Karlsruhe Institute of Technology

Adapting the Human: Human-Robot Interaction Mediated by Wearable Technology and Augmented Reality

PhD Thesis
in Intelligent Industrial Robotics
of

David Puljiz

At the Department of Computer Science
Institute for Anthropomatics and Robotics (IAR) -
Intelligent Process Control and Robotics (IPR)

First reviewer:	Prof. Dr.-Ing. habil. Björn Hein
Second reviewer:	Prof. Dr.-Sc. Ivan Petrovic

12. January 2017 – 08. July 2024

Adapting the Human: Human-Robot Interaction Mediated by Wearable Technology and Augmented Reality

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik des
Karlsruher Instituts für Technologie (KIT)

genehmigte
Dissertation

von

David Puljiz

Tag der mündlichen Prüfung: 26. November 2024
1. Referent: Prof. Dr.-Ing. habil. Björn Hein
2. Referent: Prof. Dr.-Sc. Ivan Petrović

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, July 9, 2024

. . . . . .

(**David Puljiz**)

This thesis was very long in the making. Big part of the reason is the amazing research environment at the IPR, that makes one just want to try experimenting new things. I would like to thank my former colleagues at the IPR for providing a great environment for research as well as all their ideas and feedback. Also for listening about all the complaints when things did not work as intended. Special thanks to Vojta, Dave, Sergej, Michael, Yingbing and Patrick for their friendship and support. I would like to thank my Doktorvater, Prof. Björn Hein for all the interesting projects I had the opportunity to work on, as well as all his feedback and support for this thesis and my research work in general. I would also like to thank the many students that found my work interesting enough to participate in it. It was a privilege working with them and this work would not have been possible without their tests, inputs and ideas.

I would like to thank my lovely wife Tysh, who always believed in me and my work even when I did not. Who supported me no matter how tough it got and not only had the patience to listen to my constant banter about AR and Robotics, but also provided tough but fair reviews of everything from posters to this thesis right here. I could not have done it without her and these few sentences are by far not enough to express all the gratitude I feel. And finally, last but most importantly, my amazing daughter Teo, who, even though she is only 17 months old, finally got me to write this thesis and inspires me to be a better person every single day. Daddy loves you so much !

Abstract

Adapting the Human: Human-Robot Interaction Mediated by Wearable Technology and Augmented Reality

To increase the adoption of industrial robot arms beyond large, fixed production lines, fast and intuitive ways to program and interact with them is needed. Although related research has been ongoing for years, intuitive interaction and programming systems tend to be complex, expensive and inflexible. Especially when it comes to applications with increasing number of robots, such solutions become exceedingly expensive and unfeasible. This thesis proposes a paradigm shift in Human-Robot Interaction (HRI) systems, where the majority of the intuitive interaction system is worn by the human, centered around a Head Mounted Display (HMD). Modern HMDs provide a vast sensor suite capable of mapping the environment, localisation, hand tracking, voice recognition and most importantly display of Augmented Reality (AR) "holograms". The specific use case is the programming and interaction of industrial robotic arms, ranging from collaborative lightweight ones to large, heavy-duty arms. Such a worn system would reduce the amount of sensors and general hardware needed for HRI to zero. As all the sensors are worn by the human, interaction between a single human and multiple robots in a fenceless environment suddenly becomes cost-effective and feasible. This thesis will show the development of several components that such a wearable system requires as well as the final integrated system. The thesis will present state of the art developments in mapping, robot cell setup, intuitive programming, person tracking and HRI that went into the construction of the system. Although the system was implemented on the Microsoft HoloLoens HMD, effort has been put to make components flexible in regards to the HMD type and the sensors it carries as much as possible. This often included developing multiple versions of the same component working with different sensor or data inputs.

Keywords: *Human-Robot Interaction, Augmented Reality, Robot Programming*

Zusammenfassung

Um die Akzeptanz von Industrieroboterarmen über große, fest installierte Produktionslinien hinaus zu erhöhen, werden schnelle und intuitive Möglichkeiten zur Programmierung und Interaktion mit ihnen benötigt. Obwohl seit Jahren daran geforscht wird, sind intuitive Interaktions- und Programmiersysteme meist komplex, teuer und unflexibel. Vor allem bei Anwendungen mit einer wachsenden Anzahl von Robotern werden solche Lösungen äußerst teuer und nicht mehr praktikabel. Diese Arbeit schlägt einen Paradigmenwechsel in der Mensch-Roboter-Interaktion (MRI) vor, bei dem der Großteil des intuitiven Interaktionssystems vom Menschen getragen wird und sich auf ein Head Mounted Display (HMD) konzentriert. Moderne HMDs verfügen über eine umfangreiche Sensorik, die in der Lage ist, die Umgebung zu kartieren, zu lokalisieren, die Hand zu verfolgen, die Stimme zu erkennen und vor allem Erweiterte Realität (eng. Augmented Reality - AR) "Hologrammeänzuzeigen. Der konkrete Anwendungsfall ist die Programmierung und Interaktion von Industrieroboterarmen, die von leichten, kollaborativen bis hin zu großen, schweren Armen reichen. Ein solches getragenes System würde die Menge an Sensoren und allgemeiner Hardware, die für HRI benötigt wird, auf Null reduzieren. Da alle Sensoren vom Menschen getragen werden, wird die Interaktion zwischen einem einzelnen Menschen und mehreren Robotern in einer unzugänglichen Umgebung plötzlich kostengünstig und machbar. Diese Arbeit zeigt die Entwicklung verschiedener Komponenten, die ein solches tragbares System benötigt, sowie das endgültige integrierte System.

Stichwörter: *Mensch-Roboter-Interaktion, Erweiterte Realität, Roboterprogrammierung*

Contents

Abstract 7

Zusammenfassung 9

List of Figures 15

List of Tables 17

Acronyms 19

1 Introduction 3
- 1.1 Motivation . . . 3
- 1.2 Contributions . . . 5
- 1.3 Outline . . . 7

2 Basic Concepts and Tools 9
- 2.1 Augmented Reality - Concepts and Application to Human-Robot Interaction 10
- 2.2 Augmented Reality Hardware . . . 11
- 2.3 Microsoft HoloLens . . . 14
  - 2.3.1 Sensors and Data . . . 14
  - 2.3.2 Inputs . . . 16
  - 2.3.3 Data Display . . . 16
- 2.4 Wearable-based HRI . . . 17
- 2.5 Software Libraries and Packages . . . 17
- 2.6 Conclusion . . . 19

3 Mapping and Referencing 21
- 3.1 Mapping . . . 22
  - 3.1.1 HoloLens Mapping Capabilities . . . 22
  - 3.1.2 Spatial Mesh Sampling . . . 23
  - 3.1.3 Reconstruction from the Depth Stream . . . 24
  - 3.1.4 Reconstruction from Stereo Camera Pair . . . 24
  - 3.1.5 Mapping precision . . . 27
- 3.2 Referencing and Coordinate Systems . . . 30
  - 3.2.1 State of the Art . . . 31
  - 3.2.2 Proposed Referencing Methods . . . 32
    - 3.2.2.1 Referencing by Refining the User Input . . . 32
    - 3.2.2.2 Automatic Referencing . . . 33
    - 3.2.2.3 Continuous Referencing . . . 34
  - 3.2.3 Tests and Results . . . 36
    - 3.2.3.1 Refining user Input - Parameter Selection . . . 36
    - 3.2.3.2 Refining User Input - Influence of User Precision . . . 37

