H. Hedayati, M. Walker, and D. Szafir, "Improving collocated robot teleoperation with augmented reality," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 78–86. [Online]. Available: https://doi.org/10.1145/3171221.3171251

[25] H. Peng, J. Briggs, C.-Y. Wang, K. Guo, J. Kider, S. Mueller, P. Baudisch, and F. Guimbretière, "Roma: Interactive fabrication with augmented reality and a robotic 3d printer," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[26] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. M. V. der Loos, and E. Croft, "Robot programming through augmented trajectories in augmented reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1838–1844.

[27] A. F. Siu, S. Yuan, H. Pham, E. Gonzalez, L. H. Kim, M. L. Goc, and S. Follmer, "Investigating tangible collaboration for design towards augmented physical telepresence," in *Design Thinking Research*. Springer, 2018, pp. 131–145.

[28] S. Follmer, D. Leithinger, A. Olwal, A. Hogge, and H. Ishii, "inform: Dynamic physical affordances and constraints through shape and object actuation," in *Proceedings of UIST*, vol. 13, 2013, pp. 2 501 988–2 502 032. [Online]. Available: https://doi.org/10.1145/2501988.2502032

[29] L. Qian, A. Deguet, Z. Wang, Y.-H. Liu, and P. Kazanzides, "Augmented reality assisted instrument insertion and tool manipulation for the first assistant in robotic surgery," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5173–5179.

[30] F. Muhammad, A. Hassan, A. Cleaver, and J. Sinapov, "Creating a shared reality with robots," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 614–615.

[31] K. Takashima, T. Oyama, Y. Asari, E. Sharlin, S. Greenberg, and Y. Kitamura, "Study and design of a shape-shifting wall display," in *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, 2016, pp. 796–806. [Online]. Available: https://doi.org/10.1145/2901790.2901892

[32] T. B. Sheridan, "Telerobotics, automation, and human supervisory control," *MIT Press*, 1992. [Online]. Available: https://doi.org/10.7551/mitpress/6826.001.0001

[33] M. Dianatfar, J. Latokartano, and M. Lanz, "Review on existing vr/ar solutions in human–robot collaboration," *Procedia CIRP*, vol. 97, pp. 407–411, 2021. [Online]. Available: https://doi.org/10.1016/j.procir.2020.05.259

[34] S. A. Green, M. Billinghurst, X. Chen, and J. G. Chase, "Human-robot collaboration: A literature review and augmented reality approach in design," *International Journal of Advanced Robotic Systems*, vol. 5, no. 1, p. 1, 2008. [Online]. Available: https://doi.org/10.5772/5664

[35] I. Poupyrev, T. Nashida, and M. Okabe, "Actuation and tangible user interfaces: the vaucanson duck, robots, and shape displays," in *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, 2007, pp. 205–212.

[36] C. Attig, D. Wessel, and T. Franke, "Assessing personality differences in human-technology interaction: an overview of key self-report scales to predict successful interaction," in *HCI International 2017 – Posters' Extended Abstracts*, ser. Lecture Notes in Computer Science, C. Stephanidis, Ed. Springer International Publishing, 2017, vol. 10288, pp. 19–29.

[37] V. Herdel, A. Kuzminykh, A. Hildebrandt, and J. R. Cauchard, "Drone in love: Emotional perception of facial expressions on flying robots," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–20. [Online]. Available: https://doi.org/10.1145/3411764.3445495

[38] Y. Jansen, P. Dragicevic, P. Isenberg, J. Alexander, A. Karnik, J. Kildal, S. Subramanian, and K. Hornbæk, "Opportunities and challenges for data physicalization," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 3227–3236. [Online]. Available: https://doi.org/10.1145/2702123.2702180

[39] M. Bandala, C. West, S. Monk, A. Montazeri, and C. J. Taylor, "Vision-based assisted tele-operation of a dual-arm hydraulically actuated robot for pipe cutting and grasping in nuclear environments," *Robotics*, vol. 8, no. 2, 2019. [Online]. Available: https://doi.org/10.3390/robotics8020042

[40] B. Jones, Y. Zhang, P. N. Wong, and S. Rintel, "Belonging there: Vroom-ing into the uncanny valley of xr telepresence," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW1, pp. 1–31, 2021. [Online]. Available: https://doi.org/10.1145/3449133

[41] J. E. Young, M. Xin, and E. Sharlin, "Robot expressionism through cartooning," in *2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2007, p. 309–316. [Online]. Available: https://doi.org/10.1145/1228716.1228758

