POLITECNICO DI TORINO

Department of Control and Computer Engineering (DAUIN)

Master's Degree in Computer Engineering

Academic Year 2025/2026

Master's Thesis

Augmented Reality Mediated Control of a UR3e Collaborative Robot Using HoloLens 2

A Point and Pick Interface with ROS Integration

Supervisors
Prof. Luca Mastrogiacomo
Matteo Capponi

Candidate
Seyedmohammad Khataeipour
Student ID: S301893

Torino, 2026

Abstract

Collaborative robots are increasingly adopted in manufacturing environments in which operators must interact directly with automated systems at workstation level, particularly in tasks such as collaborative assembly support, pick and place, and material handling. Despite their potential, many existing cobot interfaces still rely on detached control devices, programming oriented workflows, or interaction methods that are not sufficiently intuitive for flexible and operator centered use.

This thesis investigates the use of augmented reality (AR) as an interaction layer for collaborative robotic manipulation in a structured workstation scenario. The proposed system is developed for a Universal Robots UR3e collaborative robot equipped with an OnRobot RG6 gripper and uses Microsoft HoloLens 2 as the primary interaction device. The AR application is implemented in Unity with the Mixed Reality Toolkit (MRTK), while communication with the robotic system is handled through a ROS Noetic backend via ROS# WebSockets.

The interaction model is based on a multimodal approach that combines hand gestures, voice commands, and a Point and Pick paradigm. Through this approach, the operator designates targets directly in three dimensional space using a pointing interaction, while the matched preset and destination are logged to the ROS workstation terminal as a record of each executed command. Execution is triggered through a deliberate user command, preserving operator authority over the timing and decision to act. On the robot side, the UR3e is controlled through the Universal Robots ROS Driver, while robot specific calibration and AR to robot registration procedures are used to align the virtual and physical workspaces. Gripper actuation is implemented through an Eyebox interface, enabling reliable operation despite hardware communication constraints.

Beyond system implementation, the thesis includes an exploratory case study in a controlled laboratory workstation and a user centered usability assessment. The results indicate that the proposed interaction approach improves clarity of robot intent, supports structured supervision of collaborative manipulation, and reduces reliance on continuous low level manual correction. At the same time, the study highlights important deployment constraints, including spatial registration drift, gesture recognition sensitivity, and HoloLens side performance degradation under bursts of rapid interaction.

Overall, the work shows that AR mediated control can provide a technically feasible and operationally meaningful interaction paradigm for structured collaborative workstation tasks, while remaining compatible with ROS based industrial robot integration. The

findings suggest that the practical value of the approach lies not only in intuitive interaction, but also in improved transparency, standardized execution, and more accessible supervision in manufacturing oriented collaborative scenarios.

Keywords: Human Robot Collaboration, Augmented Reality, HoloLens 2, ROS, UR3e, Unity, MRTK, Collaborative Assembly, Multimodal Interaction, Collaborative Workstations

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Luca Mastrogiacomo, for his guidance and continuous support throughout this work. His feedback and insights have been fundamental in shaping the direction of this thesis.

I also thank the assistant supervisor, Matteo Capponi, and the laboratory responsible, Khurshid Aliev, for their technical support and for providing access to the experimental setup.

Finally, I would like to thank all the people who contributed, directly or indirectly, through discussions, suggestions, and encouragement during the development of this work.

List of Acronyms

AR	Augmented Reality
MR	Mixed Reality
HRC	Human Robot Collaboration
HRI	Human Robot Interaction
NUI	Natural User Interface
ROS	Robot Operating System
MRTK	Mixed Reality Toolkit
UWP	Universal Windows Platform
URDF	Unified Robot Description Format
IK	Inverse Kinematics
TF	Transformation Frames (ROS TF2 system)
RTDE	Real Time Data Exchange

Contents

Abstract i

Acknowledgements iii

List of Acronyms iv

List of Figures xii

List of Tables xiv

1 Introduction 1

1.1 Background and Motivation 1

1.2 Research Problem 2

1.3 Objectives and Research Questions 3

1.4 Scope and Assumptions 4

1.5 Thesis Structure 5

2 Literature Review 6

2.1 Collaborative Robotics in Manufacturing and Assembly 6

2.2 Human Robot Interaction in Collaborative Workstations 7

2.3 Augmented and Mixed Reality for Industrial Human Robot Collaboration . 8

2.4 HoloLens 2 as an Interaction Platform for Robotics 9

2.5 Multimodal Interaction for Robot Control 12

2.6 ROS Based Communication and Control Architectures 13

2.7 Technical, Human Factor, and Deployment Challenges 14

2.8 Evaluation Approaches for AR Mediated Human Robot Systems 15

2.9 Research Gap and Positioning of the Present Work 16

3 Methodology 18

3.1 Research Design and Evaluation Strategy . . . 18
3.2 Development Stack and Tools . . . 18
3.2.1 HoloLens 2 AR Interface (Unity/MRTK) . . . 19
3.2.2 ROS Backend (Ubuntu 20.04 / ROS Noetic) . . . 20
3.2.3 Universal Robots ROS Driver . . . 21
3.2.4 Unity ROS Communication (ROS# / rosbridge) . . . 22
3.2.5 Gripper Integration (OnRobot RG6 via Eyebox) . . . 23
3.2.6 Application Level ROS Nodes . . . 23
3.3 System Integration Workflow . . . 24
3.3.1 Robot Calibration . . . 24
3.3.2 Runtime Structure . . . 24
3.3.3 Connectivity Validation . . . 24
3.4 Interaction Model and Workspace Configuration . . . 24
3.4.1 System Pipeline . . . 24
3.4.2 Preset Based Workspace Modeling . . . 25
3.4.3 Interaction Design . . . 25
3.5 Exploratory Case Study Definition . . . 26
3.6 Assessment Protocol . . . 26
3.6.1 Participants . . . 28
3.6.2 Procedure . . . 28
3.6.3 Evaluation Instruments . . . 28
3.7 Data Collection and Analysis . . . 29

4 System Implementation 30

4.1 Chapter Overview . . . 30
4.2 Software Architecture and Module Decomposition . . . 30
4.2.1 Application Node Structure . . . 32
4.2.2 Custom Interaction Nodes . . . 32
4.3 Workspace Configuration and Preset Management . . . 34
4.3.1 Manual Preset Acquisition and Workspace Initialization . . . 34

4.4 HoloLens 2 Application Implementation . . . 35
4.4.1 Pointer and Gesture Processing . . . 37
4.4.2 Event Throttling . . . 37
4.5 ROS Communication Layer . . . 38
4.5.1 Message Encoding . . . 38
4.5.2 Command Handling (Application Entry Point) . . . 38
4.6 Transformation and Frame Handling . . . 39
4.7 Execution Logic . . . 39
4.8 Point and Pick Execution Pipeline . . . 39
4.8.1 Pose Generation . . . 39
4.8.2 Preset Snapping . . . 40
4.8.3 Execution Pipeline . . . 40
4.9 Robot Control Integration . . . 41
4.9.1 Motion Execution . . . 41
4.9.2 Feasibility Validation . . . 42
4.10 Gripper Control . . . 42
4.11 Error Handling and Safety . . . 42
4.12 Performance Optimization . . . 43
4.13 Build and Deployment . . . 43
4.14 Chapter Summary . . . 43

5 Exploratory Assessment Results 44

5.1 Overview . . . 44
5.2 Setup and Exploratory Case Study Context . . . 44
5.3 Evidence Sources and Assessment Criteria . . . 45
5.4 Case Study Execution Observations . . . 45
5.4.1 Task Flow and Execution Behavior . . . 47
5.4.2 Effect of Preset Based Execution . . . 48
5.4.3 Observed Limitations During Task Execution . . . 48
5.5 Basic Interaction and Control Results . . . 48

5.6 User Centred Usability Assessment . . . 49
5.6.1 Participants . . . 49
5.6.2 Perceived Usability . . . 51
5.6.3 Clarity of Robot Intent and Interaction Flow . . . 52
5.6.4 Perceived Robustness of Multimodal Interaction . . . 52
5.6.5 Comfort and Interaction Burden . . . 53
5.6.6 Interpretation of Usability Assessment Results . . . 55
5.7 Runtime Stability and Technical Constraints . . . 55
5.7.1 Communication Behavior . . . 55
5.7.2 Device Side Limitations . . . 56
5.8 Error Modes and Recovery . . . 56
5.9 Summary of Findings . . . 56

6 Discussion **58**

6.1 Overview . . . 58
6.2 Interpretation of the Main Findings . . . 58
6.3 Why Point and Pick Improves Task Execution . . . 59
6.4 Implications for Manufacturing Oriented Collaborative Workstations . . . 60
6.5 Management Engineering Perspective: Operational Value and Work System Design . . . 61
6.5.1 Productivity, Throughput, and Process Flow . . . 61
6.5.2 Quality and Error Prevention . . . 61
6.5.3 Standard Work and Variability Reduction . . . 62
6.5.4 Training, Skill Requirements, and Workforce Flexibility . . . 62
6.5.5 Safety Governance and Operational Compliance . . . 62
6.6 User Evaluation and Usability Implications . . . 63
6.7 Latency, Responsiveness, and Device Constraints as Deployment Factors . . . 64
6.8 Limitations and Edge Cases . . . 64
6.9 Threats to Validity . . . 65
6.10 Design Guidelines . . . 65

6.11 Summary 66

7 Conclusion and Future Work **67**

7.1 Summary of the Thesis 67

7.2 Key Contributions 67

7.3 Industrial Applicability 68

7.4 Limitations 69

7.5 Recommendations for Deployment 70

7.6 Future Work 71

7.7 Concluding Remarks 72

A System Architectures **73**

A.1 High Level System Architecture (HoloLens → ROS → UR3e) 73

A.2 Dataflow for Point and Pick and Preset Based Execution 74

B Topics and Command Interfaces **76**

B.1 Core Topics Used in the Project 76

B.2 High Level JSON Command Schema (`/hololens/command`) 77

B.3 Minimal Unity ROS# Payload Examples 77

C Parameters and Defaults **79**

C.1 Manipulation and Motion Parameters 79

C.2 UI and Interaction Policies 79

C.3 Gripper Parameters (RG6 via Eyebox) 79

D Calibration and Registration Procedures **81**

D.1 UR e Series Kinematic Calibration (Mandatory) 81

D.2 AR Robot Registration (HoloLens → `base_link`) 81

E Runtime Launch Procedure **83**

E.1 Overview 83

E.2 Core ROS and Robot Drivers 83

E.3 Application Layer (Unity Driven Orchestration) 84
E.4 Preset Management Utility 84
E.5 Diagnostic Test Publishers 85
E.6 Connectivity Troubleshooting 85

F Instrumentation and Logging **87**
F.1 Overview 87
F.2 ROS Side Instrumentation 87
F.3 Interface Side Instrumentation (HoloLens / Unity) 87
F.4 Performance and Responsiveness Indicators 88
F.5 Data Usage and Limitations 88

G Safety Policies and Fault Handling **89**
G.1 Overview 89
G.2 Interaction Level Safety 89
G.3 Communication and Runtime Safety 89
G.4 Robot Level Safety and Fault Handling 90
G.5 Motion Safety Strategies 90
G.6 Interface Level Stability and Safety 90
G.7 Summary 91

H Build and Deployment Notes (HoloLens 2) **92**
H.1 Overview 92
H.2 Build Configuration 92
H.3 Deployment Configuration 92
H.4 SDK Compatibility and Stability Notes 93
H.5 Practical Deployment Considerations 93

I Data and File Artifacts **94**
I.1 Overview 94
I.2 ROS Bag Files 94

I.3 Interface Log Files 95
I.4 Configuration Snapshots 95
I.5 Summary 96

List of Figures

2.1 Conceptual representation of spatial registration between the AR coordinate system and the robot reference frame. Accurate alignment ensures that virtual targets correspond to correct physical locations in the workspace. . 10

3.1 Methodological workflow of the proposed AR mediated control system, illustrating the four main phases from development to exploratory assessment. 19

3.2 Coordinate transformation chain between AR and robot frames. 25

3.3 Command-driven interaction flow of the Point and Pick system. The operator designates a target through spatial pointing, the system maps the input to the nearest preset, and execution begins only after an explicit user command. Each executed command is logged to the ROS workstation terminal. 26

3.4 Experimental laboratory setup used for the case study based evaluation of the AR mediated control system, including the UR3e robot, the structured workstation, and the operator wearing HoloLens 2. 27

4.1 Software pipeline of the implemented AR mediated control system. Operator input is acquired through the HoloLens 2 interface (Unity/MRTK), transmitted via ROS# and `rosbridge_websocket`, interpreted by custom ROS nodes, and executed through the UR control stack. Bidirectional communication enables continuous feedback for visualization, supervision, and safety monitoring. 31

4.2 Operator performing spatial target designation through the HoloLens 2 interface during use of the AR mediated control system. 36

5.1 Supervised execution during the case study task, showing the robot operating in the structured workstation environment. 46

5.2 Observed interaction and execution flow of the Point and Pick workflow. The operator designates a target through spatial interaction and explicitly triggers execution before the robot performs the manipulation task. Each command is logged to the ROS terminal. A feedback loop allows target re-designation if the wrong preset was selected. 47

5.3 User evaluation sessions showing the three participants wearing HoloLens 2 while performing the AR mediated Point and Pick interaction with the UR3e collaborative robot in the laboratory workstation. 50

5.4 System Usability Scale scores obtained from the three participants, with a mean SUS score of 80.0. 51

5.5 Mean discomfort intensity reported by participants on a 0–3 scale. The overall average was 0.396, indicating low discomfort during short evaluation sessions. 54

A.1 Detailed runtime architecture of the proposed AR mediated control system. The HoloLens 2 application communicates through a ROS# / RosSharp WebSocket client with the ROS Noetic workstation. The workstation hosts the communication bridge, robot driver, optional planning components, application level Python nodes, and the gripper actuation path for the UR3e and RG6 hardware. 73

List of Tables

5.1 Participant information for the exploratory user evaluation. 49

5.2 System Usability Scale results from the exploratory user evaluation. 51

5.3 Summary of qualitative questionnaire findings from the exploratory user evaluation. 53

5.4 Raw cyber sickness and discomfort results. 54

B.1 Core ROS topics used in the project. 76

C.1 Manipulation and motion parameters. 79

C.2 UI and interaction parameters. 80

C.3 Gripper parameters. 80

1. Introduction

1.1 Background and Motivation

Collaborative robots, or cobots, have become increasingly relevant in manufacturing environments where flexibility, rapid reconfiguration, and direct human robot cooperation are not peripheral requirements but central ones. Unlike conventional industrial robots, which operate inside fenced and segregated cells, cobots are built to share the workspace with human operators and to support tasks in which human adaptability and robotic repeatability must coexist. This makes them particularly well suited to workstation level applications such as kitting, material handling, structured pick and place, and collaborative assembly support.

In these contexts, productivity does not depend only on how fast or how accurately the robot moves. It also depends on how effectively the operator can supervise the system, understand robot intent, adjust task execution, and interact with the robot without interrupting the workflow. In many manufacturing scenarios, especially those involving product variability, small batch production, or semi structured manual activities, traditional robot interfaces remain a practical limitation. Teach pendants, external screens, and programming oriented tools often require the operator to shift attention away from the workstation, increase setup time, and rely on skills that are not always available on the shop floor. As a result, the potential benefits of collaborative robotics are often reduced by the complexity of interaction and reconfiguration rather than by any shortcoming in the robot itself.

This issue is particularly acute in collaborative assembly and workstation assistance, where the operator must remain focused on the physical process while simultaneously understanding the robot's state, intended action, and task progression. In such settings, an interface that is technically functional but difficult to interpret or cumbersome to use can negatively affect efficiency, usability, and trust. The problem of human robot interaction in manufacturing is therefore not only a technical one; it is equally an operational one.

Augmented Reality (AR) and Mixed Reality (MR) offer a promising alternative to conventional interaction paradigms. By embedding digital information directly into the physical workspace, AR can present robot state, target positions, task instructions, and motion related feedback *in situ*, reducing the need to switch attention between the workstation and an external control interface. This is particularly valuable in manufacturing, where timely understanding of robot intent can improve supervision, reduce operator

uncertainty, and support more effective collaboration.

The value of AR becomes even more significant when combined with Natural User Interfaces (NUIs) such as gesture and voice interaction. Rather than relying on low level programming or conventional manual controls, the operator can interact through more intuitive and task oriented actions, such as pointing to a target, indicating a location in space, or issuing a command verbally. In principle, this lowers the entry barrier for robot interaction, improves accessibility for non expert users, and supports more natural coordination between human and robot at workstation level.

Motivated by these considerations, this thesis investigates the use of AR as an interaction layer for collaborative robotic manipulation in a manufacturing oriented workstation scenario, with particular attention to usability, deployment constraints, and the operational needs of collaborative assembly support. The validation task implemented in this work is based on a structured pick and place workflow, conceived as a simplified assembly support scenario representative of collaborative workstation operations in manufacturing.

1.2 Research Problem

Although AR and MR technologies show strong potential for Human Robot Interaction (HRI), their integration into real time collaborative robot control remains challenging, especially when the target context is a manufacturing workstation rather than a laboratory demonstration.

The core problem is the following: collaborative robots are increasingly capable of supporting flexible manufacturing and assembly tasks, but the interfaces used to supervise, command, and reconfigure them are often not sufficiently intuitive, contextual, or operator centered for workstation level use. In practice, this creates friction between the promise of collaborative automation and the realities of day to day industrial operation. When the operator must rely on detached interfaces, specialized programming knowledge, or interaction workflows that interrupt the physical task, the overall system becomes less agile and less usable than it should be.

AR mediated control offers a possible response by relocating interaction into the workspace itself. However, this approach introduces its own challenges that must be addressed before it can be considered credible for real collaborative applications. The first is spatial registration: virtual content must remain accurately aligned with the physical robot and its surroundings. Even small drift or misalignment can produce meaningful positioning errors during manipulation and erode operator trust in the interface. The second is communication responsiveness: robot control requires predictable and sufficiently low latency data exchange between the AR device and the robot control stack. Delays or

instability affect both usability and execution reliability.

A further challenge concerns interaction design. In collaborative manufacturing tasks, the interface must be efficient enough for fluent task execution while also being robust enough to prevent unintended commands. Rather than relying on a rigid multi step interaction pipeline, the system must support a command driven interaction model in which the operator expresses intent through natural inputs and receives immediate visual feedback within the workspace. This feedback must be sufficient to allow the operator to confirm the selected target before explicitly triggering execution through a deliberate command.

Finally, deployment constraints must be taken seriously. A system that works as a proof of concept may still be difficult to reproduce or operate reliably in practice if it depends on unstable tracking, excessive device workload, fragile communication, or poorly structured setup procedures. For AR mediated robot control to have practical value, it must be studied not only as an interface concept, but as an integrated system subject to technical, ergonomic, and operational constraints.

In response to these issues, this thesis develops an AR mediated control system for a UR3e collaborative robot equipped with an OnRobot gripper, using HoloLens 2 as the operator interface and ROS Noetic as the robot side middleware. The proposed interaction model is based on a *Point and Pick* paradigm in which the operator designates a target in three dimensional space and the system interprets that input as the basis for a predefined manipulation behavior, such as a grasp and place operation. The interaction relies on explicit user commands combined with continuous visual feedback to ensure clarity and control over execution. This interaction is studied through a structured pick and place task intended as a simplified collaborative assembly support scenario at workstation level, positioning the work within a manufacturing oriented context.

1.3 Objectives and Research Questions

The main objective of this thesis is to design, implement, and evaluate an AR mediated interaction system for a collaborative robot cell that is intuitive for the operator, technically feasible, and relevant to manufacturing oriented collaborative tasks.

More specifically, the work pursues four objectives. The first is to develop an AR interface on HoloLens 2, built in Unity with MRTK, capable of supporting gesture based, voice based, and Point and Pick interaction. The second is to integrate this interface with a ROS Noetic control architecture that manages UR3e motion execution and gripper actuation through ROS# and WebSocket based communication. The third is to define a calibration and registration workflow that maintains consistent alignment between the

AR environment and the robot reference frame. The fourth is to evaluate the system through representative manipulation tasks and structured user feedback, assessing usability, predictability, and the practical constraints that affect real deployment.

These objectives lead to three research questions:

- **RQ1**: To what extent can an AR mediated Point and Pick interface support intuitive and understandable operator interaction in collaborative manipulation tasks?
- **RQ2**: How do spatial registration drift and device side performance constraints affect the practical deployability of the proposed system in a structured laboratory workstation?
- **RQ2b**: To what extent does the ROS#/WebSocket communication architecture maintain sufficient responsiveness for supervised manipulation tasks, and what are its observable failure modes?
- **RQ3**: What benefits and limitations does the proposed interaction approach show when considered in relation to manufacturing oriented workstation tasks, in particular simplified collaborative assembly support and structured pick and place scenarios?