3.2.3.3 Refining User Input - Real World Error . . . 38
3.2.3.4 Automatic Referencing and comparison . . . 43
3.2.4 Discussion . . . 44
3.3 Conclusion . . . 44

4 Robot Workspace Setup and Programming 47
4.1 Robot Workspace setup . . . 47
4.1.1 State of the Art . . . 48
4.1.2 Proposed Method . . . 48
4.1.3 Tests and Results . . . 50
4.2 Robot Programming . . . 52
4.2.1 State of the Art . . . 53
4.2.1.1 Hand Guidance . . . 53
4.2.1.2 Graphical Programming . . . 54
4.2.1.3 Learning from Demonstrations . . . 54
4.2.1.4 Reinforcement Learning . . . 54
4.2.1.5 AR-based Approaches . . . 55
4.2.2 Proposed Methods . . . 56
4.2.2.1 Using Waypoints . . . 56
4.2.2.2 Hand Guidance . . . 57
4.2.2.3 Tracking Tools . . . 59
4.2.3 Tests and Results . . . 65
4.2.3.1 Using Waypoints . . . 65
4.2.3.2 Hand Guidance . . . 65
4.2.3.3 Tool Tracking . . . 66
4.3 Conclusion . . . 68

5 Tracking and Intention Recognition 73
5.1 Person Tracking . . . 73
5.1.1 State of the Art . . . 74
5.1.2 Proposed Approach . . . 74
5.1.3 Tests and Results . . . 78
5.2 Human Intention Recognition . . . 80
5.2.1 State of the Art . . . 81
5.2.2 Proposed Approach . . . 82
5.2.3 Tests and Results . . . 87
5.3 Conclusion . . . 93

6 System Overview 95
6.1 Desktop Computer . . . 95
6.2 HoloLens . . . 96
6.3 Pipeline . . . 96
6.4 Discussion and Future Work . . . 97
6.5 Conclusion . . . 100

7 Summary and Outlook 103
7.1 Summary . . . 103
7.2 Discussion . . . 106
7.3 Outlook . . . 108

List of Figures

1.1 The current paradigm in robotics suffers from several issues. Left - Robot programming is done mostly through classical programming in simulated environments. It is both unintuitive, as robots are spatial systems, and prone to issues due to environment modeling errors and the gap between simulation and reality. Right - Non-collaborative robot arms require heavy-duty fences to separate them from humans or a vast sensor suit in the robot cell to ensure the safety of the human coworker. 4

1.2 In the proposed system, the human would interact with the robot arm in a fenceless environment, with all the interaction and safety systems worn by the human. The programming would be graphical and spatial (in full 3D) making it more intuitive both for experts and lay users. 5

1.3 The building blocks of our system. Mapping is the first step. It generates a map that is then used to find the robot and calculate coordinate transforms in the referencing block. The coordinate transform and the map are then used to define the robot arm working environment in the setup step. In the programming step various methods may be used to define the desired trajectory of the robot arm. Finally in the interaction step, the human is tracked and their intention estimated for a safer, more intuitive collaboration with robot arms. 6

2.1 The Reality-Virtuality continuum as defined by Milgram in [1]. VR is positioned at the far right 10

2.2 HoloLens cameras, including the depth sensor. Sensors also include an Inertial Measurement Unit (IMU), not pictured here. Taken from the Microsoft research blog [2] 15

3.1 An example of the workspace of a robot. A similar setup with the robot, table and conveyor has been used in other tests throughout this work. . . . 23

3.3 Point cloud constructed from a disparity map (shown in orange) compared to a point cloud constructed from depth sensor data (shown in cyan). One can see large errors caused by improper synchronisation of the frames. The picture on the right shows a best case scenario and the potential of such a stereo approach 27

3.6 The main coordinate systems of the proposed system. One can see they are mainly split between the HoloLens dependent coordinate systems and the robot dependent ones. 30

3.7 The referenced robot as displayed to the user on the HoloLens alongside the seed hologram and the spatial mesh. 33

3.8 Automatic referencing process. a) The point cloud is divided into boxes, boxes with enough points are marked in yellow. b) Signature of Histograms of OrienTations (SHOT) feature correspondences between the model and the point cloud in the bounding box with the best match. c) The light blue robot is the model, dark blue is the final transformation using correspondence grouping of SHOT features, red is the final transform using the 4 step Iterative Closest Point (ICP) rotation method. 35
3.9 The influence of rotation on the registration algorithm. One can see that the algorithm is robust to rotational errors of the seed hologram, meaning that the user guess doesn't have to be overly precise. It is also evident that the error is not mirrored due to the robot's asymmetry. 39
3.10 Example result of experiment in ROS visualization (RViz). Although the user tried to position the cube to be perfectly aligned with the edges of the table - highlighted with red lines, one can see deviations -highlighted yellow areas. 39
3.11 The mean distance error and the standard error deviations of the four batches of tests 40
3.12 The user error test with the visualized HoloLens world coordinate system. 41
3.13 The mean distance error and the standard deviations of the four batches of tests 42
3.14 The mapping components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the Robot Operating System (ROS). 45
3.15 The mapping components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS. 46

4.5 First person view from the HoloLens. The waypoints, their position in the robot base coordinate system, and their projection on the table (which was part of the robot model in this case) can all be seen. A virtual robot is overlaid over the real one and used to test the validity of the planed trajectory. 56
4.6 First person view of the hand guidance modality from the HoloLens . . . 57
4.7 The convex mesh around the link defines the area in which hand movements are translated into robot movements. 58
4.8 Graphical representation of (4.1) - (4.6). The hand motion between the two consecutive frames has been vastly exaggerated for the sake of clarity 59
4.18 The setup components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS. 70
4.19 The mapping components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS. 71
4.20 The mapping components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS. 71
4.21 The mapping components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS. 72

5.7 HMM architecture for Human Intention Estimation (HIE). Worker's change of mind tendency is captured by the parameter α and the parameter couple β and γ set the threshold for estimating intention for each goal location. Increasing β leads to quicker inference of worker's intentions and increasing γ speeds up the decision making process. Parameter δ captures model's reluctance to return to estimating the other goal probabilities once it estimated that the worker is irrational. 85
5.8 First person view of the small test environment as viewed from the HoloLens (a) and as viewed from the player perspective in Virtual Reality (VR) (b). 87
5.9 Key moments of the real world experiment in the small-scale laboratory warehouse visualized in RViz. 88
5.10 Comparison of the HIE output between the real-world and the VR experiment. Intention estimations for the three goal locations are labeled with respect to their color in Fig. 5.9 (brown, cyan and purple), the unknown goal state is labeled black and the irrational worker state is labeled red. One can see that the results are similar for both setups. 89
5.11 Key moments of theVR scenario of a larger warehouse with 24 mobile robots, as visualized in RViz. Goal nodes are labeled with colored cubes. . 90
5.12 HIE output for the VR scenario with 24 mobile robots. Intention estimations for goal locations are labeled with respect to their color in Fig. 5.11, the unknown goal state is labeled black and the irrational worker state is labeled red. 91
5.13 The setup for the HIE test with an industrial robot. The goals are predefined as a green cylinder (not visible, left of the image), a red cube and a blue sphere. 91
5.15 The interaction and tracking components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS. 94

6.1 The proposed system workflow and communication 98
6.2 The User Interface (UI) menu to select the IP of the desired computer . . 99