[42] R. Darbar, J. S. Roo, T. Lainé, and M. Hachet, "Dronesar: Extending physical spaces in spatial augmented reality using projection on a drone," in *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia*, 2019, pp. 1–7. [Online]. Available: https://doi.org/10.1145/3365610.3365631

[43] A. Roudaut, A. Karnik, M. Löchtefeld, and S. Subramanian, "Morphees: toward high "shape resolution" in self-actuated flexible mobile devices," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2013, pp. 593–602. [Online]. Available: https://doi.org/10.1145/2470654.2470738

[44] S. A. Moubayed, J. Beskow, G. Skantze, and B. Granström, "Furhat: a back-projected human-like robot head for multiparty human-machine interaction," in *Cognitive Behavioural Systems*. Springer, 2012, pp. 114–130. [Online]. Available: https://doi.org/10.1007/978-3-642-34584-5_9

[45] W. Yamada, K. Yamada, H. Manabe, and D. Ikeda, "isphere: Self-luminous spherical drone display," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, 2017, p. 635–643. [Online]. Available: https://doi.org/10.1145/3126594.3126631

[46] R. S. Andersen, O. Madsen, T. B. Moeslund, and H. B. Amor, "Projecting robot intentions into human environments," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (ROMAN)*. IEEE, 2016, pp. 294–301. [Online]. Available: https://doi.org/10.1109/ROMAN.2016.7745182

[47] I. Suzuki, S. Yoshimitsu, K. Kawahara, N. Ito, A. Shinoda, A. Ishii, T. Yoshida, and Y. Ochiai, "Gushed diffusers: Fast-moving, floating, and lightweight midair display," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, 2016, pp. 69–70. [Online]. Available: https://doi.org/10.1145/2984751.2985706

[48] H. Tobita, S. Maruyama, and T. Kuji, "Floating avatar: blimp-based telepresence system for communication and entertainment," in *ACM SIGGRAPH 2011 Emerging Technologies*, 2011, pp. 1–1. [Online]. Available: https://doi.org/10.1145/2048259.2048263

[49] A. M. Villanueva, Z. Liu, Z. Zhu, X. Du, J. Huang, K. A. Peppler, and K. Ramani, "Robotar: An augmented reality compatible teleconsulting robotics toolkit for augmented makerspace experiences," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–13. [Online]. Available: https://doi.org/10.1145/3411764.3445726

[50] C. Li, A. Fahmy, and J. Sienz, "An augmented reality based human-robot interaction interface using kalman filter sensor fusion," *Sensors*, vol. 19, no. 20, p. 4586, 2019. [Online]. Available: https://doi.org/10.3390/s19204586

[51] Z. Makhataeva, A. Zhakatayev, and H. A. Varol, "Safety aura visualization for variable impedance actuated robots," in *2019 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2019, pp. 805–810.

[52] W. P. Chan, A. Karim, C. P. Quintero, H. F. M. V. der Loos, and E. Croft, "Virtual barriers in augmented reality for safe human-robot collaboration in manufacturing," in *Robotic Co-Workers 4.0 2018: Human Safety and Comfort in Human-Robot Interactive Social Environments*, 2018.

[53] E. Rosen, D. Whitney, E. Phillips, G. Chien, J. Tompkin, G. Konidaris, and S. Tellex, "Communicating robot arm motion intent through mixed reality head-mounted displays," in *Robotics Research*. Springer, 2020, pp. 301–316. [Online]. Available: https://doi.org/10.1007/978-3-030-28619-4_26

[54] T. Groechel, Z. Shi, R. Pakkar, and M. J. Matarić, "Using socially expressive mixed reality arms for enhancing low-expressivity robots," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2019, pp. 1–8. [Online]. Available: https://doi.org/10.1109/RO-MAN46459.2019.8956458

[55] J. Ballantyne *et al.*, "The da vinci surgical system: A review," *Surgical Innovation*, vol. 9, no. 2, pp. 127–134, 2002. [Online]. Available: https://doi.org/10.1177/155335060200900203

[56] R. Murphy *et al.*, "Search and rescue robotics," *Springer Handbook of Robotics*, pp. 1–20, 2004. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5_63

[57] R. Volpe *et al.*, "Remote rover operations: Experiences from the mars pathfinder mission," *Journal of Field Robotics*, vol. 18, no. 1, pp. 1–15, 2001. [Online]. Available: https://doi.org/10.1002/rob.10001

[58] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: A survey," *Foundations and Trends in Human-Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2008. [Online]. Available: https://doi.org/10.1561/1100000005

[59] Y. Gao and C.-M. Huang, "Pati: A projection-based augmented table-top interface for robot programming," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 2019, pp. 345–355. [Online]. Available: https://doi.org/10.1145/3301275.3302326

[60] C. Papachristos and K. Alexis, "Augmented reality-enhanced structural inspection using aerial robots," in *2016 IEEE International Symposium on Intelligent Control (ISIC)*, 2016, pp. 1–6.