1.4 Scope and Assumptions

The system described in this thesis centers on a single collaborative workstation: a UR3e robot with an OnRobot gripper, controlled through ROS Noetic and accessed via HoloLens 2.

The focus is on interaction design and system integration rather than advanced perception. Grasp poses are generated through raycast based targeting combined with task specific heuristics and preset definitions. The system assumes that object locations and destinations remain reasonably consistent, which is a realistic constraint in structured industrial scenarios such as kitting, workstation material handling, and assembly support, where relevant positions are typically predefined or vary within limited bounds. The implemented validation task is therefore not intended to represent unconstrained free manipulation, but rather a simplified and structured assembly support scenario in which operator supervision and predictable execution are central.

The evaluation is conducted in a controlled laboratory setting. The results are informative about feasibility, usability, and deployment implications under structured conditions, but they do not constitute full industrial validation. Deployment in a real production environment would require additional verification under more variable conditions, including

changing lighting, cluttered workspaces, multiple operators, and less predictable network behavior.

1.5 Thesis Structure

The remainder of this thesis is organized as follows.

Chapter 2 surveys the relevant literature on collaborative robotics in manufacturing, workstation level human robot interaction, AR/MR interfaces for industrial collaboration, multimodal interaction techniques, ROS based control architectures, and evaluation approaches for AR mediated systems. Particular attention is given to collaborative assembly support, usability, deployment constraints, and the research gap addressed by this work.

Chapter 3 presents the research methodology, covering the experimental design, development stack, system integration workflow, case study definition, evaluation protocol, and data collection and analysis strategy.

Chapter 4 describes the system implementation in detail: the AR application, the ROS based runtime architecture, workspace configuration, execution logic, calibration and frame handling, gripper integration, and deployment workflow.

Chapter 5 reports the results of the case study based validation and the exploratory user centered evaluation, with attention to execution behavior, usability observations, runtime constraints, and error recovery.

Chapter 6 discusses these results from both a technical and a Management Engineering perspective, examining their implications for collaborative workstation design, process stability, safety governance, usability, and deployment readiness in manufacturing oriented scenarios.

Chapter 7 concludes the thesis with a summary of the main contributions, a discussion of industrial applicability and limitations, and directions for future work.

2. Literature Review

2.1 Collaborative Robotics in Manufacturing and Assembly

The term "cobot" was coined by Colgate and Peshkin in their 1996 work on robots intended to share workspace with human operators, and the concept has since grown from a niche safety innovation into a mainstream manufacturing technology [1]. What distinguishes collaborative robots from conventional industrial ones is not simply a safety certification or the absence of a protective cage, but a fundamentally different design intent: they are built to amplify, rather than replace, human activity in shared workspaces.

Matheson et al. conducted a systematic review of collaborative robotics in manufacturing covering 35 case studies published between 2009 and 2018 [2]. Their analysis classified the literature into three topic categories productivity, safety, and human robot interaction and found that while intrinsic safety mechanisms were the most studied theme, a growing number of studies were shifting attention toward interaction quality and usability. Importantly, they noted that the barrier to industry adoption was not primarily technical: many manufacturing organisations already had access to capable hardware but lacked interaction workflows suited to non expert operators on the shop floor. This tension between technical capability and practical usability is a thread that runs through most subsequent work in the field.

Keshvarparast et al. reviewed cobot deployment in manufacturing from an Industry 4.0 perspective and highlighted that flexible automation particularly in assembly, kitting, and material handling is where cobots provide the most unambiguous value [3]. Their survey pointed out that structured workstations, where object locations and sequences are semi standardised, represent the most realistic near term deployment context, because the control demands are manageable while the potential gains from robotic assistance are still substantial.

The Industry 5.0 literature sharpens this picture further. Dhanda et al. reviewed human robot collaboration in the context of Industry 5.0 and argued that the practical benchmark for a cobot system is no longer whether it can move safely, but whether human workers can remain effectively integrated into the process rather than being sidelined by it [4]. They identified transparency, operator trust, and recovery from deviation as the three most critical interaction level properties, and observed that these are systematically

underdeveloped in current systems compared to motion control and safety.

In assembly specifically, Cao et al. examined human robot collaboration in welding and assembly operations and identified a recurring problem: robot behaviour in shared tasks is often difficult for operators to interpret at a glance [5]. This is not merely inconvenient in assembly workstations where timing, part placement, and sequence compliance matter, an operator who cannot quickly read robot intent must compensate by slowing down or monitoring the robot continuously, which undermines the collaboration's productivity rationale. Kheirabadi et al. echoed this concern in their work on assembly line cobots, finding that the quality of human robot coordination in structured tasks depended more on the clarity of feedback mechanisms than on raw execution speed [6].

Ajoudani et al. provided an influential review of progress and prospects in human robot collaboration [7], synthesising work on bidirectional control interfaces, shared autonomy, and physical interaction. Their central argument was that effective collaboration depends not on robot autonomy as such, but on real time adaptability and clear role negotiation between human and machine. This framing is directly relevant to the present work: the Point and Pick paradigm is not an exercise in automation but an attempt to create a structured and negotiated form of interaction in which the operator retains supervisory authority while delegating execution.

Finally, Petzoldt et al. examined task allocation in human robot collaborative systems and found that operators perform most reliably when the division of labour is explicit and predictable [8]. Ambiguity in who is responsible for what step, and when, introduces hesitation and error recovery cost. This observation supports design choices made in this thesis particularly the use of explicit execution commands and preset based workspace modelling as mechanisms that make the collaboration contract between operator and robot unambiguous.

2.2 Human Robot Interaction in Collaborative Workstations

Goodrich and Schultz's survey of human robot interaction, while focused on a broad range of domains, identified transparency as the foundational property for effective collaboration [9]. Operators need to understand not only what the robot is doing, but what it intends to do and why, if they are to maintain situational awareness and intervene appropriately when something goes wrong. In a manufacturing workstation, where operator attention is already distributed across parts, tools, and sequence constraints, the cost of misreading robot intent can propagate quickly into rework or process interruption.

Ajoudani et al. [7] and Rosen et al. [10] both converge on this point from different directions. Ajoudani's framework of shared autonomy emphasises that the robot must respond to inferred human intent while also signalling its own, creating a genuine bidirectional communication channel. Rosen et al. studied this communication problem specifically for robot arm motion in shared workspaces and found that augmented reality overlays outperformed conventional physical signalling such as indicator lights and robot posture in conveying motion intent, particularly for non expert users unfamiliar with the robot's programmed behaviour.

Fiorentino et al. argued that the gap between operator intention and robot behaviour is the central design problem in industrial HRI [11]. They analysed several commercial and research interfaces and found that conventional tools such as teach pendants and programming environments are well suited for structured offline programming, but poorly suited to the in task adjustment that collaborative workstations require. Their main recommendation was to develop interfaces that support in situ control interaction that happens at the point of work rather than at a detached control station.

Salami et al. reviewed advanced collaborative workstations with particular attention to operator workload and situational awareness [12]. They found that interactions requiring the operator to shift attention from the physical task to a screen based interface even briefly reliably increased task completion time and error rate. This provides direct empirical motivation for the AR based approach developed in this thesis.

2.3 Augmented and Mixed Reality for Industrial Human Robot Collaboration

The case for AR in industrial HRI rests on a simple functional argument: if information about robot state, intent, and task context can be embedded directly in the operator's visual field rather than on an external screen, the attentional cost of accessing that information approaches zero. The empirical question is whether this theoretical advantage translates into measurable operational benefits in real task settings.

Chan et al. addressed this question directly in a study of AR interfaces for human robot teams in high value composite manufacturing [13]. In a within subjects study with 26 participants performing a physically demanding pleating task, they compared a joystick interface against two AR configurations. The AR interface reduced physical demand and task completion time while increasing robot utilisation. However, the AR interface was rated as more novel but less dependable than the joystick an important finding for deployment planning, since perceived dependability affects whether operators choose to use an interface voluntarily. The study also showed that user autonomy whether operators

chose to collaborate or work alone significantly influenced perceived usability, suggesting that interface design cannot be separated from broader questions of task structure and operator agency.

Rosen et al. investigated specifically how AR can communicate robot arm motion intent in shared workspaces [10]. They developed several explicit and implicit AR visualisation designs and found that several designs significantly improved task efficiency over a baseline in which users relied only on the physical robot's movements for intent inference. Critically, they also found trade offs between intent clarity and the user's perception of the robot as a collaborative partner: designs that were maximally explicit sometimes felt mechanical or directive rather than collaborative. This is a design tension that the present work addresses through explicit execution commands: the operator retains full authority over whether and when to trigger robot motion, with no motion occurring without a deliberate voice or gesture command.

Fiorentino et al. [11] and Coronado et al. [14] have both reviewed AR interfaces for robot interaction more broadly. Coronado et al. identified spatial registration accuracy and communication latency as the two most consistently cited technical barriers in deployed systems, noting that even modest alignment errors on the order of a few centimetres can make AR overlays appear unreliable and erode operator confidence. This finding directly motivates the calibration and reanchoring procedures described in this thesis.

Wulff et al. investigated the usability of AR assisted commissioning of industrial robot programs with 31 participants performing a robot commissioning task in a manufacturing mock up [15]. They analysed usability across the three ISO 9241-11 dimensions of effectiveness, efficiency, and satisfaction. Nine out of ten participants in the initial feasibility study completed the task successfully using the AR application, and the followup study with 31 participants produced SUS scores in the "acceptable" range. The study also found that the relationship between task performance and perceived usability was non linear: some participants achieved high completion rates while rating usability poorly, because they found the interaction model confusing even when they eventually succeeded. Wulff et al. concluded that AR feasibility is a necessary but not sufficient condition for AR usability in industrial robot applications.

2.4 HoloLens 2 as an Interaction Platform for Robotics

Microsoft HoloLens 2 occupies a distinctive position among head mounted MR devices for industrial robotics research: it combines untethered operation, articulated hand tracking, voice input, and direct spatial mapping with a development ecosystem (Unity, MRTK) that is well suited to iterative interface design. Most published systems using HoloLens 2

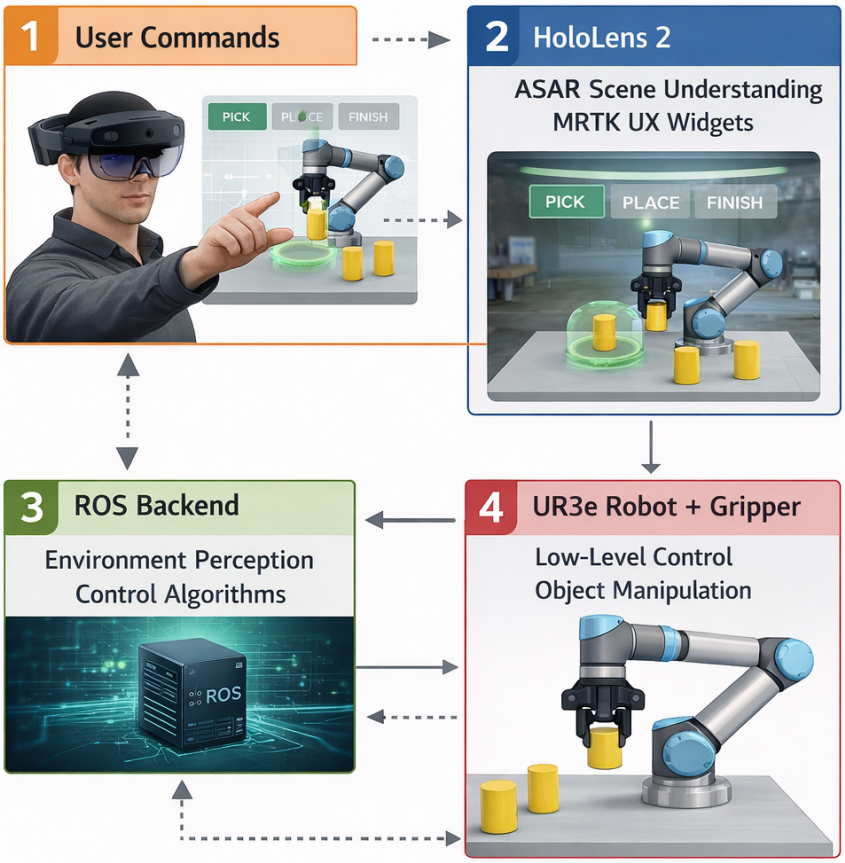

Figure 2: AR-Mediated Human-Robot Interaction Using HoloLens 2

Figure 2.1: Conceptual representation of spatial registration between the AR coordinate system and the robot reference frame. Accurate alignment ensures that virtual targets correspond to correct physical locations in the workspace.

for robot control share this combination of Unity based development and ROS based robot integration, making the present work part of a well established and comparable research tradition.

Coronado et al. [14] reviewed MR based frameworks for robot interaction and noted that HoloLens 2 was by far the most commonly used HMD in recent HRI research, partly because of its spatial mapping capabilities and partly because MRTK abstracts gesture and gaze interaction into standardised primitives that reduce implementation overhead. Their review identified the HoloLens 2/Unity/ROS# stack as the dominant architecture for this category of application, which is the same stack used in the present work.

The hardware architecture of HoloLens 2 is important to understand because it directly constrains what is possible. The device uses four environment cameras and an IMU for inside out head tracking, a structured light sensor for depth mapping, and short throw IR sensors for hand tracking. The display is a waveguide optical system with a roughly 52 degree diagonal field of view. These characteristics have two implications for HRI design. First, hand tracking quality degrades under partial occlusion and at the periphery of the tracking volume a known limitation that affects gesture based control reliability. Second, device side compute is limited relative to a workstation; rendering complex overlays alongside active tracking creates processing competition that can degrade responsiveness under high interaction load.

Chan et al. [13] reported that in their composite manufacturing study, HMD based AR was rated as more novel but less dependable than joystick control, a finding consistent with participants experiencing occasional tracking or overlay instability. A subsequent analysis of HoloLens 2 eye tracking signal quality by Aziz and Komogortsev found that eye movement data from the device was often uncalibrated at session start, which has implications for gaze dependent interaction designs, though this is less critical for the hand tracking and voice primary interaction model used in this thesis.

Kocian et al. documented the practical challenges of deploying HoloLens based robot control in real environments, noting that tracking performance degrades in environments with low visual texture, uniform lighting, or rapid changes in scene geometry [16]. They also observed that the field of view limitation operators cannot see virtual content that extends beyond roughly 26 degrees from the optical axis is a practical constraint in workstations where the robot operates across a wide spatial range. Nenna et al. studied HoloLens use for industrial virtualisation tasks and found that even short sessions of 20–30 minutes produced measurable visual fatigue in a significant proportion of participants, with symptom severity correlating with overlay complexity and interaction frequency [17]. These findings inform the interface design choices made in this thesis, particularly the use of lightweight, time limited AR overlays and the avoidance of dense or persistent visualisation that would increase rendering load.

2.5 Multimodal Interaction for Robot Control

Combining multiple input channels for robot control is motivated by a practical argument: no single modality is both spatially expressive and unambiguously deliberate. Gesture is spatially rich but prone to accidental triggering; voice is explicit and hard to trigger accidentally but lacks spatial reference; pointing combines spatial expression with low ambiguity but requires reliable tracking. Multimodal systems address these limitations by assigning each modality to the task it handles best.

Fiorentino et al. [11] argued for this division of responsibility directly, recommending that spatial intent be expressed through gesture or pointing while safety critical commands be reserved for voice or explicit touch, to prevent accidental execution. Coronado et al. reviewed gesture and voice combinations in robot teleoperation and noted that the robustness benefit of combining modalities was most pronounced in tasks where execution had significant consequences precisely because voice confirmation is hard to trigger inadvertently [14].

Empirical evidence for this design logic has grown recently. Zhang et al. conducted a within subject user study with 42 participants performing a pick and place pattern matching task under two conditions: gesture only and combined voice plus gesture [18]. Gesture only provided more reliable and efficient control for time critical actions, while voice plus gesture added flexibility but also introduced latency and recognition variability that increased workload. Their conclusion was that multimodality is most valuable as an adaptive strategy: additional modalities should be added when they address a specific limitation of the primary channel, not as a general principle that more is better. The present work follows this logic: gesture handles spatial selection while voice handles explicit execution triggering, each assigned to its functional strength.

Chan et al.'s manufacturing study also bears on this question [13]. They found that the AR interface in their HMD condition yielded the lowest physical and mental demand scores on the NASA TLX, though the differences did not reach statistical significance compared to joystick conditions. What this suggests is that the benefit of AR plus gesture over conventional input is real but context dependent, and may require a specific combination of task structure and interface design to become unambiguous.

The Point and Pick paradigm used in this thesis is a task level abstraction of this multimodal logic. Rather than specifying robot motion at the joint or waypoint level, the operator designates a target in space a single act that encodes both spatial intent and task identity and the system translates that designation into a structured manipulation sequence. Rosen et al. [10] described a similar abstraction in their work on robot arm intent communication, finding that users were more confident and accurate when the

interface allowed them to express goals rather than motion commands. This goal level interaction model is less susceptible to accumulated pointing error and more legible to non expert users, because it corresponds more directly to how people describe task intent in natural language.

2.6 ROS Based Communication and Control Architectures

The Robot Operating System (ROS), introduced by Quigley et al. in 2009 as an open source middleware framework, has become the de facto standard for robot software integration in both research and industrial prototyping [19]. Its core architecture topics for publish subscribe messaging, services for request response calls, actions for long running tasks, and TF2 for coordinate frame management provides a modular and inspectable runtime that makes complex multi component systems significantly easier to develop and debug than monolithic alternatives. In practice, the ROS ecosystem also provides an extensive library of packages for motion planning (MoveIt), sensor processing, and robot drivers (including the Universal Robots ROS Driver), which reduces the implementation overhead for systems like the one described in this thesis.

For AR applications developed in Unity, connecting to a ROS backend requires a bridge, because HoloLens 2 runs on Universal Windows Platform (UWP) and cannot host a native ROS runtime. The standard solution is ROS# (RosSharp), a C# library that communicates with a `rosbridge_websocket` server on the ROS side using JSON serialisation over WebSockets. Coronado et al. identified this architecture as the most widely adopted approach for HoloLens based robot control, noting that its main trade off is serialisation overhead: JSON over WebSocket is less efficient than native ROS inter process communication, but the resulting latency is acceptable for task level supervisory control where execution is gated by explicit user commands rather than requiring sub millisecond response [14]. Kocian et al. used an equivalent architecture and observed that communication instability manifesting as dropped messages or delayed telemetry had a direct negative effect on perceived usability, because operators lost confidence in the accuracy of AR feedback when it appeared to lag behind the physical robot [16].

The Universal Robots ROS Driver, which is used in this thesis for UR3e integration, provides RTDE (Real Time Data Exchange) based communication with the robot controller. RTDE is UR's deterministic interface for state streaming and trajectory execution, designed to operate at up to 500 Hz for state data and 125 Hz for set point commands. When combined with MoveIt for trajectory planning, the driver provides a complete control stack from high level goal specification to hardware execution, with feasibility checking and

collision awareness integrated at the planning layer. This is the integration configuration used in the present work and is consistent with configurations reported in comparable UR based research systems [14, 16].

2.7 Technical, Human Factor, and Deployment Challenges

The gap between a working AR robot interface prototype and a system that can be deployed reliably in a real workstation is wider than most published systems acknowledge. Several categories of challenge are consistently identified across the literature.

Spatial registration. AR overlays must remain accurately aligned with physical objects across the duration of a session. Drift in inside out tracking caused by motion of the headset, changes in environmental lighting, or loss of tracking features accumulates over time and can make virtual targets appear displaced from their physical counterparts. Rosen et al. and Coronado et al. both noted that spatial misalignment is the most commonly cited reason for AR feedback becoming unreliable in manipulation tasks [10, 14]. Even centimetre level drift is significant in a robot grasp context, where approach trajectories must be accurate to within a few millimetres. The present system addresses this through periodic re anchoring and a spatial tolerance threshold that prompts re registration before precision critical operations.

Communication latency. In interactive systems, latency above roughly 100–150 ms disrupts the sense that a virtual representation is responding to physical action, degrading operator trust [16]. In a pick and place workflow where execution is explicitly triggered, hard real-time constraints are less critical than in teleoperation. However, the spatial overlays displayed on HoloLens must remain synchronised with robot state, since operators rely on the accuracy of the AR environment to correctly designate targets and interpret system feedback. If the AR display lags noticeably behind physical reality, pointing accuracy and spatial trust degrade.

Device side compute saturation. HoloLens 2 has limited processing headroom, and complex interaction events rapid gesture sequences, dense spatial overlays, high frequency telemetry visualisation can saturate the device and produce temporary unresponsiveness. This is not merely a comfort issue. An unresponsive interface is a safety concern in robot control: operators must be able to issue stop or cancel commands at any moment, and interface lag can make this impossible. Nenna et al. documented processing degradation under high rendering load in HoloLens based industrial applications and recommended aggressive rate limiting of input and visualisation streams as the primary mitigation [17]. The present system implements this through input throttling at 20 Hz and time limited

overlay markers.