List of Tables

2.1 Overview of different types of AR technology with availability, benefits and limitations. Note that this is not confined to robotics but general AR systems, just to display technological availability, benefits and challenges. 12
2.2 The sensors of the Microsoft HoloLens, their inbuilt functionality, the data output and the Frames per Second (FPS) of each data stream. 14

3.1 The observed averaged depth values for each repetition of the depth sensor accuracy experiment. Each value is the average of 5 pixels over 5 frames. 27
3.2 The average Euclidean distances and the Hausdorff distance between the laser scan, ground truth point cloud and the HoloLens' point cloud 28
3.3 The percentiles of the distances of each spatial map combination to the laser scan. 28
3.4 The steps of the proposed referencing algorithm, the algorithms used for each step and the parameters of each used algorithm. 37
3.5 Statistics of the conducted parameter tests of ICP, Super4PCS in combination with Moving Least Squares (MLS). The statistics are based on Root Mean Square (RMS) of the distances between the closes points in the matched robot and the original scene point cloud, in millimeters (mm). Execution time was not tested, however ICP performed noticeably faster each time. 37
3.6 Parameter test results of the ICP tests. All parameters listed as - did not have an effect on the result. 37
3.7 The mean, minimum, maximum and standard deviations in millimetres of the conducted tests. The first row represents the 12 original human guesses. The ICP-rotation and ICP-translation are the tests conducted by rotating and translating the original user guesses respectively 38
3.8 All the measurements of the positional deviation of the user placed cube between the HoloLens coordinate system and the robot coordinate system as visualized in RViz 40
3.9 The user hologram placment errors of the second batch of experiments. . 42
3.10 RMS error in millimeters of the two referencing methods of refining the user guess and the completely automatic one on the 11 scenes taken for the KR-5 robot. The column labeled "correct" is the result of visual inspection. 43
3.11 RMS error in millimeters of the two referencing methods of refining the user guess and the completely automatic one on the 11 scenes taken for the KR-16 robot. The column labeled "correct" is the result of visual inspection. 44

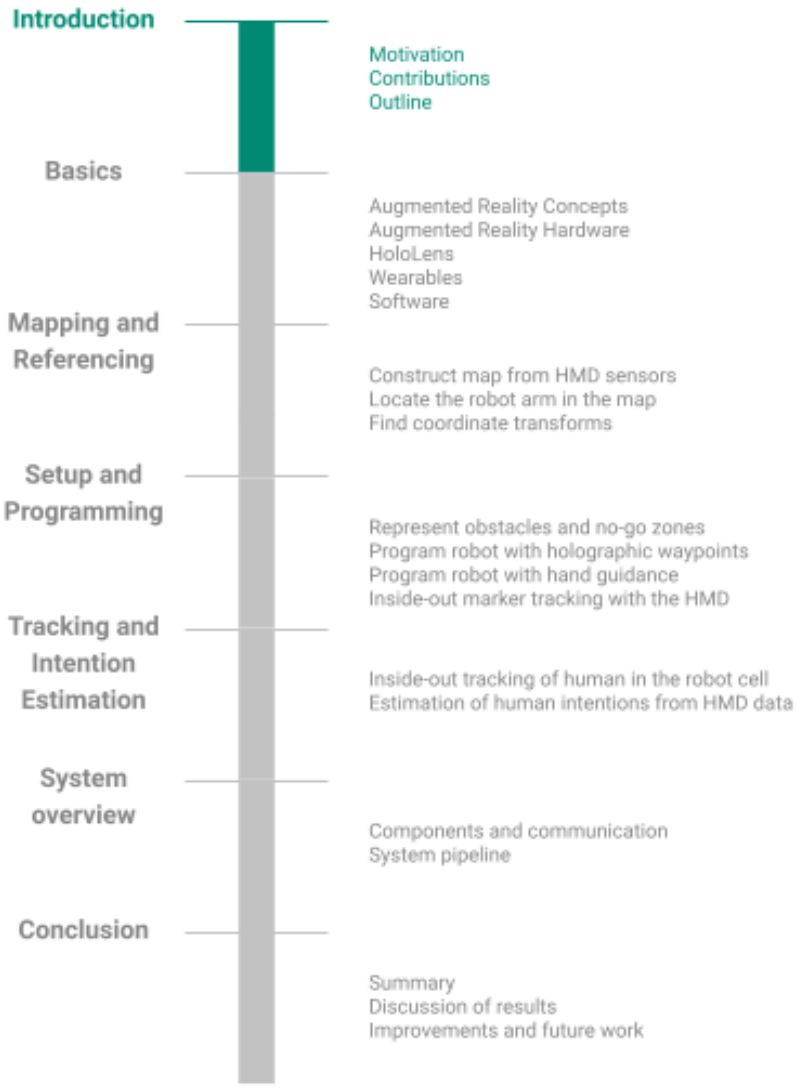
4.1 The number of false-positives and false-negatives in the 12 OctoMaps tested. The object contains between 96-160 voxels. 51

Acronyms

AGV	Automated Guided Vehicle
AR	Augmented Reality
AV	Augmented Virtuality
CNN	Convolutional Neural Network
CoBot	Collaborative Robot
DL	Deep Learning
DOF	Degrees of Freedom
FMS	Fleet Management System
FoV	Field of View
FPS	Frames per Second
GUI	Graphical User Interface
GVD	Generalized Voronoi Diagram
HIE	Human Intention Estimation
HMD	Head Mounted Display
HMI	Human-Machine Interface
HMM	Hidden Markov Model
HRC	Human-Robot Collaboration
HRI	Human-Robot Interaction
HUD	Heads-Up Display
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
IR	Infrared
JSON	JavaScript Object Notation
JTS	Joint-Torque Sensor
LfD	Learning from Demonstration
MDP	Markov Decision Proces
MLS	Moving Least Squares
MR	Mixed Reality
MRTK	Mixed Reality Toolkit

OUR-CVFH	Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram
PCL	Point Cloud Library
RANSAC	Random Sample Consensus
RGB	Red-Green-Blue i.e. Colour
RMS	Root Mean Square
ROS	Robot Operating System
RSI	Robot Sensor Interface
RViz	ROS visualization
SDK	Software Development Kit
SHOT	Signature of Histograms of OrienTations
SLAM	Simultaneous Localization And Mapping
SMEs	Small and Medium-sized Enterprises
SOM	Structure from Motion
SV	Safety Vest
SVD	Singular Value Decomposition
TCP	Tool-center point
ToF	Time of Flight
UI	User Interface
urdf	Universal Robot Description File
UWB	Ultra-wideband
VFH	Viewpoint Feature Histogram
VR	Virtual Reality
XR	eXtended Reality

1 Introduction



1.1 Motivation

The number of industrial robot arms continues to grow rapidly worldwide. Although the sales of Collaborative Robots (CoBots), such as the Kuka LWR [3], have seen a major increase, the vast majority of sold robots are still standard industrial manipulators. These robots are part of static production lines in large plants, and they are expensive and time-consuming to reprogram. Therefore they are unsuitable for flexible production paradigms