[61] L. Yuan, C. Reardon, G. Warnell, and G. Loianno, "Human gaze-driven spatial tasking of an autonomous mav," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, p. 1343–1350, 2019. [Online]. Available: https://doi.org/10.1109/LRA.2019.2895419

[62] M. E. Walker, H. Hedayati, and D. Szafir, "Robot teleoperation with augmented reality virtual surrogates," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 202–210.

[63] K. C. Hoang, W. P. Chan, S. Lay, A. Cosgun, and E. Croft, "Virtual barriers in augmented reality for safe and effective human robot cooperation in manufacturing," *arXiv preprint arXiv:2104.05211*, 2021. [Online]. Available: https://arxiv.org/abs/2104.05211

[64] I. Maly, D. Sedláček, and P. Leitao, "Augmented reality experiments with industrial robot in industry 4.0 environment," in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, 2016, pp. 176–181. [Online]. Available: https://doi.org/10.1109/INDIN.2016.7819154

[65] D. Mourtzis, V. Zogopoulos, and E. Vlachou, "Augmented reality application to support remote maintenance as a service in the robotics industry," *Procedia Cirp*, vol. 63, pp. 46–51, 2017.

[66] O. Erat, W. A. Isop, D. Kalkofen, and D. Schmalstieg, "Drone-augmented human vision: Exocentric control for drones exploring hidden areas," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 4, pp. 1437–1446, 2018. [Online]. Available: https://doi.org/10.1109/TVCG.2018.2794058

[67] K. Liu, D. Sakamoto, M. Inami, and T. Igarashi, "Roboshop: multi-layered sketching interface for robot housework assignment and management," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 647–656.

[68] Y. Ochiai and K. Toyoshima, "Homunculus: the vehicle as augmented clothes," in *Proceedings of the 2nd Augmented Human International Conference*, 2011, pp. 1–4.

[69] D. Bambusek, Z. Materna, M. Kapinus, V. Beran, and P. Smrž, "Combining interactive spatial augmented reality with head-mounted display for end-user collaborative robot programming," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019, pp. 1–8. [Online]. Available: https://doi.org/10.1109/RO-MAN46459.2019.8956315

[70] B. Jones, Y. Zhang, P. N. Wong, and S. Rintel, "Vroom: Virtual robot overlay for online meetings," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–10. [Online]. Available: https://doi.org/10.1145/3334480.3382820

[71] Microsoft, "Hololens 2: Mixed reality for business," retrieved from https://www.microsoft.com/en-us/hololens.

[72] Microsoft Corporation, *Microsoft Mixed Reality Toolkit (MRTK): Open-Source Toolkit for MR Development*. [Online]. Available: https://github.com/microsoft/MixedRealityToolkit-Unity

[73] MoveIt Contributors, *MoveIt: Motion Planning Framework for ROS*, 2025. [Online]. Available: https://moveit.ros.org

[74] F. Community, *Franka Emika GitHub Repository*. [Online]. Available: https://github.com/frankaemika

[75] ROS Community and Unity Technologies, *ROS-TCP Connector: Integrating Unity with ROS for Real-Time Robot Control*. [Online]. Available: https://github.com/Unity-Technologies/ROS-TCP-Connector

[76] ROS Community, *ROS and UDP Communication: Real-Time Data Streaming in Robotics*. [Online]. Available: https://wiki.ros.org/roscpp/Overview/Communication

[77] Microsoft, *Holographic remote play Mode for Hololens2*. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/preview-and-debug-your-app?tabs=openxr

[78] F. comunity, *franka unity model*. [Online]. Available: https://github.com/jabolcni/unity_franka_IK

[79] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," *Advances in Psychology*, vol. 52, pp. 139–183, 1988.

[80] L. Larcen, *MRTK guide*. [Online]. Available: https://onedrive.live.com/?redeem=aHR0cHM6Ly8xZHJ2Lm1zL2IvYy9jODMxNWEzNDgxYWQzZmYwL0VSU1gwRC1iQWlsRnJEX
cid=C8315A3481AD3FF0&id=C8315A3481AD3FF0%21s3fd09714029b4529ac3ff34d9116049e&parId=C8315A3481AD3FF0%21s32d993efe72b49d98f870e891230d86c&o=OneUp