Ergonomic burden. Prolonged HMD use introduces visual fatigue, oculomotor strain, and weight related discomfort. Nenna et al. found measurable fatigue after sessions of 20–30 minutes, correlating with overlay complexity and interaction frequency [17]. Chan et al. noted that comfort and perceived safety influenced whether participants chose to use the AR interface voluntarily, even when objective task performance was comparable [13]. For deployment in manufacturing workstations where sessions may extend across a full shift, ergonomic constraints are not secondary: they define the practical ceiling on how much AR interaction the operator can sustain.

Deployment reproducibility. Several studies have noted a gap between the effort required to set up an AR robot system for a demonstration and the effort required to do so reliably across repeated sessions. Keshvarparast et al. identified calibration drift, network configuration variability, and platform specific build issues as the most common causes of deployment failure in industrial AR prototypes [3]. These issues are often treated as footnotes in published work despite being primary determinants of whether a system can be reproduced by someone other than its developer.

2.8 Evaluation Approaches for AR Mediated Human Robot Systems

The standard evaluation toolkit for AR based HRI systems combines technical performance metrics (task completion rate, execution time, error rate) with subjective measures of usability and user experience, typically using the System Usability Scale (SUS) or the NASA Task Load Index (NASA-TLX). Chan et al. used both in their manufacturing study, which is one of the most methodologically thorough evaluations in the field [13]. Their study showed that objective performance gains (reduced task time, lower physical demand) were not always reflected in subjective usability scores, and that the direction of discrepancy depended on prior interface familiarity. This finding underlines a known limitation of SUS only evaluation: the scale measures perceived usability, which is influenced by prior expectations, and may not capture operational benefits that are real but subtle.

Wulff et al. [15] similarly found that their AR robot commissioning interface produced completion rates above 90% while SUS scores in the follow up study landed in the "acceptable" range, reflecting that participants found the interface manageable but not polished. They recommended combining SUS with task specific questions about clarity of robot intent and confidence during interaction, since these dimensions are not well captured by the generic SUS items.

Puspanathan et al. studied acceptance of collaborative robot interfaces and found that learnability and perceived safety were stronger predictors of acceptance than overall usability scores [20]. This is relevant for manufacturing contexts, where operators may be willing to tolerate an imperfect interface if they trust it and believe they can recover from errors. Nenna et al. evaluated discomfort specifically in HoloLens based industrial applications, finding that symptom profiles eye strain, fullness of head, blurred vision were mild for short sessions but became significant beyond 30 minutes [17]. They recommended including a discomfort questionnaire alongside SUS in any evaluation of head mounted AR for industrial use.

The evaluation in this thesis follows these recommendations: SUS is used for overall usability, supplemented by task specific questions on clarity of robot intent and interaction confidence, and a short cybersickness and discomfort instrument. The rationale for this combination is grounded in the published findings above rather than in arbitrary selection of standard tools.

2.9 Research Gap and Positioning of the Present Work

The literature reviewed in this chapter converges on a clear picture of both the potential and the limitations of AR mediated robot control in manufacturing. On the potential side: AR can reduce attentional cost, improve robot intent clarity, and support more natural forms of task specification for non expert operators. On the limitations side: spatial registration drift, device side performance constraints, communication latency, and ergonomic burden are recurring barriers that are insufficiently addressed in most published systems.

Several specific gaps motivate the present work. First, the majority of existing studies use AR primarily for teleoperation or robot programming tasks that are fundamentally different from operator supervised collaborative manipulation at workstation level. Chan et al. [13] and Rosen et al. [10] are notable exceptions, but neither work addresses the specific combination of spatial designation, preset-based target confirmation, and explicit command gating that characterises the supervisory interaction model implemented in this thesis. Second, most published AR robot systems are evaluated as proofs of concept without systematic attention to the deployment conditions that determine whether the approach is reproducible and operational across sessions. Keshvarparast et al. [3] and Dhanda et al. [4] both identify this as an underdeveloped area. Third, evaluation quality is uneven: many studies argue that AR improves intuitiveness or transparency without documenting these claims through structured user assessment, a gap that Wulff et al. [15]

and Puspanathan et al. [20] explicitly flag.

What remains underdeveloped, and what this thesis directly addresses, is the combination of: (i) a manufacturing oriented task framing (simplified collaborative assembly support via structured pick and place); (ii) a spatially grounded, multimodal interaction model in which the operator designates targets in three dimensional space rather than programming trajectories or driving teleoperation; (iii) a command-gated execution model in which the operator explicitly triggers motion through a deliberate voice or gesture command rather than through continuous manual control; (iv) end to end ROS integration with an industrial collaborative robot using the Universal Robots driver stack; and (v) structured evaluation of both technical behaviour and user perceived usability under controlled workstation conditions.

This thesis does not claim to solve all the challenges identified in the literature. What it contributes is a focused investigation of whether this specific combination of design choices produces a system that is technically feasible, interactionally understandable, and operationally meaningful for structured collaborative workstation tasks a question that the existing literature poses but does not yet answer.

3. Methodology

3.1 Research Design and Evaluation Strategy

The aim of this research is to design, implement, and evaluate an AR interface for controlling a UR3e collaborative robot through Microsoft HoloLens 2 in a manufacturing oriented collaborative workstation scenario. The system supports Human Robot Collaboration (HRC) through multimodal interaction, namely hand gestures, voice commands, and a Point and Pick model in which the operator designates a target in three dimensional space and the system translates that designation into a structured robot action.

The study is positioned as an applied experimental investigation. Its purpose is not to provide a large scale industrial benchmark, but to assess whether the proposed interaction architecture is technically feasible, understandable to the operator, and sufficiently robust for representative collaborative manipulation tasks related to workstation assistance, collaborative assembly support, and structured pick and place operations.

The central assumption guiding the work is that AR mediated interaction reduces operational friction in collaborative robot supervision by embedding robot related information directly into the workspace. Whether this holds under the specific constraints of real implementation spatial alignment, communication responsiveness, device side limitations, and operator comprehension is precisely what the methodology tests.

The methodology is organized into four phases. First, the AR interface is developed on HoloLens 2 using Unity and MRTK, with attention to multimodal interaction and visual feedback. Second, the UR3e is integrated through a ROS Noetic backend covering calibration, driver configuration, communication, and gripper actuation. Third, a representative case study is defined and executed in a structured laboratory workstation. Fourth, system behavior is assessed through logs, observation, and structured user feedback to identify strengths, limitations, and practical deployment constraints.

3.2 Development Stack and Tools

The system is distributed across two main components: the AR interface on HoloLens 2 and the robot control stack on a separate workstation. Each has a distinct role, and communication between them runs through a dedicated middleware bridge.

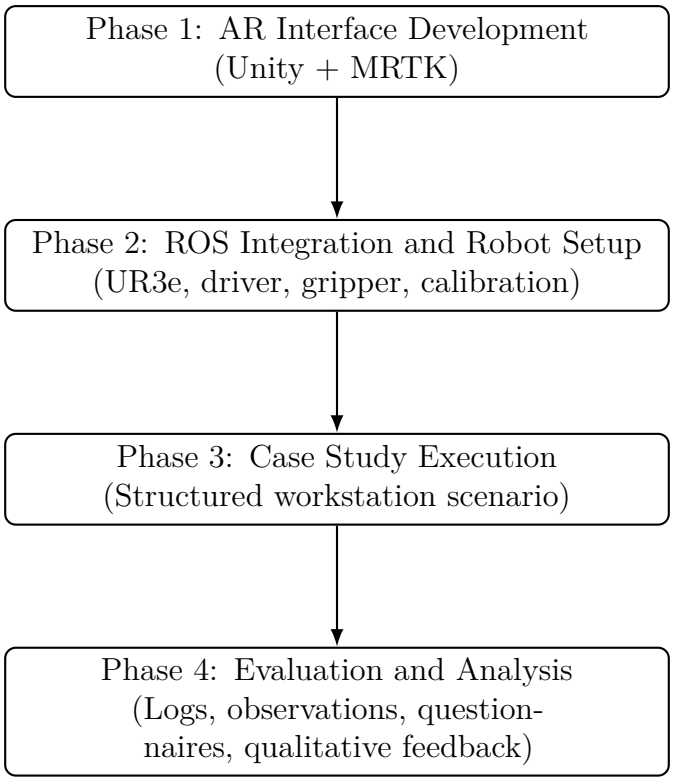

Figure 3.1: Methodological workflow of the proposed AR mediated control system, illustrating the four main phases from development to exploratory assessment.

3.2.1 HoloLens 2 AR Interface (Unity/MRTK)

The AR interface constitutes the primary interaction layer of the system and is implemented on Microsoft HoloLens 2 using the Unity game engine and the Mixed Reality Toolkit (MRTK). This component is responsible for capturing operator input, managing spatial interaction, and providing in situ visualization of robot related information directly within the workspace.

The development environment is based on Unity 2021.3 LTS, combined with MRTK version 2.8.x, and deployed as a Universal Windows Platform (UWP) application compiled through Visual Studio Community 2019. This configuration was selected to ensure compatibility with HoloLens 2 while maintaining long term stability and reproducibility across development iterations. In particular, the use of a Long Term Support (LTS) version of Unity reduces the likelihood of breaking changes during the development lifecycle, which is critical in a system where interaction behavior must remain consistent across experimental sessions.

Unity provides the runtime environment for scene management, rendering, and application logic, while MRTK abstracts low level interaction mechanisms into higher level primitives suitable for Human Robot Interaction. Through MRTK, the system leverages articulated hand tracking, ray based spatial input, spatial mapping, and integrated speech

recognition. These capabilities form the basis of the multimodal interaction approach adopted in this work.

From a design perspective, the choice of Unity and MRTK reflects a balance between interaction richness and implementation efficiency. Unity enables rapid prototyping and iterative refinement of the interface, while MRTK provides standardized components that reduce development complexity and improve consistency in interaction behavior. This is particularly relevant in a research context, where multiple iterations of interaction design are often required before arriving at a stable configuration.

The AR interface performs three tightly coupled functions. First, it captures operator intent through spatial input derived from hand tracking, complemented by discrete gesture and voice events. Second, it expresses this input within a consistent spatial reference by applying the registration between the Unity world coordinate system and the robot frame. Third, it provides visual feedback directly in the workspace, allowing the operator to interpret system behavior without relying on external displays.

To maintain responsiveness on the HoloLens device, the application implements controlled data publishing and input throttling. Continuous spatial streams, such as pointer pose updates, are rate limited to approximately 20 Hz. This choice represents a compromise between interaction smoothness and computational load, ensuring that the device remains responsive even during extended interaction sessions.

A further design consideration concerns the separation between continuous and discrete interaction channels. Continuous input, such as hand pose tracking, is used for spatial designation, while discrete inputs, particularly voice commands, are used for explicit execution triggers and control actions. This separation reduces ambiguity and improves robustness, especially under conditions where hand tracking quality may degrade.

Finally, the AR interface is designed as a thin client within the overall system architecture. It does not perform motion planning or low level robot control locally. Instead, it structures and transmits operator intent to the ROS backend, which is responsible for execution. This separation reduces the computational burden on the HoloLens device and ensures that control logic remains centralized, improving both system maintainability and runtime reliability.

3.2.2 ROS Backend (Ubuntu 20.04 / ROS Noetic)

The robot side backend runs on Ubuntu 20.04 LTS with ROS Noetic Ninjemys. ROS Noetic is the final long term support distribution of ROS 1 and provides the most mature environment for systems combining the Universal Robots ROS Driver, `rosbridge_websocket`, MoveIt based motion validation, and custom Python orchestration nodes. All of these

components have stable, tested releases for Noetic with documented launch procedures and known good configuration patterns.

A migration to ROS 2 was attempted during the initial development phase. In practice, the combination of the Universal Robots ROS driver, the ROS# WebSocket bridge, and the rosbridge suite produced persistent instability: node communication was unreliable, the bridge connection dropped frequently under moderate message load, and several driver behaviours differed from the documented ROS 1 equivalents in ways that required non-trivial debugging without clear resolution. After a sustained period of integration attempts, ROS 2 was set aside. Ubuntu 20.04 with ROS Noetic was therefore retained as the deployment environment, supporting consistent backend behaviour across experimental sessions. Observed limitations such as latency spikes or drift accumulation can consequently be attributed to the implemented system rather than to an unstable software environment.

3.2.3 Universal Robots ROS Driver

Motion execution and state monitoring are handled through the Universal Robots ROS driver, which provides a direct interface between ROS and the UR3e controller. The driver used in this work corresponds to the official `Universal_Robots_ROS_Driver` package (ROS Noetic compatible branch), which is actively maintained and widely adopted for industrial and research applications involving UR robots.

The driver exposes a set of ROS interfaces for controlling the robot, including joint state publishing, trajectory execution through ROS controllers, and access to the robot dashboard services. Internally, it relies on the Real Time Data Exchange (RTDE) interface provided by Universal Robots, enabling deterministic communication for motion control and state feedback. This architecture allows the system to execute planned trajectories while continuously monitoring robot state, which is essential for supervised interaction scenarios such as the one implemented in this thesis.

Integration is performed through the standard bring up procedure (`ur3e_bringup.launch`), which initializes communication with the robot, loads the appropriate controllers, and applies the robot specific kinematic calibration parameters extracted during setup. Using the correct calibration file is critical, since even small geometric deviations between the nominal and actual robot model would otherwise propagate into positioning errors during manipulation tasks.

The driver also provides compatibility with MoveIt, enabling optional trajectory planning, feasibility checking, and collision aware execution. Although MoveIt is not strictly required for the Point and Pick pipeline, it was used during development and testing to verify motion feasibility and improve robustness of execution.

The official repository for the driver is available at:

`https://github.com/UniversalRobots/Universal_Robots_ROS_Driver`

This repository includes documentation, launch files, and configuration packages necessary to interface with UR robots in a ROS environment.

Overall, the Universal Robots ROS driver forms the lowest software layer of the control stack in this system. It abstracts the hardware interface into ROS compliant communication primitives, allowing the higher level application logic to remain independent from low level robot control details while still achieving reliable and repeatable execution.

3.2.4 Unity ROS Communication (ROS# / rosbridge)

Communication between the HoloLens 2 application and the ROS backend is implemented using ROS# (RosSharp) over WebSockets, with `rosbridge_websocket` acting as the bridge. This approach enables a Unity based application running on UWP to interact with a ROS system without requiring a native ROS runtime on the device, which is not supported on HoloLens.

ROS# provides a client side library in C# that allows Unity to publish and subscribe to ROS topics by serializing messages into JSON and sending them through a WebSocket connection. On the ROS side, `rosbridge_websocket` translates these JSON messages into standard ROS message types and vice versa. This effectively preserves the ROS publish subscribe paradigm across a platform boundary.

The communication model is designed to separate low frequency command messages from higher frequency telemetry streams. High level task requests (e.g., Point and Pick commands) are sent as discrete JSON messages, while pose updates and interaction signals are rate limited to avoid saturating the network or the HoloLens processing pipeline. This separation proved important for maintaining responsiveness during interaction intensive phases.

The main limitation of this approach is the overhead introduced by JSON serialization and WebSocket transport, which is less efficient than native ROS communication. However, within the scope of this thesis, the resulting latency remained acceptable for supervised manipulation tasks, where execution is explicitly gated by preview and confirmation rather than requiring hard real time response.

The repositories used for this integration are:

`https://github.com/siemens/ros-sharp`
`https://github.com/RobotWebTools/rosbridge_suite`

These provide the Unity client implementation (ROS#) and the ROS side bridge (`rosbridge_suite`) required to establish communication between the AR interface and the robot control backend.

Overall, ROS# with `rosbridge_websocket` offers a practical and reliable solution for integrating Unity based AR applications with ROS, trading some communication efficiency for portability and ease of integration across heterogeneous platforms.

3.2.5 Gripper Integration (OnRobot RG6 via Eyebox)

The OnRobot RG6 gripper is integrated into the system through an Eyebox interface rather than through direct network based control. In this configuration, gripper commands are issued from ROS as high level open and close signals, which are then relayed through the Eyebox hardware to the physical gripper mounted on the UR3e.

This approach was adopted due to the absence of direct IP level access to the RG6 in the available laboratory setup. While OnRobot grippers can in principle be controlled through dedicated drivers or URCap based interfaces, such integration was not accessible in this case. The Eyebox therefore acts as an intermediary layer that enables reliable actuation without requiring low level protocol implementation or proprietary driver configuration.

From a software perspective, the gripper is abstracted as a simple binary device. Commands are published on a ROS topic (e.g., `/gripper/cmd`) using a boolean interface, where `true` corresponds to closing and `false` to opening. This abstraction keeps the control logic lightweight and decoupled from the underlying hardware details. The rest of the system does not need to be aware of how the command is physically executed, only that the requested action is triggered.

One limitation of this setup is the lack of fine grained control over gripping parameters such as force, speed, or finger width, as well as limited state feedback from the gripper. However, for the structured pick and place tasks considered in this thesis, binary actuation proved sufficient to achieve consistent manipulation.

Overall, the Eyebox based integration represents a pragmatic engineering solution. It prioritizes reliability and ease of integration within the existing ROS pipeline, while maintaining the possibility of replacing the actuation layer with a more advanced driver in future iterations without requiring changes to the higher level application logic.

3.2.6 Application Level ROS Nodes

A set of custom Python ROS nodes implements the application logic between the AR interface and the robot driver. These nodes handle Point and Pick orchestration, point

based movement utilities, and preset management, translating high level user input into structured robot commands and execution sequences.

3.3 System Integration Workflow

Reproducibility is treated as a methodological requirement throughout this work. The integration procedure is therefore defined as a repeatable sequence rather than an ad hoc setup process.

3.3.1 Robot Calibration

Reliable motion control starts with extracting robot specific kinematic calibration parameters from the UR3e and storing them in a YAML configuration file. Without this, systematic deviations arise between commanded and executed poses, and these errors propagate into every manipulation task the system attempts.

3.3.2 Runtime Structure

During operation, the system runs as a set of concurrent ROS processes: the ROS core, robot driver, optional planning components, the WebSocket bridge, and the application level nodes. Keeping these components separate improves debugging, fault isolation, and runtime stability.

3.3.3 Connectivity Validation

Before task execution, communication between HoloLens 2 and the ROS backend is verified by confirming that the IP address in the Unity application matches the active network interface of the workstation. This step excludes basic communication errors before any investigation of higher level software behavior begins.

3.4 Interaction Model and Workspace Configuration

3.4.1 System Pipeline

The interaction pipeline has three main layers. The AR interface captures operator intent and presents visual feedback in the workspace. ROS manages communication and

coordination between the interface, the application logic, and the robot controller. The UR3e executes the corresponding motion while publishing telemetry and status back through the system.

Command messages are treated as low frequency, high priority events, while telemetry is published at higher frequency but with controlled bandwidth. This separation prevents routine status updates from interfering with execution critical commands.

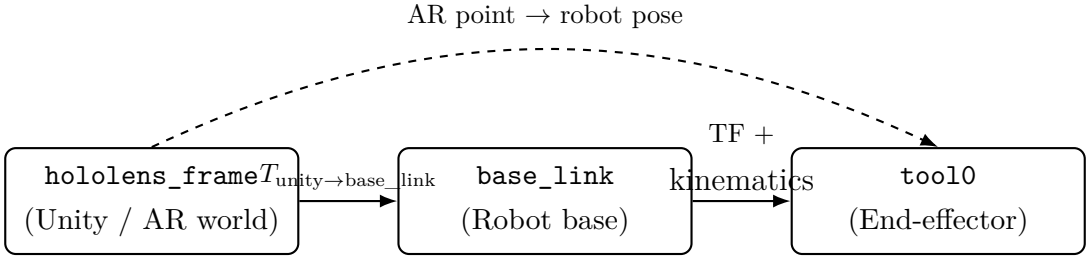

Figure 3.2: Coordinate transformation chain between AR and robot frames.

3.4.2 Preset Based Workspace Modeling

The workspace is modeled through predefined object and destination presets rather than fully dynamic perception. This reflects the structured nature of the intended application scenario, namely collaborative assembly support and workstation material handling, where relevant locations are known in advance and vary within limited bounds.

During interaction, user input is associated with the nearest valid preset within a tolerance threshold, improving repeatability and reducing sensitivity to small pointing inaccuracies. For long transfers, a predefined intermediate waypoint (the *middlepoint*) is inserted in the trajectory to improve motion predictability and safety.

3.4.3 Interaction Design

The interaction model is built around two principles: ease of use for the operator and robustness against unintended actions.