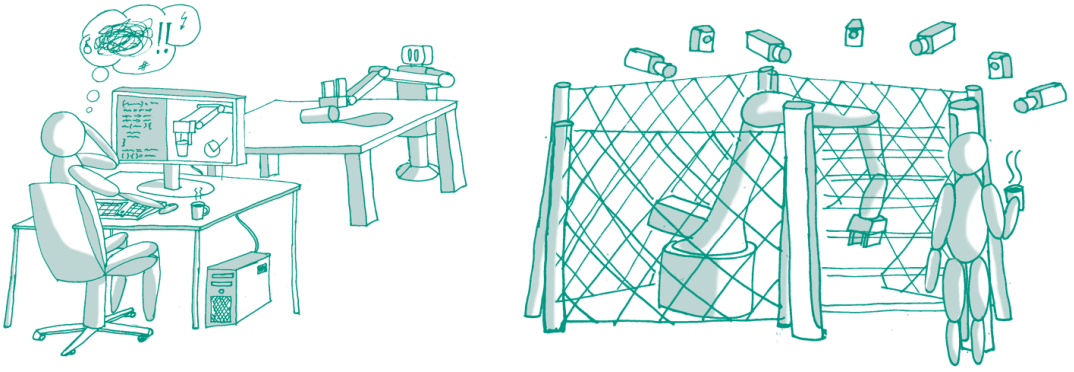


Figure 1.1: The current paradigm in robotics suffers from several issues. Left - Robot programming is done mostly through classical programming in simulated environments. It is both unintuitive, as robots are spatial systems, and prone to issues due to environment modeling errors and the gap between simulation and reality. Right - Non-collaborative robot arms require heavy-duty fences to separate them from humans or a vast sensor suit in the robot cell to ensure the safety of the human coworker.

which are required for Small and Medium-sized Enterprises (SMEs) due to small batch sizes [4], and to tackle increased product customization. Moreover, to achieve a truly flexible system, the robot needs to be able to work alongside humans in a way that is safe both for the human and the robot itself, without requiring heavy-duty security fences or extensive sensor suites, which is the case with non-collaborative arms.

Thus, one of the most pressing issues, according to the Industrial Robotics chapter of the Handbook of Robotics [5], is how to make robots easier to install and program, as well as to achieve close HRI in a fenceless environment both with lightweight CoBots as well as heavy industrial manipulators. Although there have been many attempts to solve these issues, no holistic system has yet been developed. Furthermore, the systems attempting to address the aforementioned issues require extensive fixed sensors and hardware, limiting the application to specialized and expensive robot cells. Such cells are unfeasible to be implemented in real-world applications. Firstly, they usually require extensive setup and calibration, meaning that they cannot deal with a changing cell setup and only increase the robot setup time instead of minimizing it. Secondly, due to the fixed nature of the system, every single robot cell would need to have its own hardware installed. With increasing number of robots, such an approach becomes prohibitively expensive.

This thesis proposes a paradigm shift, where devices for intuitive robot programming and HRI systems are worn by the human coworkers themselves. The worn system requires minimal calibration that can be done in a matter of minutes. It is also able to achieve everything that more complex systems can - cell setup, intuitive programming, person tracking and HRI. Finally such a system is completely portable. This means that a human wearing it can interact and program multiple industrial robots without the need for specialized hardware in every single cell.

The central component of the system is the Microsoft HoloLens HMD. This device features an extensive sensor suite, multiple interaction modalities, and the ability to fuse the real and virtual worlds through AR. It is already foreseen that AR will have a major economic impact on the industrial sector and wider, with an estimated market share of 4.7 billion dollars in the engineering sector alone (according to *Lumus*). The possibilities of

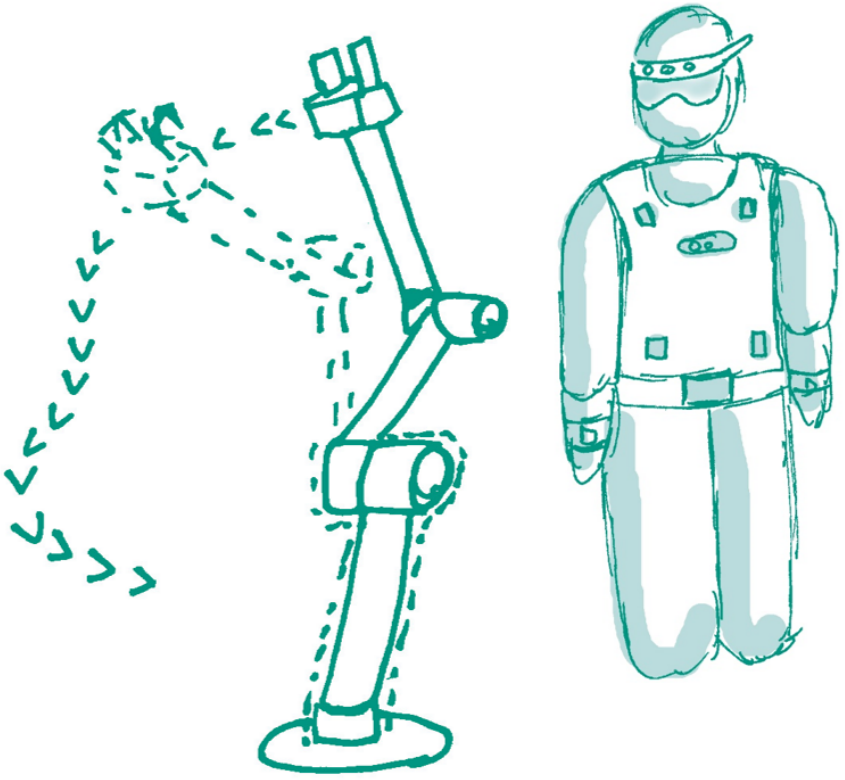


Figure 1.2: In the proposed system, the human would interact with the robot arm in a fenceless environment, with all the interaction and safety systems worn by the human. The programming would be graphical and spatial (in full 3D) making it more intuitive both for experts and lay users.

AR in improving interactions between robots and humans, particularly for robot programming, has already been well documented [6]. Additional wearables, such as IMUs are used to provide a level of safety and to extend the range of applications. Although this thesis does not develop certifiable safety hardware, a Safety Vest (SV) system such as the one developed by Končar for the SafeLog project (`https://www.koncar-institut.hr/en/content-center/projects/safelog/`) could easily be integrated. It uses Ultra-wideband (UWB) ranging hardware and safe, real-time computing to accurately measure distances between the vest and a robot and engages the emergency stop should the human get too close. This would require very minor modifications of the robot arm. This safety vest has already been declared safety-certifiable by TüV Nord.

Throughout this thesis the terms robot, robot arm, industrial robot or manipulator are used. In the context of this thesis they all refer to static articulated robots with six or seven rotational joints in a chain (one after another as opposed to parallel).

1.2 Contributions

The goal of this thesis is to develop an intuitive programming and interaction system for industrial robot arms based on the Microsoft HoloLens HMD. It should support the user in every step of using the robot - from setting up the robot cell, to programming the robot, to interacting with it in a safe and intuitive manner. Although previous research has gone

into specific areas of AR-based HRI, such as programming for example, to the best of the author's knowledge, no such holistic system has been proposed or developed.

Furthermore, the system should be fully portable and not require any fixed hardware beyond what the robot needs to perform its tasks. No extensive calibration will be required beyond the initial referencing between the robot and the HMD. Again, to the best of the author's knowledge, this has not been achieved before.

The developed system consists of three main hardware parts. The HoloLens HMD, a desktop computer, and an industrial robot arm. The HoloLens is connected to the computer via a wireless network. The HoloLens streams sensor data to the computer which runs most of the computation and returns to the HoloLens data to be visualized. Meanwhile the computer is also connected to the robot controller via an Ethernet cable and is responsible for sending control commands to the robot arm and receiving the status of the robot (the current joint rotations).