Gesture interaction is primarily used for spatial targeting and selection within the AR scene, while voice commands are used for mode control and explicit operational actions such as start, pause, resume, or cancel. Together, these modalities provide redundancy for critical functions and support a more structured command process than either channel could provide alone.

The Point and Pick workflow follows a command-driven sequence in which the operator designates a target through spatial pointing, the system maps that input to the nearest valid preset, and the executed command — for example, `obj1 to dest` — is logged to the

ROS workstation terminal. There is no separate AR overlay showing robot intent on the HoloLens display: transparency is achieved through the structured preset-based interaction model and the explicit voice command gate. Execution begins only after the operator issues a deliberate voice command, after which the robot performs the full pick-and-place sequence.

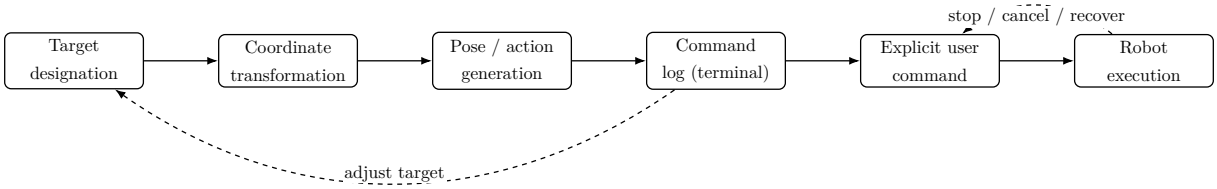

Figure 3.3: Command-driven interaction flow of the Point and Pick system. The operator designates a target through spatial pointing, the system maps the input to the nearest preset, and execution begins only after an explicit user command. Each executed command is logged to the ROS workstation terminal.

3.5 Exploratory Case Study Definition

To move beyond a purely descriptive system presentation, the methodology includes a small exploratory case study representative of a structured collaborative workstation scenario. The selected task is a pick and place operation: The user identified a target object through the AR interface, issued an explicit execution command, and supervised the robot as it transferred the object to a predefined destination.

This task was chosen because it captures a simplified but meaningful version of workstation level collaborative manipulation relevant to manufacturing support. It is intended as a simplified assembly support scenario rather than unconstrained free manipulation. It covers the essential interaction steps addressed in this thesis: spatial target designation, explicit command-based execution, and supervised manipulation.

The case study is executed in a controlled laboratory environment with predefined object and destination positions. The workspace is kept stable across repetitions so that the evaluation can focus on interaction quality, system behavior, and operational constraints rather than uncontrolled environmental variability. Figure 3.4 shows the experimental workstation used during validation.

3.6 Assessment Protocol

The assessment combines technical observation with small-scale user-centred feedback. Its purpose is not to produce statistically generalisable results, but to collect initial evidence

Figure 3.4: Experimental laboratory setup used for the case study based evaluation of the AR mediated control system, including the UR3e robot, the structured workstation, and the operator wearing HoloLens 2.

about usability, interpretability, and practical limitations of the proposed interaction approach.

3.6.1 Participants

A limited number of participants are involved to obtain initial feedback on system usability. The group represents potential end users or technically literate users capable of performing the assigned task after a short familiarization phase. Because the study is exploratory, the emphasis is on collecting indicative rather than large sample evidence.

3.6.2 Procedure

Each participant follows the same sequence. First, the purpose of the system and the main interaction modalities are explained. Second, the participant completes a short familiarization phase covering gesture and voice commands. Third, the participant performs the defined case study task using the Point and Pick interface. Fourth, structured feedback is collected immediately after task execution.

During the task, participants designate the target, trigger execution through an explicit voice command, and monitor the robot through completion of the manipulation sequence. Use of stop or cancel mechanisms is available throughout.

3.6.3 Evaluation Instruments

The evaluation relies on three categories of evidence:

- **System logs:** ROS and application logs are used to reconstruct execution order, communication behavior, and interaction events.
- **Direct observation:** task execution is observed to identify hesitation, interaction difficulties, failures, and recovery behaviors.
- **Structured feedback:** after completing the task, participants provide qualitative feedback and complete a usability oriented questionnaire.

The primary questionnaire is the System Usability Scale (SUS), complemented by task specific questions on clarity of robot intent, ease of learning, confidence during interaction, perceived practicality, and any discomfort experienced while using HoloLens 2. If cybersickness or visual discomfort is considered relevant in the final experimental design, a short supplementary discomfort instrument may also be included.

3.7 Data Collection and Analysis

Data collection spans three levels.

At the system level, ROS logs record robot state, command history, diagnostics, and execution flow throughout each session. The AR application logs interaction events, such as gesture recognitions, voice commands, execution triggers, and target selections, making it possible to reconstruct the full execution sequence and identify failures related to communication, interaction, or runtime behavior.

At the observational level, notes are taken during each participant session to capture visible hesitation, command repetition, requests for clarification, interruptions, and recovery actions. These observations provide context that software traces alone cannot supply.

At the user feedback level, questionnaire responses and qualitative comments are collected after task execution to assess whether the interaction model is understandable, whether robot intent is clear before execution, whether the system feels manageable in use, and what limitations participants identify.

Because the assessment is exploratory and based on a limited number of participants, analysis is primarily descriptive. Log data verify execution consistency and surface technical issues; observation notes are summarised to identify recurring usability patterns; questionnaire outcomes provide indicative evidence on perceived usability. The combined interpretation of these sources supports the discussion of strengths, limitations, and deployment implications in the chapters that follow.

4. System Implementation

4.1 Chapter Overview

This chapter details the software implementation of the proposed AR mediated Human Robot Collaboration (HRC) system. Whereas Chapter 3 presented the methodological framework and the rationale behind the main design choices, this chapter focuses on the concrete implementation: software architecture, data flow, communication mechanisms, and execution logic.

The system is implemented as a distributed architecture with three main components: a Unity based AR front end on HoloLens 2, a ROS Noetic middleware layer, and a control layer interfacing with the UR3e robot and the OnRobot RG6 gripper. Much of this chapter focuses on the *Point and Pick* implementation, where the most consequential technical decisions are concentrated, including coordinate transformations, message handling, configuration management, and the logic that makes task execution structured and reproducible.

4.2 Software Architecture and Module Decomposition

The system is built as a modular distributed application with clear separation of concerns. Each component owns a well defined piece of functionality, making the system easier to maintain, extend, and debug. Problems arising in one layer can generally be isolated and addressed within that layer without propagating unnecessary changes across the whole stack.

The architecture breaks down into four main layers:

- **AR Interface Layer (Unity/MRTK):** handles gesture recognition, voice input, spatial interaction, and in situ visualization on HoloLens 2.
- **Communication Layer (ROS# / rosbridge):** provides bidirectional communication between the Unity application and the ROS backend over WebSockets.
- **Application Layer (ROS Nodes):** implements task orchestration, preset management, and execution logic through custom ROS nodes.

- **Control Layer (UR Driver):** interfaces directly with robot hardware, sending motion commands and receiving state feedback.

Communication between layers follows a publish subscribe model built on ROS topics. This keeps components loosely coupled and allows asynchronous data exchange, so neither side needs to block on the other. The result is improved modularity, scalability, and fault tolerance.

The complete software pipeline is illustrated in Figure 4.1.

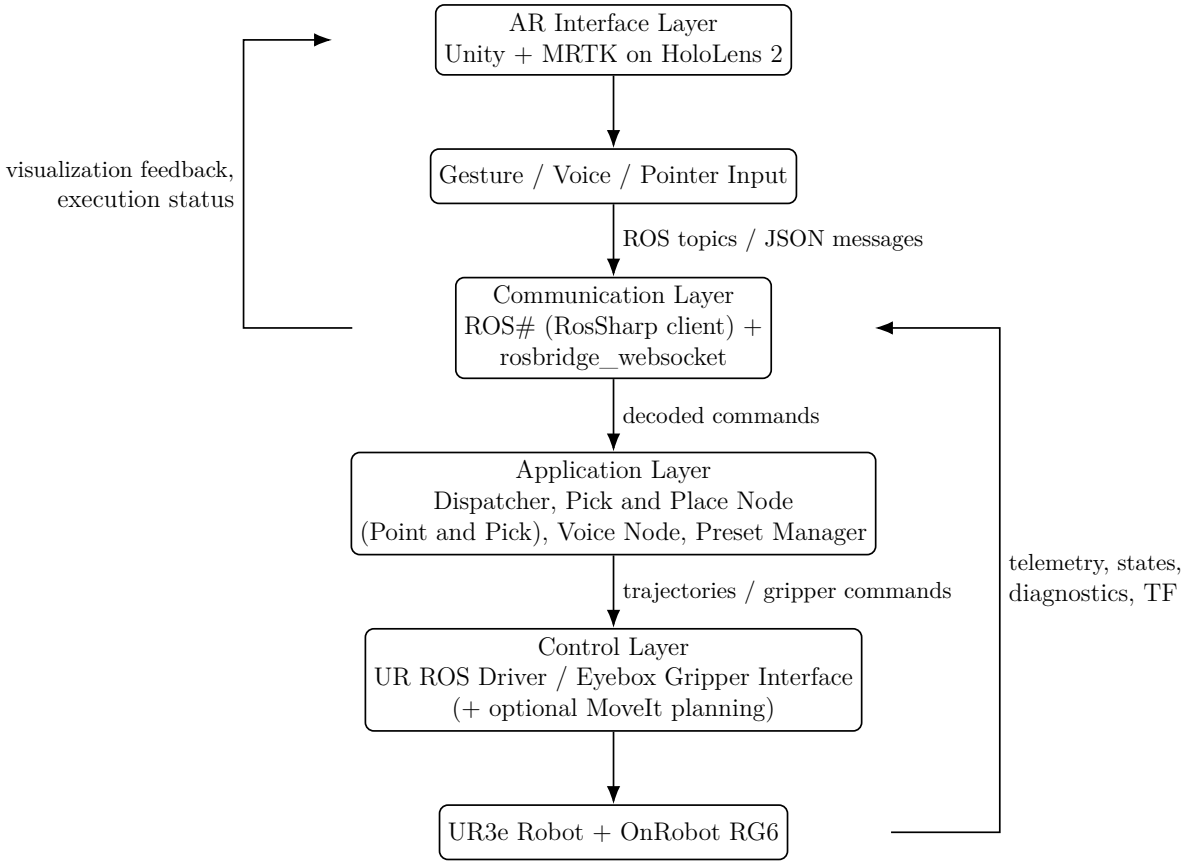

Figure 4.1: Software pipeline of the implemented AR mediated control system. Operator input is acquired through the HoloLens 2 interface (Unity/MRTK), transmitted via ROS# and `rosbridge_websocket`, interpreted by custom ROS nodes, and executed through the UR control stack. Bidirectional communication enables continuous feedback for visualization, supervision, and safety monitoring.

The diagram highlights two important characteristics. First, the system has a well defined top to bottom operational flow: input enters at the AR interface, is interpreted by the middleware and application logic, and is ultimately executed at the robot level. Second, the architecture is genuinely bidirectional: robot telemetry travels back through the communication layer into the AR interface, keeping the visualization synchronized with the actual runtime state of the robot.

4.2.1 Application Node Structure

The application layer is implemented as a set of Python ROS nodes, each responsible for a specific part of the Point and Pick workflow:

- `hololens_pick_place.py`: the main orchestration node, handling target interpretation, pose generation, and execution sequencing;
- `hololens_point_move.py`: a utility node supporting point based motion and spatial interaction functions derived from HoloLens input;
- `save_pose.py`: a configuration utility for acquiring and storing workspace presets, including object and destination poses.

Together, these nodes cover command interpretation, motion orchestration, and workspace configuration management.

4.2.2 Custom Interaction Nodes

Alongside the core application nodes, the system includes specialized nodes that translate raw operator input into ROS compatible commands. These sit between the AR interface and the robot control logic, acting as an interpretation layer that keeps interaction design decoupled from execution.

Gesture Processing Node

The gesture processing logic receives high level interaction events from the HoloLens interface, including gesture labels and spatial input, and maps them to structured robot related commands:

```
def gesture_callback(msg):
    gesture = msg.data

    if gesture == "pinch":
        publish_command("pick")
    elif gesture == "open_palm":
        stop_robot()
```

The callback is triggered for each incoming gesture message, extracts the semantic label, and routes it to the corresponding command. Keeping gesture interpretation separate

from downstream motion execution means that changes to the interaction design, such as adding a new gesture or remapping an existing one, do not require structural modifications to the rest of the control pipeline.

Every received gesture is also logged:

```
rospy.loginfo("Gesture received: %s", gesture)
```

During experiments, these logs were useful both for debugging and for reconstructing interaction sequences after the fact.

Voice Command Processing Node

A dedicated node handles voice input by subscribing to the `/hololens/voice_cmd` topic and translating recognized commands into control actions:

```
def voice_callback(msg):
    command = msg.data

    if command == "Execute":
        publish_command("execute")
    elif command == "Pause":
        pause_robot()
```

Voice input is treated as a high authority channel, primarily used for explicit operational actions such as execution, pausing, or cancelling a task. This is a deliberate design choice: commands that have direct consequences for robot motion are assigned to a modality that is less likely to be triggered accidentally than continuous hand input. It also means the system retains meaningful control even when hand tracking is degraded.

Voice commands are logged in the same way as gestures:

```
rospy.loginfo("Voice command: %s", command)
```

Together, the gesture and voice nodes form the multimodal interaction layer. They process input independently, but both feed into the same command structure for the execution pipeline.

4.3 Workspace Configuration and Preset Management

4.3.1 Manual Preset Acquisition and Workspace Initialization

Before the system can run, the workspace must be configured through a manual preset acquisition phase. This is performed once for each setup and produces a configuration file that the runtime uses without modification.

The process is straightforward: the operator jogs the robot end effector to each relevant location, including pick points and placement destinations, and saves each pose using a Python utility. Each preset captures:

- Cartesian pose (position and orientation);
- a logical identifier (e.g., `obj1`, `binA`);
- object dimensions, for pick targets.

The acquisition steps are:

1. manually jog the robot to the desired location;
2. verify the pose against the physical workspace;
3. run the preset saving script;
4. assign a name and optional object dimensions;
5. store the preset.

```
def save_preset(name, pose, size=None, preset_type="object"):
    preset = {
        "name": name,
        "type": preset_type,
        "position": {
            "x": pose.pose.position.x,
            "y": pose.pose.position.y,
            "z": pose.pose.position.z
        },
        "orientation": {
```

```
            "x": pose.pose.orientation.x,
            "y": pose.pose.orientation.y,
            "z": pose.pose.orientation.z,
            "w": pose.pose.orientation.w
        }
    }

    if size is not None:
        preset["size"] = {
            "x": size[0],
            "y": size[1],
            "z": size[2]
        }

    store_to_json(preset)
```

The function extracts pose data from the robot state and structures it into JSON. The optional `size` field associates geometric information with objects, which can later inform grasp planning or collision reasoning. Externalizing all of this into a configuration file means the system contains no hard coded pose values. Adapting it to a different workspace therefore requires rerunning the acquisition procedure rather than modifying source code.

Presets are stored persistently at:

```
config/presets.json
```

This separation between workspace configuration and runtime execution is one of the most important design decisions in the system. It makes reconfiguration faster, reduces operator dependent variability, and keeps the codebase stable across different physical setups.

4.4 HoloLens 2 Application Implementation

The HoloLens 2 application is the primary front end through which the operator interacts with the robotic system. Implemented in Unity with MRTK, it handles spatial interaction, visual feedback, gesture processing, and voice command integration. In practice, it allows the operator to designate targets directly in the workspace, and supervise task execution through the HoloLens spatial interface without relying on a detached control screen. Figure 4.2 shows the operator during spatial target designation.

Figure 4.2: Operator performing spatial target designation through the HoloLens 2 interface during use of the AR mediated control system.

4.4.1 Pointer and Gesture Processing

Spatial input is captured using MRTK and published to ROS at a controlled rate:

```
public class PointerPublisher : MonoBehaviour
{
    private float lastPublishTime = 0f;
    private float publishInterval = 0.05f;

    void Update()
    {
        if (Time.time - lastPublishTime < publishInterval)
            return;

        lastPublishTime = Time.time;

        Vector3 pos = GetPointerPosition();
        Quaternion rot = GetPointerRotation();

        PublishPose(pos, rot);
    }
}
```

The rate limiter enforces a minimum 50 ms interval between updates, capping publication at around 20 Hz. Without this, continuous hand movement would generate a flood of messages capable of saturating both the network and the HoloLens processing pipeline. At 20 Hz, pointer updates remain smooth enough for natural interaction, while robot telemetry, such as joint states and TF frames, can continue to run at higher frequencies set by the ROS driver configuration.

4.4.2 Event Throttling

High frequency input is filtered at the application level to prevent HoloLens saturation. Repeated events are coalesced, and visualization elements are time limited to maintain responsiveness throughout extended interaction sessions.

4.5 ROS Communication Layer

Internally, ROS uses typed message topics. Across the Unity ROS bridge, messages are serialized as JSON via `rosbridge_websocket`. This is a necessary trade off: the UWP environment on HoloLens cannot run native ROS, so JSON over WebSockets is the practical solution that preserves the publish subscribe architecture within platform constraints.

4.5.1 Message Encoding

Commands from the AR interface arrive as JSON:

```
{"command":"pick","target":"obj1","destination":"dest"}
```

This encodes high level task intent rather than low level motion parameters. The backend therefore interprets what the operator wants to accomplish, not how each robot joint should move. This abstraction is what allows structured manipulation sequences to be exposed to the user as simple interaction primitives.

4.5.2 Command Handling (Application Entry Point)

```
import rospy
from std_msgs.msg import String
import json

def callback(msg):
    data = json.loads(msg.data)
    if data["command"] == "pick":
        execute_hololens_pick_place(data)

rospy.init_node("hololens_command_handler")
rospy.Subscriber("/hololens/command", String, callback)
rospy.spin()
```

This node acts as the entry point of the execution pipeline. It subscribes to command messages, parses the JSON, and routes each request to the appropriate handler. Keeping this routing logic separate from both the communication layer and the downstream execution logic means that adding a new command type generally requires only a local extension of this command dispatcher.

4.6 Transformation and Frame Handling

```
point_base = transform_to_base(point_hololens)
```

This function applies the transformation between the HoloLens coordinate system and the robot base frame using ROS TF2, realizing the system wide transformation

$$T_{\text{unity}\rightarrow\text{base_link}}.$$

It queries the TF tree for the current transformation matrix and applies it to the incoming point. The result is that all spatial inputs, wherever they originate in the AR environment, are expressed in the same reference frame used by the robot controller. Without this alignment, a point that appears correct in HoloLens would correspond to a different physical location in the robot workspace.

4.7 Execution Logic

Task execution is governed by an explicit command driven control logic rather than by a rigid finite state machine with formal stages such as *Preview* or *Confirmed*. The operator first designates a target through the AR interface, the system computes the corresponding robot action, and the command — specifying the selected object and destination preset — is logged to the ROS terminal for traceability. Execution begins only when the operator issues a deliberate command, such as a voice based execution trigger.

This logic still enforces structure and safety, but it does so through validation and explicit command gating rather than through a strict user interface state sequence. Invalid targets, unreachable poses, or missing configuration data are rejected before motion is started, and interruption commands remain available during runtime. In this sense, predictability is achieved through controlled execution conditions rather than through a formal state machine exposed as part of the interaction model.

4.8 Point and Pick Execution Pipeline

4.8.1 Pose Generation

```
def generate_pose(point):
    pose = PoseStamped()
    pose.header.frame_id = "base_link"
```

```
    pose.pose.position.x = point.x
    pose.pose.position.y = point.y
    pose.pose.position.z = point.z + 0.02
    pose.pose.orientation = fixed_orientation()
    return pose
```

The function generates a valid robot pose from a selected spatial point. The vertical offset lifts the approach point above the target surface, providing safe pre grasp clearance. Using a fixed orientation is a deliberate simplification: it trades grasp flexibility for robustness, which is appropriate in structured environments where object orientations are predictable.

4.8.2 Preset Snapping

```
def snap(point, presets, threshold=0.05):
    for p in presets:
        if distance(point, p) < threshold:
            return p
    return point
```

The snapping function absorbs pointing noise by mapping user input to the nearest preset within a 5 cm threshold. Small pointing errors, which are inevitable with hand based interaction, therefore do not propagate directly into execution. If no preset is close enough, the raw point is retained, preserving flexibility for targets outside the preset grid.