The system requires that these connections are setup and configured properly. It also requires a 3D model of the robot arm and a robot description file (specifically an Universal Robot Description File (urdf)). The path planner that is used to generate robot trajectories also requires a setup step. This is done through a Graphical User Interface (GUI) and is based on the previously mentioned robot model and description file.

The proposed system consists of several building blocks, most of them proposed and developed for the first time:

- Mapping of the robot arm's working environment via HMDs.
- Methods of referencing between a HMD and the robot arm.
- Setup of robot cells with HMDs - defining obstacles and safety zones.
- AR-based robot programming through waypoints, robot-agnostic hand guidance or inside-out tool tracking via Infrared (IR) markers.
- Inside-out tracking of humans inside robot working cells.
- HIE based on HMD during Human-Robot Collaboration (HRC).

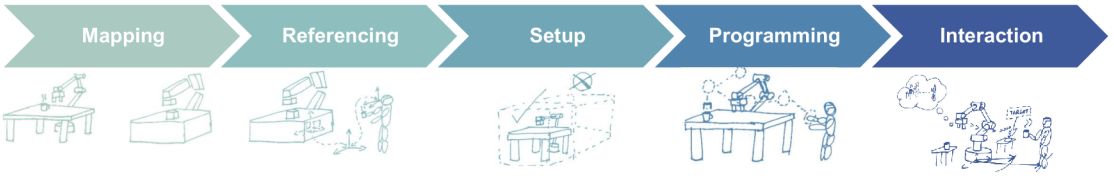


Figure 1.3: The building blocks of our system. Mapping is the first step. It generates a map that is then used to find the robot and calculate coordinate transforms in the referencing block. The coordinate transform and the map are then used to define the robot arm working environment in the setup step. In the programming step various methods may be used to define the desired trajectory of the robot arm. Finally in the interaction step, the human is tracked and their intention estimated for a safer, more intuitive collaboration with robot arms.

1.3 Outline

This thesis is split into chapters covering the specific system building blocks, with each chapter aiming to be as standalone as possible. **Chapters 2** and **3** present the basics for the rest of the chapters. Each other chapter has its own introduction, state of the art and experiments to provide the validity of the implemented paradigms.

Chapter 2 deals with the basics of AR and HMDs, as well as presenting the hardware and software packages used in this work for robot interfacing, path planning, point cloud processing etc.

Chapter 3 deals with methods to generate a map of the environment using the HMD's sensors and how to obtain a robust coordinate transformation between the HMD's and the robot's world coordinate systems. This is the basis for the other interaction paradigms presented in this work.

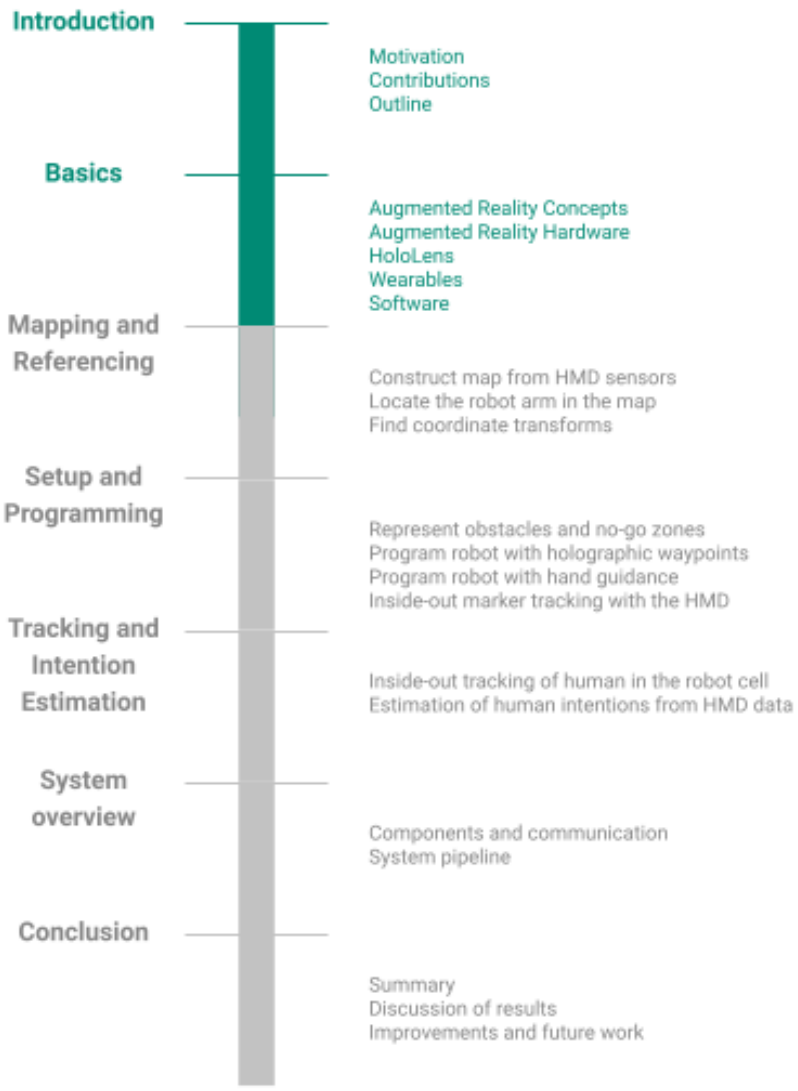
In **Chapter 4**, methods to quickly setup the robot working environment, meaning defining all static obstacles and safety zones using a HMD will be presented first. After the robot working environment is properly configured, different robot programming methodologies can be implemented. These include defining trajectories through holographic waypoints, by hand guidance or through various tools, tracked by the HMD itself.

Chapter 5 describes methods to track the human coworker inside the robot working environment and to estimate their intentions. Knowing the precise location of the human coworker is paramount for a safe HRI system, especially when dealing with non-compliant manipulators. Estimating the intention of the human allows the robot to predict the next actions of its human coworker, adapting its trajectory and workflow.

Chapter 6 presents the holistic system concept, the implementation of components, the interaction between the components and the workflow of the system.

Finally, **Chapter 7** presents the summary of this work, the overall evaluation, the issues and open points as well as the possible improvements. At the end an outlook with future research directions is presented.

2 Basic Concepts and Tools



This chapter will present the basic of AR, both theoretically and technologically, present the hardware of the HMD used in this thesis - the Microsoft HoloLens, define what are wearables and present the basic programming tools and libraries used to implement the system.

2.1 Augmented Reality - Concepts and Application to Human-Robot Interaction

AR itself is best explained using the Reality-Virtuality continuum, defined by Milgram [1] in 1994 and depicted in Fig. 2.1. AR starts by adding digital information into the real environment. Meanwhile adding information from the real world into a virtual scene is call Augmented Virtuality (AV). VR is a form of visualising a pure virtual environment, however modern VR hardware is often placed in the realm of AV as it usually some real world data as well (boundaries of the tracking area and hand or controller pose are the most common). Mixed Reality (MR) groups both AR, AV, while eXtended Reality (XR) is a catch-all term for AR, AV, and VR. In common use these definitions are often misused, with MR sometimes describing AR or XR, and glsAV commonly not being mentioned outside the research communities.