4.8.3 Execution Pipeline

```
def execute_hololens_pick_place(data):
    target = load_preset(data["target"])
    destination = load_preset(data["destination"])

    pre_grasp_pose = compute_pre_grasp(target)
    grasp_pose = compute_grasp(target)
    place_pose = compute_place(destination)

    if not is_reachable(pre_grasp_pose):
        rospy.logwarn("Pre grasp not reachable")
        return
```

```
move_to(pre_grasp_pose)
move_to(grasp_pose)

close_gripper()

move_to(retreat_pose(grasp_pose))
move_to(get_middlepoint())

move_to(pre_place_pose(destination))
move_to(place_pose)

open_gripper()
```

This is the core execution sequence of the system. It retrieves preset data, computes the necessary poses, validates reachability before committing to motion, and then steps through a deterministic manipulation sequence. The structure is intentionally explicit: each step is visible, each failure point is logged, and the sequence does not silently skip ahead. This predictability makes the system both debuggable and trustworthy from the operator's perspective.

One acknowledged limitation of the current implementation is the absence of gripper state feedback between the close command and the subsequent retreat motion. Because the RG6 is actuated through the Eyebox interface rather than a direct driver, the system cannot confirm whether a grasp succeeded before committing to the retreat trajectory. In practice, binary actuation proved consistent for the objects used in the case study. In a production deployment, however, this step should be gated on verified gripper closure through a state topic from a richer driver or a wrist force threshold to prevent the robot retreating with an empty gripper and dropping the object mid transfer.

4.9 Robot Control Integration

4.9.1 Motion Execution

```
group.set_pose_target(target_pose)
plan = group.plan()
group.execute(plan, wait=True)
```

Motion planning and execution are handled through MoveIt. The planner computes a feasible trajectory under the active kinematic constraints and current environment

representation, and the robot controller then executes it. This abstraction keeps planning logic separate from the orchestration code and allows the system to leverage MoveIt's existing collision checking and constraint handling instead of reimplementing them.

4.9.2 Feasibility Validation

```
if not plan or len(plan.joint_trajectory.points) == 0:
    rospy.logwarn("Invalid plan")
    return
```

Before any trajectory is executed, the system verifies that the planner has produced a valid result. An empty or invalid trajectory is rejected immediately rather than passed downstream. This simple check eliminates an entire class of execution failures.

4.10 Gripper Control

```
gripper_pub = rospy.Publisher("/gripper/cmd", Bool, queue_size=1)

def close_gripper():
    gripper_pub.publish(True)

def open_gripper():
    gripper_pub.publish(False)
```

Gripper control is abstracted behind a binary interface: `True` to close and `False` to open. This reflects the hardware constraints of the current integration, in which the RG6 is controlled through the Eyebox rather than through direct IP communication, while keeping the software interface clean and replaceable if the hardware changes. The rest of the system therefore remains independent of low level gripper control details.

4.11 Error Handling and Safety

```
if rospy.Time.now() - last_msg > rospy.Duration(2.0):
    stop_robot()
```

Communication loss is handled by a timeout watchdog: if no message arrives within two seconds, the robot stops automatically. This simple mechanism covers a failure mode

that would otherwise allow the robot to continue executing without current operator oversight. Beyond the timeout, the system also rejects invalid poses before motion is commanded and requires an explicit execution command before robot movement begins.

4.12 Performance Optimization

The main optimizations used to keep the system stable under real time constraints are input throttling, separation of command and telemetry streams, and lightweight visualization. Throttling prevents message floods from saturating the network or the HoloLens. Stream separation ensures that high priority execution messages are not delayed by routine state updates. Lightweight visualization reduces rendering load on the headset during active operation.

4.13 Build and Deployment

The HoloLens application is compiled as a UWP project and deployed via Visual Studio 2019. The ROS backend requires coordinated startup of multiple nodes in the correct order; the full procedure is documented in Appendix E.

4.14 Chapter Summary

This chapter has presented the implementation of the AR mediated control system as a distributed software architecture. Modular ROS nodes, a structured execution pipeline, and configuration driven presets work together to produce reliable and predictable collaborative manipulation.

The integration of AR interaction with industrial robot control is supported by well defined interfaces, controlled communication, and explicit execution logic. These design choices are intended not only to make the system function correctly, but also to keep it understandable, maintainable, and extensible for future development.

5. Exploratory Assessment Results

5.1 Overview

This chapter presents the results from validating the proposed AR mediated Human Robot Collaboration system: a UR3e collaborative robot with an OnRobot RG6 gripper, controlled through a HoloLens 2 interface built in Unity/MRTK and connected to a ROS Noetic backend via ROS# WebSockets.

Consistent with the methodology in Chapter 3, the results are organized around two complementary dimensions. The first concerns technical and operational system behavior: task execution flow, communication stability, runtime constraints, and error recovery. The second concerns the user centered evaluation: perceived usability, clarity of robot intent, ease of interaction, and practical manageability during a representative collaborative task.

This chapter presents the results from the exploratory case study and the user-centred usability assessment described in Chapter 3. The findings are organised around two dimensions: technical and operational system behaviour, and user perceived usability and interpretability. The reported case study should therefore be read as a simplified collaborative assembly support scenario implemented through a structured Point and Pick task in a controlled laboratory environment.

5.2 Setup and Exploratory Case Study Context

Experiments were conducted in a controlled laboratory work cell consisting of a benchtop workstation, a fixed UR3e robot base, and predefined object and destination positions. The robot was connected to the ROS workstation over Ethernet, while HoloLens 2 communicated with the same workstation through the local network via `rosbridge_websocket`.

The runtime configuration included the AR application on HoloLens 2, a ROS Noetic backend on Ubuntu 20.04, the Universal Robots ROS Driver, optional MoveIt based validation components, and the custom ROS nodes implementing the Point and Pick workflow. Gripper actuation was routed through the Eyebox interface.

The exploratory case study consisted of a structured pick and place task representative of workstation level collaborative manipulation. The user identified a target object through the AR interface, observed the AR feedback associated with the intended robot

action, explicitly triggered execution, and supervised the robot as it transferred the object to a predefined destination. Although simplified, this scenario was chosen because it concentrates the core interaction sequence addressed in the thesis, namely target designation, explicit command-based execution, and supervised manipulation, within a controlled assembly support setting where predictability and operator oversight are central.

Before each session, calibration and spatial registration were verified. Where drift was detected, re anchoring was performed before continuing any precision relevant task.

5.3 Evidence Sources and Assessment Criteria

The evaluation combined three sources of evidence: system logs, direct observation, and structured user feedback.

At the system level, ROS logs verified execution sequences, robot state changes, and command handling. Interface side logs on HoloLens 2 recorded gesture recognition events, voice commands, target selections, AR visualization events, and execution triggers, making it possible to reconstruct the complete interaction and execution flow for each session.

At the observational level, task sessions were monitored for hesitation, repeated commands, interruptions, visible confusion, and recovery behavior. These observations provided context that software traces alone cannot capture.

At the user feedback level, participants completed a post task usability assessment and provided short qualitative comments. The evaluation criteria covered:

- correctness and predictability of task execution;
- clarity of robot intent before execution;
- ease of learning and using the interaction model;
- perceived robustness of gesture and voice commands;
- perceived manageability of the system during the task.

5.4 Case Study Execution Observations

The exploratory case study focused on the structured Point and Pick task. During each trial, the user designated a target through the AR interface, explicitly triggered execution via voice command, and supervised the robot as it transferred the object to a predefined

Figure 5.1: Supervised execution during the case study task, showing the robot operating in the structured workstation environment.

destination. Figure 5.1 shows the system during supervised execution in the experimental workstation.

Although the task is structurally simple, it serves as a useful validation scenario precisely because it concentrates the core interaction elements of the proposed system into a single observable sequence. It therefore provides a meaningful basis for discussing the interface's role in simplified assembly support and workstation assistance contexts.

5.4.1 Task Flow and Execution Behavior

Execution followed the intended Point and Pick logic consistently across sessions. Users designated a target through spatial pointing, issued an explicit voice command, and the robot then performed the manipulation. This sequencing preserved a clear separation between target specification and robot action, which was one of the central design intentions of the system.

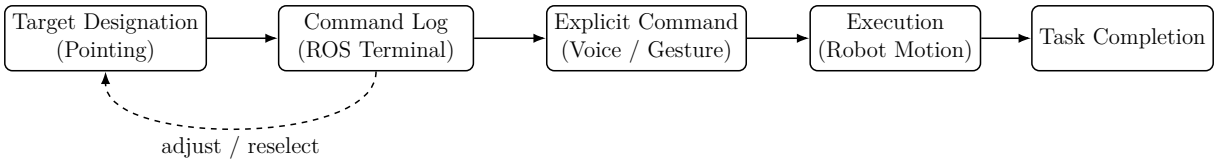

Figure 5.2: Observed interaction and execution flow of the Point and Pick workflow. The operator designates a target through spatial interaction and explicitly triggers execution before the robot performs the manipulation task. Each command is logged to the ROS terminal. A feedback loop allows target re-designation if the wrong preset was selected.

As Figure 5.2 illustrates, the command-gated execution model — in which the robot only moves after an explicit voice command — reduced the likelihood of unintended robot actions and supported a structured interaction flow.

Across evaluation sessions, the pick and place task was completed successfully in **4** out of **6** execution attempts (completion rate: **66%**). Incomplete executions were caused by high cpu load and blockage of Hololens, weak network connection and emergency stop of the robot according to its moving restrictions because the way its installed on the work branch. In successful runs, the robot followed the intended motion order, namely approach, grasp, retreat, transfer, place, and release. Transitions between phases were generally stable when registration remained accurate and input was correctly interpreted.

Observation also showed that the system functioned reliably as a supervision oriented interface. That said, practical usefulness remained partly dependent on the simplicity of the workstation layout and the responsiveness of the HoloLens during interaction intensive phases. In other words, the case study demonstrates the feasibility of the interaction logic, while also showing that operational value depends on the surrounding workstation

conditions.

5.4.2 Effect of Preset Based Execution

Preset based workspace modeling contributed substantially to execution consistency. Associating user input with predefined object and destination locations reduced sensitivity to small pointing inaccuracies and limited the need for repeated spatial adjustment. The predefined middlepoint improved motion predictability during longer transfers and reduced near surface movement.

These design choices were especially beneficial in a structured workstation context, where target positions are stable and the goal is reliable execution within a known task environment rather than unconstrained free space interaction. This matters for the interpretation of the results: the proposed approach is more credible in controlled manufacturing support scenarios than in highly variable manipulation settings.

5.4.3 Observed Limitations During Task Execution

Despite the generally stable execution flow, several limitations emerged. On reflective or geometrically ambiguous surfaces, inferred approach orientation was less reliable. In longer sessions, spatial drift occasionally accumulated to a point that required re anchoring before precision critical operations could continue. In a limited number of cases, gesture recognition degraded under partial hand occlusion or rapid movement.

These limitations did not prevent task completion, but they affected fluidity and, in some cases, required target reselection, re anchoring, or switching to voice based execution commands. The observed practical usefulness of the prototype was also partly constrained by the characteristics of the available workstation and by HoloLens device side performance during rapid interaction. The system operated successfully, but the combination of a relatively simple bench configuration and occasional headset performance degradation reduced the extent to which the AR interface could demonstrate a clear operational advantage beyond feasibility and intelligible supervision.

5.5 Basic Interaction and Control Results

Basic movement and control were evaluated through gesture and voice based commands. The command set proved understandable after a short familiarization phase, and users were generally able to trigger start, execute, pause, and stop operations without extended training.

Gesture interaction was most effective for spatial intent, particularly target designation. Voice commands proved more reliable for explicit control actions and safety relevant triggers. This produced a useful functional division: gestures for pointing and spatial selection, and voice for deliberate action commands.

When gesture tracking quality decreased, voice commands provided an effective fallback that allowed the task to continue without major interruption. This separation of roles across modalities appeared to reduce ambiguity and produce a more manageable interaction flow overall.

5.6 User Centred Usability Assessment

5.6.1 Participants

The user-centred usability assessment involved three participants. Each participant completed a short familiarisation phase before performing the case study task and then completed the post-task questionnaire. Because the assessment is exploratory and based on a limited sample, the results are interpreted as indicative evidence of usability and perceived practicality, not as statistically generalisable findings.

The participants came from different engineering backgrounds: Management Engineering, Embedded Systems, and Materials Engineering. All participants were female, with ages of 35, 26, and 30 years respectively.

The small sample size is consistent with the exploratory, formative purpose of this evaluation. Nielsen and Landauer's analysis of usability testing found that three to five participants are sufficient to identify the majority of usability problems in an interface [21], and subsequent work on early stage HRI evaluation confirms that small samples are appropriate when the goal is to surface interaction problems and generate hypotheses rather than produce statistically generalisable claims [15]. The findings are therefore interpreted as indicative evidence describing what these participants experienced, not as population level conclusions.

Participant	**Age**	**Gender**	**Field / profession**
P1	35	Female	Management Engineering
P2	26	Female	Embedded Systems
P3	30	Female	Materials Engineer

Table 5.1: Participant information for the exploratory user evaluation.

(a) Participant P1 (b) Participant P2 (c) Participant P3

Figure 5.3: User evaluation sessions showing the three participants wearing HoloLens 2 while performing the AR mediated Point and Pick interaction with the UR3e collaborative robot in the laboratory workstation.

5.6.2 Perceived Usability

The System Usability Scale (SUS) was used to obtain an initial indication of perceived usability. The three SUS scores were 87.5, 82.5, and 70.0 out of 100, producing a mean score of 80.0. The score range was therefore 70.0 to 87.5.

Given the exploratory nature of the study and the small number of participants, the SUS result should be interpreted cautiously. However, the mean score of 80.0 suggests a positive usability perception for the current prototype. In particular, the participants generally found the system understandable, manageable, and easy to learn after a short familiarization phase.

Participant	**Background**	**SUS score**
P1	Management Engineering	87.5
P2	Embedded Systems	82.5
P3	Materials Engineer	70.0
Mean	–	**80.0**

Table 5.2: System Usability Scale results from the exploratory user evaluation.

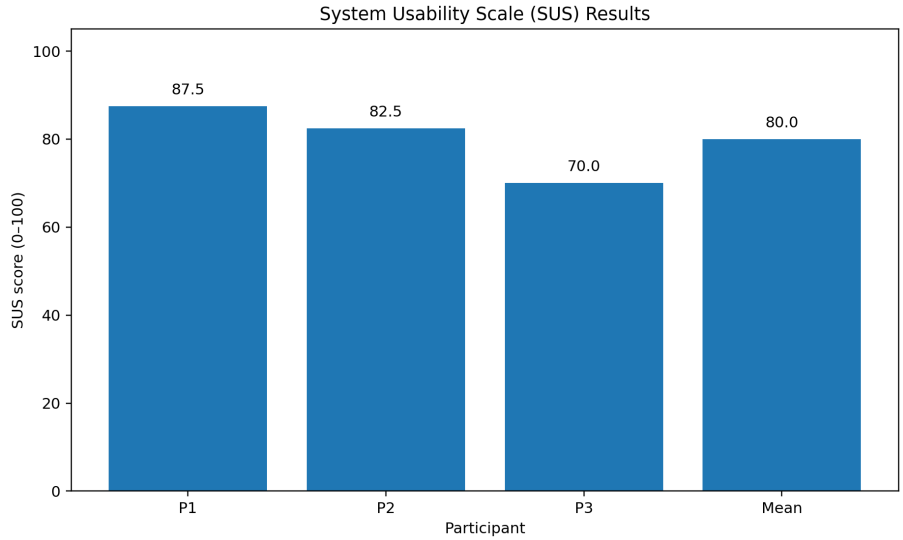

Figure 5.4: System Usability Scale scores obtained from the three participants, with a mean SUS score of 80.0.

The highest usability impressions were associated with the clarity of the interaction model, the ease of learning the pointing interaction, and the explicit voice command gate that prevented accidental execution. The lower usability impressions were associated mainly with limited movement flexibility, since the system accepted predefined target points rather than arbitrary positions, and with device side constraints such as processing delay when several commands were issued in sequence.

5.6.3 Clarity of Robot Intent and Interaction Flow

The qualitative feedback consistently indicated that participants could understand the robot's intended action before execution. All participants reported that the robot behavior was generally clear, especially because the actions were structured around predefined target points and predictable movements.

The structured preset-based interaction was identified as one of the most useful aspects of the system. Participants reported that the predictable robot behaviour — resulting from execution always following a fixed sequence toward a known preset location — made the robot action easier to anticipate, reduced uncertainty, and increased confidence during interaction. The command-gated execution model is important in this context because it ensures no motion begins without a deliberate operator decision, regardless of what spatial input was received.

The command driven interaction flow was also perceived positively. The distinction between target designation and explicit execution helped keep the interaction understandable. Users could first indicate the target and then explicitly command execution, retaining full control over when the robot moved. This supported a supervisory form of interaction rather than direct low level robot control.

5.6.4 Perceived Robustness of Multimodal Interaction

The multimodal design was generally perceived as useful. Participants described pointing to targets as intuitive and easy to learn. The system did not require much explanation for users to understand how spatial targeting worked. However, all participants also identified the same limitation: the robot could only operate on predefined target points and did not accept every possible movement or arbitrary workspace position.

This limitation was interpreted in two ways. On one hand, it reduced flexibility, especially for tasks requiring free movement or highly variable targets. On the other hand, participants recognized that predefined points improved accuracy, precision, reliability, and repeatability. For this reason, the limitation is not only a weakness; it also supports the intended use case of structured pick and place and simple assembly support, where repeatability is often more important than complete movement freedom.

Voice commands were also evaluated positively, especially for hands free control and safety related interaction. Participants valued the ability to control or stop the robot through voice commands, since this reduced the need for physical input during operation. At the same time, voice interaction showed practical limitations. One participant reported that the microphone performance was weak and that commands had to be spoken loudly.

Two participants also reported slower system response when many commands were issued in sequence. The system did not freeze, but the smart glasses required more time to process repeated commands.

Evaluation theme	**Observed result from participant feedback**
Clarity of robot intent	Participants reported that the robot's intended action was generally clear before execution, especially because the task used predefined targets and predictable movements.
Interaction clarity	The structured preset-based interaction and predictable robot behaviour reduced uncertainty and increased confidence during execution.
Pointing interaction	Pointing was considered intuitive and easy to learn, requiring little explanation from the operator side.
Predefined target points	Predefined points limited flexibility, but improved perceived accuracy, reliability, repeatability, and suitability for structured tasks.
Voice commands	Voice commands were considered useful for hands free control and safety, especially for explicit commands such as stopping the robot.
Technical limitations	Participants reported weak microphone performance, slower processing when many commands were issued, and limited device side responsiveness.
Suggested improvements	Improve microphone quality, reduce processing delay, make the headset lighter, and support more flexible target selection beyond predefined points.

Table 5.3: Summary of qualitative questionnaire findings from the exploratory user evaluation.

5.6.5 Comfort and Interaction Burden

The cyber sickness and discomfort responses were generally low. Across the three participants, the total raw symptom score was 19 out of a possible 144, corresponding to an average symptom intensity of 0.396 on a 0–3 scale. No severe symptoms were reported.

The most frequent symptoms were fatigue, eye strain, blurred vision, and fullness of head. One participant reported moderate eye strain and moderate fullness of head, while the remaining symptoms were mostly absent or slight. These results suggest that the system was acceptable for short evaluation sessions, but they also confirm that HoloLens based interaction introduces ergonomic and visual comfort constraints.

The open feedback confirmed these findings. Two participants mentioned the weight of the headset or possible fatigue during longer use. This suggests that the interface is manageable for short task sessions, but extended deployment would require attention to headset comfort, visual simplicity, and interaction pacing.

Participant	Raw symptom sum	Mean score	Main symptoms
P1	6/48	0.375	Slight fatigue, headache, eye strain, fullness of head, blurred vision, and dizziness with eyes closed.
P2	4/48	0.250	Slight fatigue, eye strain, fullness of head, and blurred vision.
P3	9/48	0.563	Slight fatigue, headache, sweating, blurred vision, and vertigo; moderate eye strain and fullness of head.

Table 5.4: Raw cyber sickness and discomfort results.

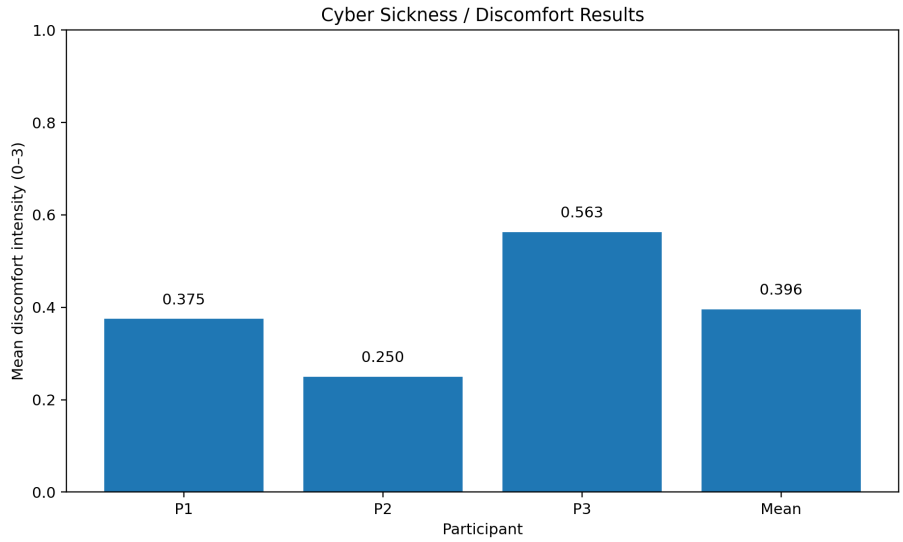

Figure 5.5: Mean discomfort intensity reported by participants on a 0–3 scale. The overall average was 0.396, indicating low discomfort during short evaluation sessions.