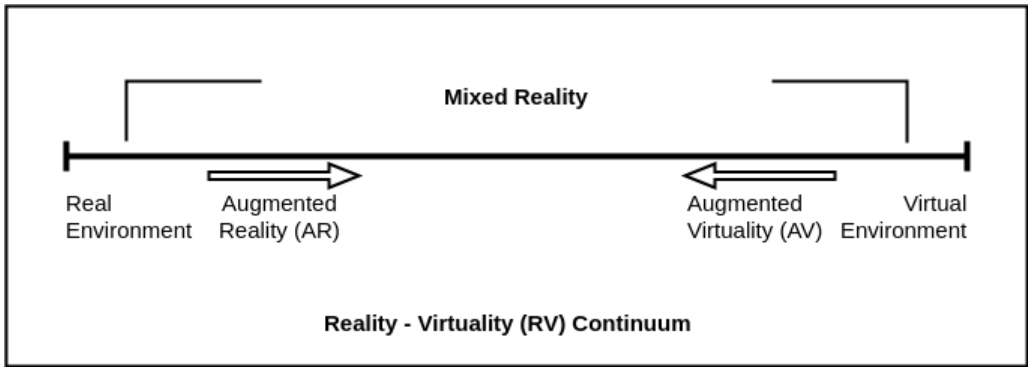


Figure 2.1: The Reality-Virtuality continuum as defined by Milgram in [1]. VR is positioned at the far right

Other definitions also exist. For example, Azuma [7] defines AR as a system that posses three distinct elements. Firstly, it combines real and virtual elements in a real environment, like the definition from Milgram. Additionally it runs interactively in real time and it registers real and virtual elements with each other. The third point requires interaction with the environment and therefore having mapping and localisation capabilities. By that definition, if lacking localisation capabilities, the AR device just becomes a Heads-Up Display (HUD).

AR has been used in robotics almost since its inception [8]. With the advent of feasible commercially available HMDs the field grew exponentially and it is now quite well established. Though relevant papers in the field will be presented in the appropriate subsection, it is opportune to direct the viewers to survey papers showing the whole breadth of research in the field.

The Reality-Virtuality Interaction Cube [9] was proposed as an extension of the Reality-Virtuality continuum for HRI. This classification extends the 1D continuum by adding two additional dimensions representing the expressivity of view (EV) and flexibility of controller (FC). Expressivity would denote how expressive or intuitive the additional data provided by the MR system is. In that sense, a simple logging of data will provide low expressivity, while new features, surveys, and visualisation of the data will provide a high one. The flexibility of controller indicates how well the MR system lets the user influence higher level behaviours of the system without the need for defining low-level commands.

Conversely, it will be low if one provides e.g. direct joystick command, while it will be high if one is able to influence the higher-level beliefs and intentions.

Walker *et al.* present a full taxonomy of HRI based on VR,AR and MR systems [10]. The taxonomy is based on the the type of interface implemented, and is divided into four main categories of virtual design elements: virtual entities, virtual alterations, robot status visualisation and robot comprehension visualisation. An interactive taxonomy framework for many of the works relevant to this thesis, including those of the author, are available under: `https://vam-hri.mybluemix.net/#/`.

Suzuki *et al.* present an impressive survey of 460 papers between the years 2000 and 2021 concerning the joint use of AR and robotics [11]. The papers are classified and evaluated based on eight criteria: I) Approach - location of AR device and weather the robot or the robot's environment is augmented II) Characteristics of augmented robots - form factor (arms, drones, humanoid etc.), number of humans vs. the number of robots, size of the robot, proximity of human-robot interaction III) Purpose and benefits - support programming, support real-time control, improve safety, communicate intent, increase expressiveness IV) Presented information - internal (robot status and capabilities), external (object status and sensor data), plan and activity or supplemental content V) Design Components - UIs and widgets, spatial references and visualisation, embedded visual effects (anthropomorphic effects, virtual replicas and texture mapping effects) VI) Interactions based on the level of interactivity (only output, explicit, implicit indirect, implicit direct) and modalities (tangible, touch, controller, gestures, gaze, voice, proximity) VII) Application domain and VIII) Evaluation strategy. The paper presents the findings of the survey as well as future opportunities.

2.2 Augmented Reality Hardware

Implementing the concepts discussed in the previous section can be achieved in several different ways. The discussion will be centered around around the different ways of overlaying digital data to the real world, shown in Table 2.1 (The original table was presented by Hirokazu Kato during the second VAM-HRI workshop [12]). The implementation is divided by portability (head-mounted, handheld or static) and by the display technology (optical see-through technology, camera-video pairs also called video see through, and projectors). For each portability and display technology benefits and disadvantages are listed. For each combination the availability of such technology is also shown.

Static systems benefit from lack of restrictions in regards to weight and size. On the other hand, though such systems are appropriate for specialised work cells, they are immobile and scale poorly with the number of robots.

Static video see-through systems were the first AR systems to be used in robotics. Milgram proposed a video see-through system to help human operators during robot teleoperation task in 1993 [17]. This consisted of a stereo camera system in the robot working environment which could provide depth information to the operator such as a virtual tape meter that can then be used between two virtual pointers or a pointer and the end-effector, as well as a virtual tether between the pointer and the end-effector to help guide the operator. This was extended in [18] to include a stereoscopic display. Such systems were also intended to be used for telemanipulation tasks on the international space station [19]

Table 2.1: Overview of different types of AR technology with availability, benefits and limitations. Note that this is not confined to robotics but general AR systems, just to display technological availability, benefits and challenges.

	Optical see-through	Video see-through	Projector	
HMD	HoloLens	HTC Vive Pro	Rare, research [13], [14]	+ Mobile, Hands Free - Bad Form Factor
Handheld	Conceptual	Smart Phones	Rare, research [15], [16]	+ Mobile, Input technique - Not Hand-free
Static	HUDs	Any camera and monitor	most projectors	+ Multiple Users - Not Mobile
	+ Natural Observation - Difficult Calibration, Transparency	+ Easy Composition - Image Quality	+ Fit for Surfaces - Surface Colour, Bright Environments	

and in nuclear rectors [20]. More recently such telemanipulation tasks are usually coupled with a VR headset to give full and immersive 3D representation of the environment, such as in our paper [21] where the user is presented with a full, real-time, 3D view of the environment which can also be extended with other information. A recent proposal in [22] is too use two cameras and a two screens on a large-scale telepresence robot for mixed in-person and remote meetings. Though effective at telemanipulation tasks, such implementation is unsuitable for the goal of enabling HRI in shared spaces.

On the other hand, static projector-based systems have a long history of being used in shared human-robot environments. These where usually combined with motion capture systems to track pen-like input devices to define collision free spaces [23], planning and editing trajectories [24], and digitisation of objects (scanning of surfaces) [25]. The major downside of such system is that they are innately immobile. This could be mitigated by automatic, adaptive calibration approaches and easily movable hardware. Still such an approach is quite cumbersome.

Static, optical see-through displays found their use as HUDs in vehicles. There is currently no research on the use of such systems in robotics, however the system would suffer from all of the previous drawbacks mentioned for static systems.

Handheld systems are portable and may be used to interact with different robots. They suffer from the restrictions of weight and size inherited from their portability and the fact that hands are required to hold and interact with them.

Handheld systems are mostly relegated to video see-through devices, consisting almost exclusively of smartphones and tablets using purposely built AR Software Development Kits (SDKs) such as ARCore or ARToolkit. These tool kits have seen increased development providing ever improved capabilities. At the end of 2019, ARCore announced the introduction of spatial mapping capabilities, bringing smart phone based AR closer to capabilities of HMDs, if still not as precise. Smart phone based AR is particularly adapted to interactions in home and service robotics due to the ubiquity of such devices. For example Sprute *et al.* have proposed the use of a tablet to define no-go zones for vacuuming robots [26], [27]. Although the researches originally used a Google Tango device equipped with a depth sensor, the needed capabilities are already available in newer editions of SDKs.

See-through handheld displays are being researched by smartphone companies, most notably Samsung, but none are available at the moment of writing.

Handheld projectors are interesting for collaborative AR interactions [15], [16] as it comes out of the box. Some research has gone into developing HRI using handheld projectors,

be it for games [28], mobile robot commands [29] or general smart home device control [30]. The interaction through handheld projectors are somewhat more complex and cumbersome than other methods as they lack an intractable screen. Moreover still suffer from the common problem of handheld devices - the hands are not free to perform tasks.

Head mounted devices, HMDs, are fully portable and provide intuitive display of information while keeping hands free to manipulate objects, although hands are also commonly used to interact with the device.

Head mounted projectors, similar to the hand-held versions, have the innate advantage of on-site hologram sharing and collaboration. On the other hand they suffer from the standard deficiencies of projectors - bad performance in bright environments and dependency on surface geometry and color. This technology is currently being developed by some research groups, such as the devices by Rolland *et al.* [14] and Cortes *et al.* [13]. As such they didn't see much use outside the communities developing the device.

The first generation of head mounted video see-through devices where VR headsets adapted with stereo-cameras such as the ZED Mini to display the input of the two cameras mixed with virtual elements. These devices had outside-in localisation, meaning that it used external sensors to localise itself. These devices where also tethered and thus had limited work spaces. Since then stereo cameras have been integrated into the devices themselves. Most headsets can also now be equipped with wireless adapters. Although outside-in tracking is still used to to higher precision, some devices such as the Microsoft Mixed Reality Headset family of devices demonstrated complete inside-out tracking, although the device itself was tethered. Such devices can also display any combination of virtual and real elements from the RV-continuum. Although this work used HMD with optical see-through displays all of the developed concepts and modalities can easily be implemented in any wireless VR headset possessing a stereo camera and inside-out localisation. One downside is that, unlike optical see-through displays, the real-world information is captured through cameras and displayed on the screen, meaning that the field of vision and user comfort is somewhat diminished.

Head mounted optical see-through displays have progressed significantly since the first Google Glass release in 2013. Google Glass didn't possess localisation or spatial mapping capabilities and had a single optical combiner, meaning 3D holograms couldn't be generated, relegating the device to a HUD. With the introduction of Microsoft HoloLens and later Magic Leap One full localisation and mapping capabilities, as well as optical elements capable of producing truly 3D holograms has been achieved. These devices are also untethered, integrating all the needed hardware on-board. This means that in theory they have an unrestricted work space, although they are not guaranteed to perform well outdoors. This restriction doesn't impact the concepts and the algorithms proposed in this work. The newest generation of HMDs such as the HoloLens 2, improve the field of view where the holograms can be projected as well as providing better resolution cameras and depth sensors. In addition full hand skeleton tracking as well as eye tracking was added. In this work, the first generation of HoloLens was used, though the concepts can be easily migrated to the newer generation of the HoloLens. Though most of the presented methodology is not dependent on the generation of the device, the applicability of the new features present in HoloLens 2 will be discussed in relevant sections, when applicable.

In conclusion, although many different combinations of hardware exists to implement augmented reality interactions in a robot working cell, only optical see-through devices

provide the portability, flexibility and capabilities to implement the desired system while not hindering the worker in the execution of their tasks.

2.3 Microsoft HoloLens

This section aims to list the hardware and general capabilities of the Microsoft HoloLens that are most relevant to the presented work. This will cover the sensors, displays and general Human-Machine Interface (HMI). The HoloLens is an untethered, head worn device. The see-through display based on optical waveguides allows digital objects to be blended with the real world perceived through the see-through display. The device is able to construct a 3D map of its operational area and to know its location inside the map itself. The HoloLens can detect five hand gestures as well as voice commands, which can be used to interact with the device itself

2.3.1 Sensors and Data

A detailed review of the overall sensor capabilities and the precision of each sensor is presented in [31]. A short overview of the sensors, their inbuilt function and their output data is visible in Table 2.2. Their location on the device can been seen in in Fig. 2.2.

Table 2.2: The sensors of the Microsoft HoloLens, their inbuilt functionality, the data output and the FPS of each data stream.

Sensor	Inbuilt Functionality	Data	FPS
IMU	Localisation	N/A	N/A
4 Wide-angle Cameras	Localisation	6-DOF Location	30
		Grayscale Image	30
RGB Camera	Mixed-reality video capture	1.2 Mega Pixel RGB Image	30
Time of Flight (ToF) Depth Sensor	Environment Mapping Hand Tracking	Short-throw Reflectivity Image	30
		Short-throw Depth Image	30
		Long-throw Reflectivity Image	1-5
		Long-throw Depth Image	1-5
		3D Hand Position	30
		Gesture	30
		Spatial Mesh	>1
4 Microphones	Voice Commands	Audio	N/A
Ambient Light Sensor	Adaptive Brightness	Illumination	N/A

As one can see the amount of sensor data is considerable, allowing the use of numerous different algorithms. Unfortunately the raw data of the IMU remains unavailable. Most of the sensor data was only made available with the introduction of the research mode. Here several examples of how to leverage these sensor data will be presented.

The most obvious is leveraging the localisation capabilities of the device. The tested localisation accuracy of the HoloLens is around 2cm. We arrived to similar error estimation both in autonomous warehouses [32], by estimating hologram error and referencing [33] and by tests with infrared marker tracking [34].

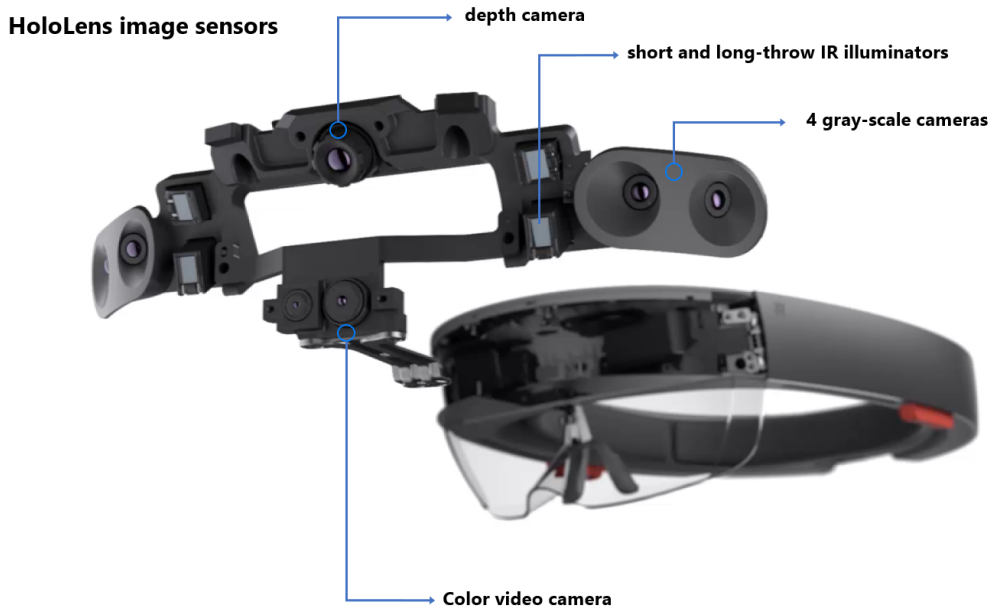


Figure 2.2: HoloLens cameras, including the depth sensor. Sensors also include an IMU, not pictured here. Taken from the Microsoft research blog [2]

These results were also corroborated by [31]. The localisation feature is used to keep the virtual holograms in the same position in relation to the HMD. Provided a coordinate transform between the HMD's coordinate system and the coordinate system a real world object exists, holograms can be overlaid on real objects. The exact methods of obtaining coordinate transforms will be discussed in **Chapter 3.2**. If the real world object is an industrial robot, the position of the user in the robot coordinate system can be tracked inside-out, which is paramount for secure HRI.