5.6.6 Interpretation of Usability Assessment Results

Taken together, the usability assessment results indicate that the proposed AR mediated interface was perceived as usable, understandable, and suitable for structured collaborative tasks. The mean SUS score of 80.0 provides positive initial evidence of usability, while the qualitative feedback explains why participants responded positively: the structured preset interaction made robot behaviour predictable, the predefined targets improved precision and repeatability, and voice commands supported hands-free and safety-oriented control.

At the same time, the results also identify clear limitations. The most important limitation is the restricted flexibility of predefined target points. This restriction supports repeatability in structured tasks, but it limits the system's applicability to more flexible or unstructured manipulation scenarios. Device side limitations also affected the user experience, especially processing delay under repeated commands, microphone weakness, and headset comfort.

Overall, the findings support the interpretation that the system is most appropriate for structured workstation tasks such as pick and place, simple assembly support, and repetitive material handling. In these contexts, predefined target points and preview based execution are not only constraints, but also mechanisms that improve predictability, accuracy, and operator confidence. For broader deployment, future improvements should focus on more flexible target selection, faster device side processing, improved voice recognition, and better headset comfort.

5.7 Runtime Stability and Technical Constraints

5.7.1 Communication Behavior

The ROS# WebSocket communication pipeline remained stable under normal operating conditions across all three evaluation sessions. No message loss was observed on high priority command topics (`/hololens/command`, `/hololens/voice_cmd`) during any session. Telemetry streams on `/joint_states` occasionally showed brief buffering when the HoloLens was processing rapid gesture input, but this did not interfere with command handling or execution triggering. Separating command and telemetry streams a deliberate architectural choice described in Section 4.5 was effective in practice: execution critical messages were not delayed by routine visualisation traffic even during interaction intensive phases.

5.7.2 Device Side Limitations

The most significant technical limitation appeared under high interaction load. Rapid gesture sequences and frequent pointer updates could saturate HoloLens side processing and produce temporary interface unresponsiveness. Event throttling, lightweight visualization, and limiting continuous updates reduced the frequency of these episodes, but did not eliminate the underlying constraint.

This result has an important implication: device side performance is not merely a usability concern, but directly affects the practicality and safety of AR mediated control. In the present setup, it also limited the extent to which the interface could be perceived as operationally advantageous rather than simply functional.

5.8 Error Modes and Recovery

The most common observed error sources were spatial drift requiring re anchoring, reduced gesture recognition reliability under adverse tracking conditions, feasibility rejection of unsafe or invalid target poses, and protective stops triggered by insufficient motion clearance.

In most cases, recovery was achievable through one of four actions: retargeting the object, switching input modality, re anchoring the AR frame, or adjusting workspace configuration parameters. What mattered as much as recoverability was that these failure modes were generally intelligible. Participants could usually infer why execution had not proceeded as expected, and in collaborative systems this intelligibility is essential for maintaining operator trust and allowing the task to continue effectively.

5.9 Summary of Findings

Five main findings emerge from the results.

First, the command-gated execution model improved clarity during task execution by ensuring robot motion only began after an explicit operator command, with predictable behaviour resulting from preset-based target association. Second, the Point and Pick interaction model enabled structured task specification without requiring continuous manual control. Third, preset based workspace modeling and safe middlepoint routing improved repeatability and predictability in the defined case study scenario. Fourth, multimodal interaction increased robustness by complementing gesture based spatial selection with voice based execution and control. Fifth, device side performance, spatial

registration, and tracking quality emerged as critical deployment constraints that must be explicitly managed.

At the same time, the results also indicate that the current prototype should be interpreted primarily as a feasible and intelligible supervision system rather than as a fully optimized productivity tool. The practical value of the interface was shaped not only by the interaction design itself, but also by the simplicity of the available workstation and by HoloLens performance limitations during interaction intensive use.

Taken together, these observations confirm that the proposed system is technically feasible and interactionally promising for structured collaborative workstation scenarios, while also indicating that practical value depends on careful management of drift, responsiveness, workstation suitability, and user workload.

6. Discussion

6.1 Overview

This chapter interprets the results from Chapter 5 and examines their meaning for the proposed AR mediated Human Robot Collaboration system. The system combines HoloLens 2, ROS# WebSockets, ROS Noetic, and a UR3e robot with an OnRobot RG6 gripper, with Point and Pick at the center of the interaction model.

The discussion develops along two complementary lines. The first concerns the technical and interaction related meaning of the results: how the system behaved, why the interaction model worked under certain conditions, and which limitations emerged during use. The second concerns the broader relevance of the system from a Management Engineering perspective, with attention to process stability, usability, standardization, safety governance, and practical value in manufacturing oriented collaborative workstation scenarios.

The discussion connects the findings from Chapter 5 to the three research questions stated in Section 1.3, and situates the results within the broader literature reviewed in Chapter 2.

6.2 Interpretation of the Main Findings

The results confirm that the system achieves its primary design objective: robot control and state awareness remain available at the point of work, while supervisory control stays explicit, predictable, and manageable.

The most significant outcome is a qualitative shift in how interaction is organized. The system stops resembling low level robot driving and starts resembling task specification followed by supervised execution. The user designates a target, issues an explicit execution command, and supervises the robot through completion rather than manually guiding every intermediate motion step. In the evaluated case study, this contributed to more predictable task flow and reduced the need for repeated manual adjustment, particularly when the workspace remained stable and preset based modeling could be applied effectively.

The user centered results support this interpretation. Participants associated usability with two specific factors: the visibility of robot intent before motion began and the presence of an explicit execution command before robot movement. What made the

system manageable was not the novelty of AR or the fact that the interaction was hands free, but the opportunity to interpret the intended action before it was carried out.

The results also reveal clear practical limits. Gesture interaction was intuitive for spatial selection but not uniformly robust across tracking conditions. Voice improved overall reliability for explicit execution and mode control, but depended on stable recognition conditions. Device side responsiveness emerged as a non trivial constraint. The prototype should therefore be understood primarily as a feasible and intelligible supervision system rather than as a fully optimized productivity solution. The interaction logic worked, but the extent to which its benefits became visible in practice was partly constrained by the workstation's simplicity and the headset's performance ceiling under rapid interaction.

Taken together, the results support the central argument of the thesis: AR improves transparency and reduces context switching in collaborative robot supervision, but its practical value depends on disciplined system design and on explicit management of drift, responsiveness, user workload, and workstation suitability.

6.3 Why Point and Pick Improves Task Execution

The effectiveness of Point and Pick can be explained by examining how it redistributes control effort between the operator and the system.

The first advantage is compact intent representation. A single act of spatial designation replaces a longer chain of corrective adjustments. Rather than nudging the robot through successive interface operations, the user communicates a goal at task level. This reduces interaction overhead and allows attention to remain on the workspace rather than on the mechanics of robot control.

The second advantage is structured mediation. The system does not map pointing directly to uncontrolled motion. Instead, it introduces intermediate processing steps such as coordinate transformation, feasibility checking, preset association, and terminal-level confirmation of the matched target. These mechanisms make execution safer and more interpretable. The system absorbs part of the control burden while keeping the user in a supervisory role, not because it automates the decision, but because it structures the path from intent to action.

The third advantage is operational consistency. Because execution is constrained through presets, middlepoint routing, and explicit execution commands, repeated tasks become more standardized across repetitions and operators. In manufacturing oriented scenarios, this matters because it reduces operator dependent variability, which is a process stability concern rather than merely a usability preference.

That said, the present results also show that these benefits are most visible in structured settings where the task and workspace are aligned with the method's assumptions. Point and Pick is best understood as a practical supervisory paradigm for constrained collaborative tasks, not as a universal manipulation solution.

6.4 Implications for Manufacturing Oriented Collaborative Workstations

A central question raised by this thesis is whether the proposed interaction model has relevance beyond the laboratory. The results suggest that it does, particularly for structured workstation scenarios in which the operator remains physically engaged in the process and the robot acts as an assistive agent rather than an autonomous production unit.

The most natural application contexts are collaborative assembly support, structured pick and place, kitting assistance, and workstation material transfer between known locations. In such tasks, the operator needs to supervise the robot closely, adjust actions at short notice, and remain focused on the physical workspace. An interaction model that embeds state awareness and spatial target designation directly into the operator's visual field has practical value in these conditions because it reduces interface detachment from the task itself.

The command driven interaction logic is especially well aligned with assembly related workstations. Assembly scenarios often involve small part handling, repeated transfers, operator supervision, and the need for recoverable execution when deviations arise. Decision support before motion is useful wherever part placement, task sequence, and operator confidence matter as much as raw execution speed.

At the same time, the results also require a careful reading of relevance. The case study is a simplified scenario, not a full collaborative assembly process. Its value lies in demonstrating the interaction logic under controlled conditions, not in proving production readiness. The results also show that practical usefulness depends strongly on workstation suitability. If the scenario is too simple or insufficiently demanding from a supervision perspective, the added value of AR remains less visible. AR mediated control becomes more meaningful when the workstation actually benefits from in situ supervision, frequent decision points, and transparent execution feedback.

The contribution of this work is therefore best understood as a focused interaction approach for structured collaborative workstation scenarios, not as a general solution for all robotic manipulation contexts.

6.5 Management Engineering Perspective: Operational Value and Work System Design

6.5.1 Productivity, Throughput, and Process Flow

In collaborative workstations, throughput depends not only on robot motion speed, but also on the coordination overhead between human and robot. Traditional interfaces often impose switching effort because the operator must interrupt the physical task to inspect or modify robot behavior through a detached control device. The AR mediated model reduces this overhead by embedding supervision and interaction into the workspace.

Point and Pick compresses control further into a short loop of target designation, AR based action interpretation, and explicit execution. In operational terms, this can reduce avoidable corrective actions and support smoother task flow. However, the results show that these gains are conditional. When the interface becomes unstable or overloaded, the productivity benefit is reduced. When the workstation does not strongly demand continuous in situ supervision, the advantage of AR remains limited. Throughput improvement therefore depends on maintaining stable runtime conditions and on matching the interface to a scenario that can genuinely benefit from it.

6.5.2 Quality and Error Prevention

The user evaluation results bear directly on quality assurance. All three participants reported that the predictable, preset-based execution helped them verify the correct object had been selected, and two specifically noted that knowing the robot would always move to a known location reduced their uncertainty before issuing the execution command (Section 5.5.3). Detection happens before motion commits rather than after an incorrect pick has already occurred the preventive logic that quality engineering frameworks associate with upstream error control.

Preset snapping contributed to this concretely. By mapping pointing input to the nearest predefined target within a 5 cm tolerance, small designation errors unavoidable in hand tracking based interaction were absorbed before reaching the execution layer. The limitation is the same as the strength: this mechanism restricts the system to predefined locations. As participants noted, this improves precision and repeatability for structured tasks but makes the system unsuitable for scenarios requiring free or variable placement.

6.5.3 Standard Work and Variability Reduction

The clearest Management Engineering implication of the system lies in its potential to support standard work. Several design choices already move in this direction: predefined targets and destinations, a consistent execution grammar, explicit command gating, and standardized safe transit through the middlepoint.

These mechanisms reduce operator dependent variability, support reproducibility across sessions, and simplify reconfiguration. Adapting the workstation to a different layout means updating configuration assets rather than rewriting source code or reprogramming trajectories from scratch. In this sense, the system supports not only task execution, but also structured operational control of collaborative work.

6.5.4 Training, Skill Requirements, and Workforce Flexibility

All three participants reached independent task completion after a short familiarisation phase covering gesture and voice commands. None had prior experience with HoloLens or robot control interfaces. This is consistent with the claim that the Point and Pick grammar is learnable within a single briefing session a prerequisite for any argument about reduced onboarding burden relative to conventional robot programming.

The interaction was not, however, self explanatory. One participant initially attempted to trigger execution through pointing alone, before understanding that a voice command was required. This illustrates that the distinction between spatial designation and explicit execution triggering cannot be assumed to be intuitive and must be addressed explicitly in operator training. The skill requirement shifts from trajectory authoring to supervised interaction literacy, but that literacy still needs to be taught deliberately, not discovered through trial and error.

6.5.5 Safety Governance and Operational Compliance

The results confirm that safety in AR mediated collaborative robotics must be understood as a layered property rather than as a feature of any single component. In this system, safety operates at three levels: explicit execution gating and intent feedback at the interaction level; watchdog and communication supervision at the middleware level; and protective stops and conservative motion limits at the robot control level.

This structure resembles a defense in depth strategy. No single barrier is assumed sufficient; safety emerges from their combination. The discussion of device side responsiveness belongs in this safety picture as well. Temporary interface unresponsiveness is a safety concern, not merely a comfort issue, because it directly affects the operator's ability to

remain in control. Responsiveness must therefore be treated as part of the safety model, not as a secondary performance metric.

6.6 User Evaluation and Usability Implications

A key addition to this work is the attempt to move beyond pure technical demonstration and include a small-scale user-centred usability assessment. Although exploratory in scope, this part of the study clarifies what users actually found useful or problematic in the proposed interaction model.

The most relevant implication is that users valued interpretability more than novelty. Positive responses were driven mainly by the predictability of the robot's behaviour — knowing it would always move to a known preset location — and by the presence of an explicit execution command that prevented accidental triggering. In collaborative workstation contexts, the usability advantage of AR may lie less in futuristic interaction and more in improved supervision quality.

The second implication concerns multimodal design. The combination of gesture and voice did not merely add interface richness; it provided functional redundancy. Gestures handled spatial specification, while voice handled explicit control commands when needed. This division improved resilience and supports the idea that multimodal interaction is valuable not because it offers more options, but because it assigns each channel a clear operational role.

The third implication concerns onboarding. The interaction became more understandable after a short familiarization phase, but not instantly. This is important for deployment planning. Even an intuitive AR interface requires explicit instruction, and the quality of that instruction will influence acceptance. The system appears promising as a user facing control layer, but only when accompanied by concise operating procedures and a stable, well documented interaction grammar.

A fourth implication concerns the distinction between usability and operational usefulness a distinction Chan et al. made explicit in their manufacturing study, where AR reduced objective task time but was rated less dependable than the joystick [13]. The present results show a milder version of the same pattern: participants rated the system as usable (mean SUS 80.0) and understood the interaction, but also noted that predefined target constraints and device processing delays reduced the sense of operational advantage. A positive SUS score is evidence that the system is manageable, not that it is operationally superior to a simpler alternative for this specific task. The more informative question whether AR mediated control reduces coordination overhead compared to a tablet or teach pendant in the same workstation remains open and represents a meaningful direction for

future comparative evaluation.

6.7 Latency, Responsiveness, and Device Constraints as Deployment Factors

Responsiveness directly affects trust, perceived safety, and operational fluency. The results show that the communication pipeline remained stable under normal conditions, particularly due to separation of command and telemetry channels. Device side compute saturation under interaction bursts, however, emerged as a significant deployment level concern.

Three deployment implications follow from these observations. First, the interface must be workload aware: the 20 Hz pointer rate limit and time limited AR markers implemented in Section 4.2 were necessary conditions for stable operation, not optional refinements removing either produced noticeable lag in development testing. Second, interaction pacing should be part of operator briefing: the familiarisation sessions included an explicit instruction not to issue repeated commands in rapid succession, which was sufficient for short evaluation sessions. In longer operational sessions, this guidance should be reinforced by a visual or audio indicator when the device is near its processing limit. Third, the latency warning and blocking thresholds defined in Appendix C 120 ms for a warning, 200 ms to block high risk actions implement the principle that responsiveness gates execution. These values were set conservatively based on the HRI literature [16] and should be validated empirically against the network and hardware conditions of each deployment site.

These implications show that deployment readiness depends not only on algorithmic correctness, but also on computational ergonomics, interface discipline, and operational rules for maintaining responsiveness over time. They also help explain why the current prototype should be interpreted with caution: even when the interaction logic is sound, device limitations can reduce the extent to which that logic translates into smooth and clearly advantageous practical use.

6.8 Limitations and Edge Cases

Several limitations define the current boundaries of the system.

Grasp generation relies on raycasting, heuristics, and preset assumptions rather than richer perception. Reflective, curved, or visually complex objects remain more difficult to handle reliably. Registration drift is manageable through checkpointing and re anchoring,

but remains a practical obstacle for precision critical interaction in longer sessions. Workstation suitability is itself a limitation: in relatively simple or highly constrained setups, the system may demonstrate technical feasibility and intelligibility without yet showing a strong operational advantage.

Ergonomic burden from prolonged HoloLens use is a real constraint that suggests the interface is most justified where in situ interaction clearly adds operational value. Device saturation under rapid gesture input makes throttling and deliberate pacing necessary design conditions rather than optional refinements.

Finally, the evaluation remains exploratory in scale. The case study and participant based assessment provide structured initial evidence, not statistically generalizable conclusions. The results support a reasoned argument for feasibility and usability potential, not definitive large scale industrial validation.

6.9 Threats to Validity

Several threats to validity deserve acknowledgment.

Internal validity is influenced by learning effects, since users improve with repeated exposure, and while standardized onboarding reduces this, it cannot eliminate it. Construct validity is limited by the evaluation instruments: the study captures perceived usability and qualitative observations, but does not fully measure constructs such as long term trust, sustained workload, or situation awareness through dedicated instruments. External validity is constrained by the laboratory setting, which does not reproduce the variability of real production environments. Instrumentation related bias from tracking quality, network behavior, and device side responsiveness may also influence observed performance, complicating interpretation even with careful logging.

6.10 Design Guidelines

Ten practical guidelines emerge from the findings for AR mediated collaborative robot control in structured workstation settings:

1. Execution of grasp and place operations must be gated by an explicit operator command; no robot motion should begin without a deliberate voice or gesture trigger.

2. Command and telemetry channels should remain separated; visualization updates should be throttled and time limited.

3. Responsiveness should be visible to the user and should gate high risk actions.
4. Anchor management should be standardized through checkpoint validation and explicit re anchoring procedures.
5. Multimodal redundancy should be preserved, with gesture allocated primarily to spatial intent and voice to execution, mode control, and safety.
6. Conservative defaults, including offsets, speed scaling, and safe middlepoint routing, should remain the baseline unless evidence justifies changing them.
7. Presets should be managed as versioned operational assets through configuration files, not embedded in source code.
8. User training should focus on interaction grammar, execution logic, and recovery procedures rather than low level robot programming.
9. Device compute limits should be part of the operational safety model, addressed through throttling, simplified overlays, and interaction pacing guidance.
10. AR mediated control should be prioritized for workstation scenarios where in situ supervision, frequent decision points, and transparent intent visualization offer a clear operational advantage.

6.11 Summary

The discussion confirms that AR mediated control using HoloLens 2 and ROS can support predictable and intelligible collaborative manipulation when robot intent is made visible and execution is gated through explicit user commands. Point and Pick reduces the burden of continuous manual correction by converting spatial designation into a structured loop of targeting, AR based feedback, explicit execution, and supervised manipulation.

From a Management Engineering standpoint, the most significant contribution is process stabilization. Presets, safe transit patterns, explicit command gating, and user centered supervision support standard work, reduce variability, and provide a foundation for training and safety governance in structured collaborative workstation scenarios.

The study also makes clear that the value of the approach is conditional. Calibration quality, drift control, network stability, interface responsiveness, user familiarization, and workstation suitability all influence whether the system remains effective in practice. The proposed system should be interpreted as a promising and operationally meaningful step toward AR mediated collaborative workstation control, not as a fully general solution for all industrial robotics contexts.

7. Conclusion and Future Work

7.1 Summary of the Thesis

This thesis developed and evaluated an AR mediated Human Robot Collaboration system for a UR3e collaborative robot, using HoloLens 2 as the interaction device and ROS Noetic as the robot side control stack. The system architecture, interaction model, and validation procedure are described in detail in Chapters 3–5. This chapter summarises the main contributions, discusses industrial applicability, and identifies directions for future work.

The core concept is the *Point and Pick* workflow. The operator designates a target in three dimensional space, and the system translates that designation into a structured reach grasp place sequence. To keep execution predictable and manageable, the system uses preset-based target association combined with explicit execution commands issued through voice or gesture, ensuring the robot only moves after a deliberate operator decision. In structured layouts, preset based target and destination definitions, together with safe transit through an intermediate waypoint, reduce sensitivity to pointing imprecision and improve execution consistency.

Beyond system implementation, the thesis included an exploratory usability assessment based on a representative case study and structured user feedback. The case study was a structured Point and Pick task intended as a simplified assembly support scenario in a controlled workstation setting. The results showed that the proposed system is technically feasible and interactionally promising in structured collaborative workstation scenarios. AR mediated control can improve clarity of robot intent, reduce reliance on continuous low level manual control, and support more understandable supervision of collaborative manipulation. The study also identified practical limitations that are highly relevant for real deployment, including spatial drift, gesture recognition sensitivity, workstation suitability, and device side responsiveness under bursts of rapid interaction.

7.2 Key Contributions

Five main contributions emerge from this work.

1. **Point and Pick as a goal-based interaction model for collaborative manipulation.** The workflow captures operator intent at task level, synthesizes a

feasible manipulation sequence, and gates execution through preset-based target confirmation and explicit user commands, supporting structured and intelligible robot supervision.

2. **An end to end AR ROS architecture compatible with industrial robot integration.** The system implements a practical pipeline based on HoloLens 2, Unity/MRTK, ROS# WebSockets, ROS Noetic, and the Universal Robots driver stack, preserving compatibility with ROS based industrial robotics workflows.

3. **A safety-centred interaction grammar.** Explicit execution command gating, multimodal command triggering for operationally relevant actions, always-available interruption mechanisms, and constrained execution logic are integrated into a coherent framework oriented toward predictability and operator trust.

4. **A structured approach to calibration, registration, and reproducibility.** Robot specific kinematic calibration, AR to robot frame registration, drift handling, and repeatable bring up procedures are formalized as part of the system methodology, supporting more stable and reproducible operation across sessions.

5. **An exploratory usability assessment.** Through a structured case study and user centred questionnaire, the thesis moves beyond pure implementation and provides initial indicative evidence that the proposed interaction model is understandable, manageable, and relevant to manufacturing-oriented collaborative workstation tasks.

7.3 Industrial Applicability

A key outcome of this thesis is that AR is treated as an operational interface for collaborative work rather than as a visualization demonstration. In industrial settings, the value of collaborative automation depends not only on whether the robot moves correctly, but on whether the overall system improves supervision quality, reduces interaction friction, supports predictable execution, and enables safe recovery when problems arise.

The results suggest that the system is particularly suitable for structured workstation scenarios in which object locations and destinations are repeatable, reconfiguration must remain practical, and the operator continues to play an active supervisory role. The most natural application contexts are collaborative assembly support, kitting assistance, workstation material transfer, and structured pick and place between known locations.

One of the most significant design choices is the transition from unrestricted pointing to *preset aware pointing*. In many manufacturing workstations, object and destination positions are already standardized or semi standardized. Treating these as configurable

operational assets and using pointing primarily as a selection mechanism reduces variability caused by interaction imprecision while preserving intuitive spatial control, which is an important balance when repeatability must hold across sessions, shifts, and different users.

The safe middlepoint transit strategy is directly relevant for similar reasons. In real work cells, unconstrained diagonal movements near fixtures and parts increase collision risk and make robot behavior harder to predict. Routing through a predefined waypoint supports safer and more understandable motion, simplifying workstation planning and increasing predictability from the operator's perspective.

From a workforce standpoint, the system shifts the competence requirement away from robot programming and toward supervised interaction literacy. Users learn a compact interaction loop involving target designation, execution triggering, and supervision rather than defining motion trajectories directly.. This does not eliminate training needs, but it may reduce onboarding complexity and improve workforce flexibility in environments where multiple operators rotate across shifts.

Industrial applicability remains conditional, however. The practical value of the interface depends not only on its design, but also on the suitability of the workstation scenario. In tasks that do not strongly benefit from in situ supervision or transparent action feedback, the added value of AR remains less evident. Effective deployment would require treating calibration data, preset configurations, safety parameters, throttling policies, and runtime procedures as version controlled operational assets. Registration drift, network stability, device side responsiveness, and workstation task fit would all need to be addressed through defined operational routines rather than through ad hoc correction.

7.4 Limitations

The present work has several limitations that define the current boundaries of its applicability.

Grasp generation relies on raycasting, heuristics, and preset assumptions rather than richer perception. Reflective, curved, or highly variable objects remain more difficult to handle reliably. Spatial registration drift is manageable through checkpointing and re anchoring, but it affects confidence in precision critical interaction during longer sessions and would require more systematic handling in production deployments.

Workstation suitability is itself a limitation. The case study demonstrates that the interaction logic is feasible and intelligible, but practical usefulness depends strongly on whether the workstation benefits from AR mediated supervision. In simple or highly constrained setups, the system may show credibility without yet producing a strong

operational advantage.

Prolonged HoloLens use introduces ergonomic burden and potential fatigue, suggesting that AR mediated interaction is most justified where in situ value clearly outweighs the cost of continuous device wear. Device saturation under rapid interaction bursts makes throttling and deliberate interaction pacing necessary conditions for stable operation.

The usability assessment remains exploratory in scope. The case study and participant-based observations provide initial indicative evidence, not statistically generalisable conclusions. The work supports a reasoned argument for feasibility and usability potential, not definitive large-scale industrial demonstration.

Finally, the validated stack is specific to the adopted hardware and software combination. Alternative robots, grippers, or XR devices would require separate validation.

7.5 Recommendations for Deployment

Eight recommendations can be formulated for practical deployment in structured collaborative workstation scenarios:

The design guidelines in Section 6.8 address operation and extension of the current system. The following recommendations are directed at practitioners considering deployment of a similar AR mediated control approach in a real manufacturing context.

1. **Validate registration at the start of every shift, not only at commissioning.** Spatial registration should be checked against a fixed reference point before any precision task begins, with a defined tolerance threshold that triggers re anchoring automatically. Treating re anchoring as a fallback rather than a routine step is a common cause of in session failure.

2. **Run a functional end to end check before handing the system to an operator.** One complete pick and place cycle takes under two minutes and eliminates the most common sources of session failure: stale calibration, changed network address, or updated preset configuration that has not been verified.

3. **Separate the commissioning role from the operating role.** Preset acquisition and AR registration require different tools and skills than day to day Point and Pick operation. In a shift based environment, the person who configures the workspace should not be assumed to be the same person who operates it. Training and access controls should reflect this separation.

4. **Budget for device lifecycle management.** HoloLens 2 units require operating system updates, application redeployment, and periodic spatial calibration. These

are non trivial maintenance tasks in a manufacturing environment with limited IT support. Deployment planning should include device management procedures alongside application development.

5. **Deploy AR mediated control only in workstations that genuinely require in situ supervision.** If an operator can manage the robot effectively from a panel PC or tablet, the ergonomic cost of HMD based interaction is not justified. AR adds unambiguous value when the operator needs information that is only meaningful in spatial context such as target selection among closely spaced objects or pre execution confirmation of grasp pose in a partially cluttered workspace.

7.6 Future Work

Several directions for future work emerge from this thesis.

The most immediate is perception assisted Point and Pick. Integrating depth sensing, object detection, or pose estimation could improve grasp generation for reflective or less structured objects while preserving the usability advantage of rapid spatial designation.

A second direction is more advanced planning and collision modeling. Industrial grade motion planning could extend the approach beyond fixed safe transit policies, with AR potentially presenting alternative trajectories for operator interpretation or selection.

A third direction is shared autonomy and exception handling. The robot could propose candidate grasps, destinations, or recovery actions that the user approves or refines through lightweight interaction. This would be particularly valuable in semi structured scenarios where full specification is impractical.

A fourth direction is long session robustness, including characterization of drift behavior over time, investigation of more stable anchor management strategies, and optimization of the rendering and interaction pipeline to reduce device saturation during extended use.

A fifth direction is broader usability evaluation. Larger participant studies, more rigorous instruments, and comparisons across user profiles would strengthen the evidence base regarding learnability, trust, workload, and long term acceptance.

A sixth direction is validation in production like environments, testing the system under changing lighting, clutter, multiple users, and less controlled network conditions to determine how well the approach generalizes beyond the laboratory.

A seventh direction is workstation task matching: investigating more explicitly which types of collaborative workstation benefit most from AR mediated supervision, and under what task characteristics the added value of the interface becomes operationally significant

rather than merely demonstrative.

Finally, economic and organizational evaluation represents an important next step for establishing industrial relevance. Assessing effects on onboarding effort, downtime, defect prevention, process stability, and supervision quality would move the contribution from demonstrated feasibility toward stronger evidence of industrial value.

7.7 Concluding Remarks

This thesis shows that AR mediated, multimodal robot control can make collaborative manipulation more intelligible, supervisable, and operationally meaningful when the interface makes robot intent visible, gates execution through explicit user commands, and connects through a robust ROS based control architecture. The Point and Pick workflow provides a practical bridge between human intent and robot execution by transforming spatial designation into a structured and verifiable reach grasp place sequence.

Beyond the technical implementation, the work contributes to a broader Management Engineering perspective on collaborative automation. Preset based execution, AR based intent feedback, constrained supervision, and repeatable runtime procedures together support process stabilization, standard work, and safer interaction in structured collaborative workstation scenarios.

The thesis also makes clear that the value of the approach is conditional rather than automatic. Calibration quality, registration stability, runtime responsiveness, device ergonomics, user familiarization, and workstation suitability all influence whether the system remains effective in practice. The results should therefore be read as strong evidence of feasibility and promising operational relevance, not as proof of universal deployability across all industrial contexts.

For these reasons, this work provides a concrete foundation for further development of AR mediated collaborative manipulation as a human centered capability for modern manufacturing workstations.

A. System Architectures

A.1 High Level System Architecture (HoloLens → ROS → UR3e)

The overall architecture follows a structured distributed pipeline: the HoloLens 2 interface communicates with a ROS based control backend, which in turn manages the UR3e robot and the OnRobot RG6 gripper. Each component has a clearly defined role, and the boundaries between them are maintained through explicit communication interfaces.

The structure is illustrated in Figure A.1.

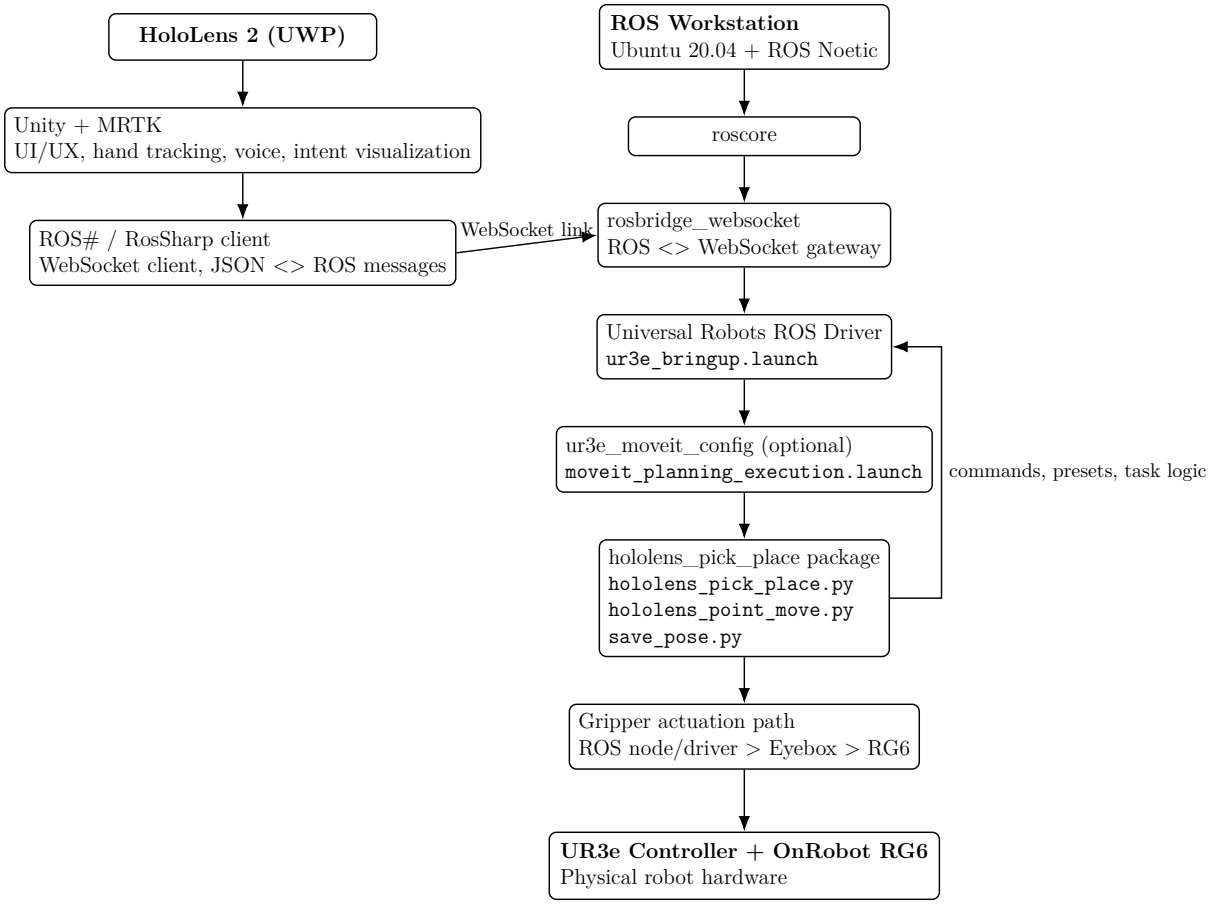

Figure A.1: Detailed runtime architecture of the proposed AR mediated control system. The HoloLens 2 application communicates through a ROS# / RosSharp WebSocket client with the ROS Noetic workstation. The workstation hosts the communication bridge, robot driver, optional planning components, application level Python nodes, and the gripper actuation path for the UR3e and RG6 hardware.

Within this architecture, HoloLens captures operator intent and renders in situ visualization, while the ROS workstation handles communication, orchestration, and all interaction with the robot controller.

One aspect worth noting is the gripper path. In this setup, the RG6 does not expose a direct IP interface, so gripper commands are routed through the Eyebox hardware interface, which acts as an intermediary between the ROS node and the physical gripper.

A.2 Dataflow for Point and Pick and Preset Based Execution

The complete execution flow for a Point and Pick operation in a structured preset based environment unfolds as follows:

1. **Target designation on HoloLens:** MRTK raycasting or hand pose estimation is used to identify the operator's intended target in three dimensional space.
2. **Coordinate transformation:** Unity world coordinates are mapped into the robot reference frame (`base_link`) through the registration transform

$$T_{\text{unity}\to\text{base_link}}.$$

3. **Preset association:** the selected point is matched to the nearest predefined object or destination pose within a specified tolerance, absorbing small pointing errors that would otherwise propagate into execution.
4. **Action generation and command logging:** based on the mapped target and the selected preset, the system computes the corresponding robot action and logs the command — for example, `obj1 to dest` — to the ROS workstation terminal for traceability.
5. **Explicit execution command:** once the operator has interpreted the displayed feedback, execution is triggered through a deliberate command, such as a voice or gesture based input.
6. **Safe transit routing:** for longer distance motions, the trajectory is routed through a predefined intermediate waypoint (the middlepoint), which serves as a safe apex above the workspace.

7. **Execution sequence:** the manipulation task then proceeds as a structured deterministic sequence:

$$\text{pre grasp} \rightarrow \text{grasp (close)} \rightarrow \text{retreat} \rightarrow \text{transit} \rightarrow \text{place} \rightarrow \text{open}.$$

8. **Runtime supervision:** watchdog mechanisms, protective stop handling, and operator triggered pause and stop commands remain active and accessible throughout the entire execution sequence.

B. Topics and Command Interfaces

B.1 Core Topics Used in the Project

Table B.1 lists the primary ROS topics exchanged during runtime. These channels carry the main communication flows between the HoloLens interface, the ROS backend, and the robot control stack.

Channel	**Type**	**Direction**	**Purpose**
`/hololens/finger_pose`	`geometry_msgs/PoseStamped`	HL → ROS	Continuous hand/finger pose stream for pointing, jogging utilities, and spatial context
`/hololens/gesture`	`std_msgs/String`	HL → ROS	Discrete gesture events (e.g., `pinch`, `open_palm`)
`/hololens/voice_cmd`	`std_msgs/String`	HL → ROS	Voice commands for mode control and explicit action triggers (e.g., `Pause`, `Execute`)
`/hololens/command`	`std_msgs/String`	HL → ROS	High level task commands encoded as JSON (e.g., pick and place requests)
`/joint_states`	`sensor_msgs/JointState`	UR → ROS	Real time robot joint telemetry
`/tf`	`tf2`	ROS	Frame transformations (`base_link`, `tool0`, AR anchors)
`/diagnostics`	`diagnostic_msgs/DiagnosticArray`	ROS → logs/UI	System health information, faults, and stop states
`/gripper/cmd`	`std_msgs/Bool` or driver msg	ROS → Eyebox	Gripper actuation command (open/close) routed through Eyebox
`/gripper/state`	driver/state msg	Eyebox → ROS	Gripper state feedback, when available

Table B.1: Core ROS topics used in the project.

Motion execution is handled through the controller interfaces exposed by the Universal Robots ROS driver. When MoveIt is enabled, motion plans are generated, validated, and sent to the configured controller pipeline before execution begins.

B.2 High Level JSON Command Schema (`/hololens/command`)

High level task requests travel as JSON encoded strings over the `/hololens/command` topic. A representative example is:

```
{"command":"pick","target":"obj1","destination":"dest"}
```

The three fields each carry a distinct piece of information:

- `command`: the task type (e.g., `"pick"`, `"place"`);
- `target`: the logical identifier of a predefined object pose (e.g., `obj1`, `obj2`);
- `destination`: the logical identifier of a predefined destination pose (e.g., `dest`, `binA`).

This schema allows the AR interface to trigger discrete, structured task execution rather than issuing a continuous stream of low level motion commands. The interaction therefore remains goal oriented, which is one of the defining characteristics of the Point and Pick paradigm.

B.3 Minimal Unity ROS# Payload Examples

The following examples show representative message payloads exchanged between the Unity ROS# client and the ROS backend.

PoseStamped example (`/hololens/finger_pose`)

```
{
  "header": {"frame_id": "base_link"},
  "pose": {
    "position": {"x":  0.40, "y": 0.31, "z": 0.38},
    "orientation": {"x": 0, "y": 0, "z": 0, "w": 1}
  }
}
```

This message carries a spatial sample of the operator's pointing direction or hand pose, already expressed in the robot reference frame.

Gesture example (`/hololens/gesture`)

```
{"data":"pinch"}
```

Gesture messages encode discrete interaction events. In the implemented system, they are primarily used for spatial interaction support, selection related input, or interruption related actions rather than as formal transitions in a rigid interface state machine.

Voice example (`/hololens/voice_cmd`)

```
{"data":"Execute"}
```

Voice commands provide the secondary interaction channel, primarily for mode switching, explicit execution triggers, and safety overrides. Their main practical value is as a robust fallback when gesture tracking is unreliable.

C. Parameters and Defaults

C.1 Manipulation and Motion Parameters

Table C.1 lists the primary manipulation and motion parameters used in the final system configuration. These values define the geometric clearances, safety margins, and feasibility constraints that govern robot motion throughout task execution.

Parameter	Default	Description
`approach_dist`	0.025 m	Offset used to define the pre grasp pose before contact
`retreat_dist`	0.040 m	Withdrawal distance applied after grasp completion
`use_middlepoint`	true	Enables routing through a safe intermediate waypoint for long distance motions
`middlepoint_pose`	cell defined	Preset pose defining the safe transit apex within the workspace
`keepout_margin`	0.010 m	Minimum clearance from obstacles and fixtures during motion
`ik_timeout`	0.25 s	Maximum time allocated for inverse kinematics feasibility evaluation

Table C.1: Manipulation and motion parameters.

C.2 UI and Interaction Policies

Table C.2 lists the main interface and interaction parameters. These parameters have direct practical importance, because poor choices at this level can make the interface feel sluggish, unstable, or unnecessarily demanding on HoloLens resources during extended use.

C.3 Gripper Parameters (RG6 via Eyebox)

Table C.3 summarizes gripper control parameters for the current hardware setup.

Parameter	Default	Description
`latency_warn_ms`	120	Threshold above which the operator is warned and only low risk actions are permitted
`latency_block_ms`	200	Threshold above which high risk actions (e.g., grasp/place) are blocked entirely
`reanchor_tol_m`	0.010	Spatial deviation threshold that triggers a re anchoring prompt
`feedback_ttl_s`	2.0	Lifetime of AR feedback markers, limiting visual clutter and rendering load
`event_throttle`	enabled	Rate limits input events to prevent device saturation
`max_publish_rate_hz`	bounded	Upper bound on message publication rate for pose and gesture streams

Table C.2: UI and interaction parameters.

Parameter	Default	Description
`actuation_path`	Eyebox	Gripper control is routed through the Eyebox interface due to the absence of direct IP access
`mode`	open/close	Binary actuation mode; extended presets depend on hardware and driver capabilities
`close_timeout_s`	1.0	Maximum allowed time for the closing action before a fault condition is triggered

Table C.3: Gripper parameters.

D. Calibration and Registration Procedures

D.1 UR e Series Kinematic Calibration (Mandatory)

Accurate motion execution with the UR3e depends on using the kinematic parameters specific to that individual robot unit rather than the generic model defaults. Calibration extraction is therefore a mandatory step before operating the system, not an optional optimization.

Calibration is obtained using the official Universal Robots procedure:

```
roslaunch ur_calibration calibration_correction.launch \
    robot_ip:=<robot_ip> \
    target_filename:="${HOME}/my_robot_calibration.yaml"
```

The resulting YAML file contains corrected kinematic parameters that compensate for the manufacturing tolerances of this specific robot unit. This file must be referenced at robot bring up so that the UR ROS driver operates with an accurate kinematic model.

Skipping this step has a predictable consequence: discrepancies accumulate between commanded and executed poses, and the effect is most visible in tasks requiring precise positioning, exactly the class of task for which this system is intended. The calibration file must be re extracted whenever the robot or controller is replaced, or if the file is lost or corrupted.

D.2 AR Robot Registration (HoloLens → `base_link`)

For spatial interaction in the Point and Pick workflow to be reliable, the Unity world coordinate system and the robot base frame (`base_link`) must be consistently aligned. This alignment is defined by the transformation:

$$T_{\text{unity}\rightarrow\text{base_link}}.$$

Registration follows a point based rigid alignment approach, in which the transformation is estimated from a set of corresponding spatial points collected in both frames. The rigid

transform takes the standard form:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

where $R \in SO(3)$ is the rotation component and $t \in \mathbb{R}^3$ is the translation. A point in the HoloLens frame is mapped to the robot base frame as:

$$p_{\text{base_link}} = T_{\text{hololens_frame} \to \text{base_link}} \cdot p_{\text{hololens_frame}}.$$

The registration procedure consists of the following steps:

1. Establish reference features in the workspace, such as a fiducial marker or a set of known anchor points.
2. Acquire corresponding point pairs: positions in the Unity coordinate frame (via HoloLens spatial mapping) and in the robot frame (`base_link`).
3. Compute $T_{\text{unity} \to \text{base_link}}$ using a least squares rigid alignment.
4. Validate the resulting transform at predefined workspace checkpoints.
5. Store the transform for runtime use and schedule periodic re validation during operation.

Registration is not treated as a one time setup step. Spatial drift is an inherent property of AR tracking systems, and alignment error can accumulate over time even when the initial registration is carefully performed. When deviation exceeds the configured tolerance threshold, re anchoring is performed before any precision critical operation proceeds. This keeps spatial targeting reliable and, equally importantly, preserves the operator's trust in what the AR overlay is actually showing.

E. Runtime Launch Procedure

E.1 Overview

This appendix documents the complete procedure for bringing the system online. The launch sequence mirrors the layered architecture of the system itself: ROS core services first, then robot drivers, then the communication bridge, and finally the application level nodes. Each layer depends on the one below it, so the order is not flexible.

Following this procedure helps ensure deterministic behavior, stable communication, and safe robot operation from the start of every session.

E.2 Core ROS and Robot Drivers

Terminal 1 ROS Core

```
roscore
```

Initializes the ROS master. Nothing else can start until this is running.

Terminal 2 UR3e Bring Up

```
roslaunch ur_robot_driver ur3e_bringup.launch robot_ip:=192.168.1.107
```

Starts the Universal Robots driver, loads controllers, establishes communication with the robot, and applies the kinematic calibration parameters extracted during setup.

Terminal 3 Motion Planning (Optional)

```
roslaunch ur3e_moveit_config moveit_planning_execution.launch \
    ur_type:=ur3e robot_ip:=192.168.1.107
```

MoveIt is not required for standard operation but is useful during testing and development for motion validation and debugging. It can be enabled when needed and omitted during production like sessions where the added overhead is not justified.

Terminal 4 Unity Communication Bridge

```
roslaunch rosbridge_server rosbridge_websocket.launch \
    unregister_timeout:=10 \
    websocket_ping_interval:=5 \
    websocket_ping_timeout:=10
```

The `rosbridge_websocket` server is the gateway between the Unity application on HoloLens and the ROS backend. The timeout and ping parameters are configured for active connection monitoring. If the link drops, the system detects it rather than waiting indefinitely for a response that will never arrive.

E.3 Application Layer (Unity Driven Orchestration)

Terminal 5 Pick and Place Orchestration

```
rosrun hololens_pick_place hololens_pick_place.py
```

This node receives high level commands from the AR interface and executes the structured pick and place sequences. It is the primary orchestration component of the application layer.

Terminal 6 Point Based Motion Utilities

```
rosrun hololens_pick_place hololens_point_move.py
```

Handles point based motion control and auxiliary targeting functions used during spatial interaction.

E.4 Preset Management Utility

Terminal 7 Pose Capture and Update (On Demand)

```
rosrun hololens_pick_place save_pose.py
```

Run this only when the workspace configuration changes, for example new object positions, updated destinations, or a layout reconfiguration. It is a setup and maintenance tool, not part of normal operation.

E.5 Diagnostic Test Publishers

The following commands allow direct testing of system components without requiring the AR interface. They are useful for verifying node behavior during development or when isolating a problem during troubleshooting.

Pick and Place Command Test

```
rostopic pub /hololens/command std_msgs/String \
"data: '{\"command\":\"pick\", \"target\":\"obj1\",
\"destination\":\"dest\"}'" --once
```

Simulates a high level task request exactly as if it had been sent from the HoloLens interface.

Finger Pose Test

```
rostopic pub /hololens/finger_pose geometry_msgs/PoseStamped \
"header: {frame_id: 'base_link'}
pose:
  position: {x: 0.40, y: 0.31, z: 0.38}
  orientation: {x: 0, y: 0, z: 0, w: 1}" -1
```

Simulates a continuous pose input stream for testing targeting and motion utility behavior.

Gesture Test

```
rostopic pub /hololens/gesture std_msgs/String "data: 'pinch'" -1
```

Simulates discrete gesture input events for testing gesture callback handling.

E.6 Connectivity Troubleshooting

If the Unity application on HoloLens cannot reach the ROS backend, the first thing to check is which network interface the workstation is actually advertising.

Linux (ROS Workstation)

```
ip a
```

Windows (Deployment Host)

```
ipconfig
```

The IP address configured in the Unity application must match the interface on which `rosbridge_websocket` is listening. In setups with multiple interfaces, such as Ethernet for the robot and Wi-Fi for HoloLens, selecting the wrong interface is by far the most common cause of connection failures.

F. Instrumentation and Logging

F.1 Overview

The system includes a structured logging strategy that spans both the ROS backend and the HoloLens Unity application. Capturing data at both levels makes it possible to correlate user input, system decisions, and robot execution, which is essential for understanding what actually happened during a session, particularly when something did not go as expected.

The collected data supports qualitative evaluation, consistency checks across trials, and identification of failure modes or responsiveness bottlenecks.

F.2 ROS Side Instrumentation

Robot side logging is handled through ROS bags, which record time synchronized message streams from key topics. The main recorded channels are:

- `/tf`: coordinate frame transformations, including `base_link`, `tool0`, and AR related frames;
- `/joint_states`: joint positions, velocities, and effort values;
- `/diagnostics`: system health, fault states, and protective stop indicators;
- `/hololens/*`: incoming commands, gesture events, and voice commands from the AR interface;
- `/gripper/*`: gripper command and state information, where available.

These logs allow the execution sequence to be verified after the fact (approach → grasp → retreat → place) and enable analysis of timing relationships between command issuance and robot response, which is useful both for debugging and for confirming that the system behaved as intended during recorded sessions.

F.3 Interface Side Instrumentation (HoloLens / Unity)

The Unity application logs interaction level events as they occur. These include:

- gesture detections (e.g., pinch, open palm) with timestamps;
- voice command recognitions (e.g., `execute`, `pause`, `cancel`);
- target designation events and raycast hit data;
- AR feedback generation and update events;
- execution trigger events.

These logs are particularly useful for identifying mismatches between operator intent and system interpretation, namely cases where the operator believed a specific target had been designated but the system registered something different. Without interface side logging, this kind of mismatch would be invisible in the ROS bags.

F.4 Performance and Responsiveness Indicators

Additional instrumentation captures device side behavior:

- event throttling activation flags, indicating when input streams are being rate limited;
- frame timing indicators, used to detect rendering slowdowns or temporary UI unresponsiveness;
- connection and latency indicators for the WebSocket communication layer.

These signals are essential for diagnosing performance degradation under rapid interaction bursts, namely the cases where the interface rather than the robot or the network becomes the limiting factor. They also provide the evidence needed to understand when and why the throttling mechanisms activate.

F.5 Data Usage and Limitations

The logs support qualitative analysis and consistency verification rather than precise numerical benchmarking. Timestamps allow approximate temporal alignment between interaction and execution, but measurement precision is bounded by device timing resolution and network variability.

Within those limitations, the combination of ROS side and UI side logging provides sufficient visibility to characterize system behavior, diagnose failures, and support reproducibility of experimental observations across sessions. For a system of this complexity, having both sides of the interaction recorded is more valuable than relying on either source alone.

G. Safety Policies and Fault Handling

G.1 Overview

Safety in this system is implemented as a layered mechanism spanning the user interface, the communication layer, and the robot control stack. The design objective goes beyond preventing unsafe execution. It is also to ensure that system behavior remains predictable, interruptible, and recoverable even when conditions degrade. A system that only prevents the worst case is not sufficient; one that remains understandable and controllable throughout degradation is the actual design goal.

G.2 Interaction Level Safety

Explicit execution gating. All grasp and place operations require a deliberate voice command before motion begins. The operator designates a target, the system maps it to the nearest preset, and no robot movement occurs until the execution command is received. Each command is logged to the terminal for traceability. This creates a conscious decision point before any potentially hazardous motion, and canceling at this stage has zero physical cost.

Explicit execution command. Executing a safety critical action requires a deliberate trigger via gesture or voice command. This is not simply a usability convention; it is a safeguard against ambiguous tracking signals or accidental input that might otherwise trigger unintended robot motion. The explicit command creates a conscious decision point before movement begins.

G.3 Communication and Runtime Safety

Watchdogs and heartbeat monitoring. The communication layer actively monitors the health of the connection between HoloLens and the ROS backend. If telemetry becomes stale or the link is interrupted, the system transitions to a safe halt condition rather than continuing to operate on outdated information.

Safe halt behavior. When link degradation or loss is detected, active motion stops and further execution is blocked. The system does not resume automatically. Stable

communication must be restored and the operator must explicitly authorize continuation. The underlying assumption is simple: an unexpected disconnection should always be investigated before proceeding, and the cost of a short pause is lower than the cost of uninstructed robot motion.

G.4 Robot Level Safety and Fault Handling

Protective stop integration. Robot level protective stops, triggered by collision detection or safety threshold violations, are propagated to the AR interface and surfaced clearly to the operator. The goal is that the operator should never have to wonder why the robot stopped. If the robot halts, the interface should indicate the reason.

Recovery procedure. Recovery from a fault follows a controlled sequence: acknowledge the fault, optionally reposition, and then explicitly authorize resumption. Automatic continuation is deliberately avoided. Operator awareness and control after a fault are treated as more important than minimizing downtime, which reflects the collaborative nature of the workstation.

G.5 Motion Safety Strategies

Safe middlepoint transit. For long distance movements, trajectories are routed through a predefined intermediate waypoint above the workspace. This reduces collision risk with the workspace, fixtures, and surrounding objects, and makes robot motion more predictable for anyone near the cell. A robot that moves through a known safe apex is easier to anticipate than one that takes the shortest possible path.

G.6 Interface Level Stability and Safety

Event throttling. High frequency gesture and pose updates are rate limited to prevent overloading the HoloLens. This matters for safety, not only for comfort: an unresponsive interface means the operator may be unable to issue stop commands when needed. Keeping the interface responsive is therefore a safety requirement.

Responsiveness as a safety factor. Interface responsiveness is treated as part of the safety envelope rather than as a secondary performance metric. If latency exceeds acceptable thresholds or the interface becomes unstable, execution is restricted until stable conditions are restored. The system does not allow risky actions to proceed through a degraded interface.

G.7 Summary

The safety architecture follows a defense in depth approach. Interaction level safeguards (AR feedback and explicit execution commands), communication level monitoring (watchdogs and heartbeats), robot level protections (protective stops), and interface level stability controls (event throttling) operate as independent but complementary mechanisms. No single one of them is sufficient on its own. Together, they help ensure that the system remains controllable, understandable, and recoverable under the operating conditions for which it was designed.

H. Build and Deployment Notes (HoloLens 2)

H.1 Overview

This appendix documents the build and deployment configuration used for the HoloLens 2 application. The goal is a repeatable and stable deployment pipeline, one that produces consistent behavior across development machines and avoids the SDK compatibility issues encountered during development.

H.2 Build Configuration

The Unity project is exported as a Universal Windows Platform (UWP) application and compiled using Visual Studio Community 2019. The key build settings are:

- target platform: UWP with ARM/ARM64 architecture;
- export from Unity to a Visual Studio solution before building;
- MRTK compatible build settings for spatial interaction and input handling.

H.3 Deployment Configuration

Deployment to HoloLens 2 is performed from Visual Studio with the following settings:

- build mode: *Release*; Debug builds introduce overhead that measurably affects runtime responsiveness and should not be used for evaluation sessions;
- target architecture: *ARM64*;
- deployment target: remote device via HoloLens IP address;
- authentication mode: Universal (the default for device pairing).

Release mode with ARM64 consistently produced the most stable runtime behavior. Debug builds remain useful during development, but they should not be treated as equivalent to Release builds for experimental sessions, because doing so can produce misleading performance observations.

H.4 SDK Compatibility and Stability Notes

A compatibility issue was encountered with Windows SDK version 10.0.20, which caused build and deployment failures that were not immediately easy to diagnose. The error messages were unhelpful, and identifying the root cause required additional troubleshooting.

Stable builds were consistently achieved by pinning the SDK to one of the following versions:

- 10.0.19
- 10.0.18

Anyone reproducing this system on a different development machine should verify the SDK version early in the setup process. It is easy to overlook and unnecessarily time consuming to debug after the rest of the setup is already in place.

H.5 Practical Deployment Considerations

For deployment to succeed, the following conditions must all be satisfied:

- the HoloLens device must be reachable on the same network as the development workstation;
- developer mode must be enabled on the device;
- the correct device IP address must be specified during remote deployment;
- firewall rules or network restrictions must not block communication with the device.

In practice, network reachability is the most common source of deployment failure. Either the device is on a different subnet or a firewall rule is silently blocking the connection. If deployment fails without a clear error, checking network configuration before anything else usually saves time.

I. Data and File Artifacts

I.1 Overview

The system generates three categories of artifacts during execution and experimentation: ROS bag files, interface logs, and configuration snapshots. Together, these provide a complete and correlated record of system behavior that supports validation, debugging, and reproducibility.

I.2 ROS Bag Files

ROS bags record time synchronized message streams from key topics. Files follow a session based naming convention:

`bags/session_YYYYMMDD_HHMM.bag`

The recorded topics include:

- `/tf`: coordinate frame transformations;
- `/joint_states`: robot joint telemetry;
- `/diagnostics`: system status and fault information;
- `/hololens/*`: commands, gestures, and voice inputs;
- `/gripper/*`: gripper commands and state feedback.

These logs make it possible to reconstruct full execution sequences after the fact and to analyze timing relationships between interaction events and robot responses. They are especially valuable when a session produces unexpected behavior, because the full synchronized record makes it possible to reconstruct the sequence of events without relying on memory.

I.3 Interface Log Files

The HoloLens application generates UI side logs in JSON Lines format:

`logs/ui_session_YYYYMMDD_HHMM.jsonl`

Each entry captures one of the following event types:

- gesture and voice interaction events;
- execution trigger events;
- target designation and AR feedback updates;
- latency and responsiveness indicators, where available.

These logs complement the ROS bags by providing the interaction side of the record, namely what the operator did, in what order, and how the interface interpreted it. The two sources together provide a complete picture; neither alone is sufficient.

I.4 Configuration Snapshots

Configuration artifacts are stored alongside runtime logs to ensure that experiments can be reproduced under consistent conditions:

- robot calibration file (YAML);
- preset definitions for objects and destinations (JSON);
- network configuration parameters (e.g., ROS bridge IP address).

These files represent the operational state of the system at the time of each session. Keeping them versioned and stored alongside the corresponding session logs is what makes it possible to reconstruct exactly which configuration was active during a given experiment. This is important both for debugging individual sessions and for ensuring that comparisons across sessions remain meaningful.

I.5 Summary

ROS bags, UI logs, and configuration snapshots together form a comprehensive and correlated record of system behavior. This combination supports not only debugging and validation, but also the kind of traceability that would be required in a production setting, where knowing exactly what happened and under exactly which configuration is a basic operational requirement.

Bibliography

[1] J. E. Colgate, M. A. Peshkin, and W. Wannasuphoprasit, "Cobots: Robots for collaboration with human operators," in *Proceedings of the ASME International Mechanical Engineering Congress and Exposition*, (Atlanta, GA), 1996.

[2] E. Matheson, R. Minto, E. G. G. Zampieri, M. Faccio, and G. Rosati, "Human–robot collaboration in manufacturing applications: A review," *Robotics*, vol. 8, no. 4, p. 100, 2019.

[3] A. Keshvarparast, D. Battini, O. Battaia, and A. Pirayesh, "Collaborative robots in manufacturing and assembly systems: literature review and future research agenda," *Journal of Intelligent Manufacturing*, vol. 35, pp. 2065–2118, 2024.

[4] M. Dhanda, B. A. Rogers, S. Hall, E. Dekoninck, and V. Dhokia, "Reviewing human-robot collaboration in manufacturing: Opportunities and challenges in the context of industry 5.0," *Robotics and Computer-Integrated Manufacturing*, vol. 93, p. 102937, 2025.

[5] Y. Cao, Q. Zhou, W. Yuan, Q. Ye, D. Popa, and Y. Zhang, "Human-robot collaborative assembly and welding: A review and analysis of the state of the art," *Journal of Manufacturing Processes*, vol. 131, pp. 1388–1403, 2024.

[6] M. Kheirabadi, S. Keivanpour, Y. Chinniah, and J.-M. Frayret, "A review on collaborative robot assembly line balancing problems," *IFAC-PapersOnLine*, vol. 55, no. 10, pp. 2779–2784, 2022.

[7] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Autonomous Robots*, vol. 42, no. 5, pp. 957–975, 2018.

[8] C. Petzoldt, M. Harms, and M. Freitag, "Review of task allocation for human-robot collaboration in assembly," *International Journal of Computer Integrated Manufacturing*, vol. 36, no. 11, pp. 1675–1715, 2023.

[9] 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.

[10] TODO: replace with the exact Rosen source you intended to cite.

[11] TODO: replace with the exact 2022 source you intended to cite for Fiorentino.

[12] M. Salami, P. Bilancia, M. Peruzzini, and M. Pellicciari, "A framework for integrated design of human–robot collaborative assembly workstations," *Robotics and Computer-Integrated Manufacturing*, vol. 97, p. 103108, 2026.

[13] W. P. Chan *et al.*, "Design and evaluation of an augmented reality head-mounted display interface for human-robot teams collaborating in physically shared manufacturing tasks," *ACM Transactions on Human-Robot Interaction*, 2022.

[14] TODO: replace with the exact Coronado 2023 source you intended to cite.

[15] L. A. Wulff, M. Brand, and T. Schüppstuhl, "Usability of augmented reality assisted commissioning of industrial robot programs," in *Annals of Scientific Society for Assembly, Handling and Industrial Robotics 2022*, pp. 91–102, Cham: Springer, 2023.

[16] TODO: replace with the exact Kocian 2023 source you intended to cite.

[17] F. Nenna, V. Orso, D. Zanardi, and L. Gamberini, "The virtualization of human–robot interactions: a user-centric workload assessment," *Virtual Reality*, vol. 27, pp. 553–571, 2023.

[18] Y. Zhang, B. Orthmann, S. Ji, M. Welle, J. van Haastregt, and D. Kragic, "Multimodal puppeteer: Exploring robot teleoperation via virtual counterpart with LLM-driven voice and gesture interaction in augmented reality," *arXiv preprint arXiv:2506.13189*, 2025.

[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, (Kobe, Japan), 2009.

[20] T. Puspanathan, "Systematic literature review on the acceptance of human-robot collaboration among assembly workers in manufacturing," *Procedia CIRP*, 2024.

[21] J. Nielsen, *Usability Engineering*. Boston, MA: Academic Press, 1993.