The four environmental cameras, together with the IMU, are used as the basis for the in-built Simultaneous Localization And Mapping (SLAM) algorithm that allows the HoloLens to localise itself. These can also be used to reconstruct the 3D environment, using Structure from Motion (SOM) and/or disparity mapping from stereo camera pairs. They can also be used for other machine vision application, e.g. object recognition, though the Red-Green-Blue i.e. Colour (RGB) camera is most commonly used in that area.

The filtered reflectively streams are the basis of calculating the depth streams, which the device does on its own. Interestingly this stream may also be of use in tracking IR markers on tools or robots, that are themselves lit by the IR emitters of the depth sensor on the device itself.

The depth stream, together with the device location, can be used to create the point cloud of the environment from scratch. The short-throw depth stream is particularly useful for hand tracking.

The RGB camera is most useful for object detection, adding color information by combining the color data with the depth stream, and finally may be used for hand tracking by algorithms that rely on RGB data to patch the issue of the HoloLens only tracking four gestures.

The spatial mesh itself can be used as yet another source of the environmental point cloud through mesh sampling.

The hand tracking can be used to track the position of the hands in the HoloLens coordinate system. This can be useful e.g. as a way to track the users hands during proximal HRC. Though newer devices such as the HoloLens 2 offer better hand tracking capabilities including full skeleton tracking, the 1st generation of the HoloLens tracks only 5 gestures. Still this capability was use as a proof of concept for the applicability of hand tracking in HRI and can be easily adapted to newer hand tracking methods.

2.3.2 Inputs

The following input modalities are supported: gaze, gestures and voice commands. Newer devices such as the HoloLens 2 also include eye tracking as a possible input modality.

Gaze is simply the middle-point of the Field of View (FoV), where the cursor is located. The location of the cursor combined with gestures indicates which virtual object is interacted with. The gaze itself can also be used to locate the focus of attention. For example a persistent gaze at a digital twin of a robot can invoke a window with the current task data that the robot is performing. Newer devices may offer eye tracking capabilities which could than be used instead of the gaze for better performance.

The HoloLens 1 supports only five gestures: bloom, ready, tap, hold and drag. The bloom is used to to invoke the main menu and exit apps. Therefore it is normally not usable. The ready gesture is performed by closing all fingers except the index finger which should be pointing towards the ceiling. Lowering and quickly raising the index finger performs the tap gesture which would be a left mouse click on classical computers. Holding the index finger down performs the hold gesture, which is again equivalent to holding down the left mouse button. The drag gesture is performed by moving the hand while performing the hold gesture and is again equivalent to the mouse example. One thing to note is that the HMD only tracks gestures and not the hands themselves. If full hand tracking is needed it should be implemented with a separate algorithm based on camera or depth sensor input, or a newer device can be used.

Regarding voice commands, the HoloLens supports a speech to text engine to execute specifically define actions e.g. "Launch" to launch an application, "Take a picture" to make a mixed reality screenshot etc. These commands are mapped one to one and there is no language understanding capabilities.

2.3.3 Data Display

Two main ways to display data to the user exist - holograms and spatial sound. Holograms are digital objects perceived in 3D from stereo image pairs produces by optical waveguides. The waveguides not only transmit the picture but also let ambient light through, allowing for the mixing of real and digital worlds. This technology has the issue of reducing the FoV The FoV of the HoloLens is perhaps its biggest drawback, being only 30° horizontal and 17.5° vertical. This has already been improved by both Magic Leap One (40°x30°) and HoloLens 2 (43°x29°).

The spatial sound can create a virtual, moving source of sound through the use of the speaker array.The source of the sound will be perceive as moving by the user, both in direction and distance. This can be a very useful addition to deal with the limited FoV

when applicable. An example here would be to position a spatial sound source on the end-effector of an industrial manipulator so that the user knows its location even though they cannot see it.

2.4 Wearable-based HRI

Wearables is a shorthand term for wearable computers. Any device that is worn on the body that possesses computational capability may so be defined as a wearable. Modern wearables ubiquitously possess sensors and communication capabilities. Likewise, in this work, wearables will need to possess these capabilities as they need to interact with each other to form a system.

The use of wearables to interact with robots became significant with the modern development of smartphones and smartwatches. Being able to use "off-the-shelf" components greatly increases the flexibility and adoption of such systems, while reducing cost.

A generic wearable system to interact with robots was never proposed, however *Kemp* [35] did envision a future where wearable computing and sensors would allow the robots to experience the world through the human perspective, thus facilitating robot learning. Although in this work wearables are not generally used to teach robots through deep learning, the main concept of bringing the robot and human perspectives closer together is an important one. In this work the robot not only gets useful data from the human perspective, but also vice-versa, the human gets useful data from the perspective of the robot. In that sense the wearable system acts as a translator between machine-readable and human-readable data.

2.5 Software Libraries and Packages

To implement the entire system, a large number of different software packages and libraries were used. In this section the most generic ones, used virtually in every application, will be listed. They represent the state of the art in the sense that these libraries are likewise used in the vast majority of other papers. More specific application-dependent libraries and packages will be mentioned in the *Implementation* sections of their respective chapters. The list here doesn't represent the only possible choices of components. For instance the Unreal Engine can be used as the 3D engine for the AR visualisation instead of Unity3D.

Unity3D is perhaps the most popular SDK and engine for developing computer games, 3D visualisation and mixed reality applications. It supports both 2D and 3D applications and allows applications to be deployed on various platforms. In this work, Unity was used to implement all of the visualisation components on the Microsoft HoloLens. Together with Microsoft's Mixed Reality Toolkit (MRTK) it is possible to interact with the localization and environmental map of the HoloLens itself. The Student and personal editions of Unity3D are readily available to download: `https://store.unity.com/#plans-individual`.

The **MRTK** was originally develop by Microsoft as a set of tools for developing HoloLens applications using the Unity3D game engine. Over the years this has been extended to include Windows Mixed Reality Headsets, HoloLens 2 as well as support for other devices such as the Meta Quest. Using MRTK allows: implementation of GUIs, holographic overlays and objects; use of voice commands; interaction via hand tracking and gestures; accessing the spatial mesh. It also offers communication capabilities with other devices, such as desktop PCs. To access raw sensor data and implement machine vision algorithms, HoloLensForCV, which will be described in the following subsections is also necessary. The package is available under: `https://github.com/microsoft/MixedRealityToolkit-Unity`.

OpenCV is a collection of libraries intended for computer vision - processing and analysing images. The applications range from object detection to gesture recognition, structure from motion, ego-motion detection etc. It is an open-source cross-platform collection written in C++. In this work OpenCV is mostly used as part of the HoloLensForCV to process and work with the raw camera data produced by the HoloLens, be it greyscale, RGB or depth. To offload some computational weight it may also be implemented on a desktop PC to which raw sensor data is streamed. The repository is found under: `https://github.com/opencv/opencv`.

HoloLensForCV is a collection of libraries based on OpenCV intended to ease the access to the HoloLens' image sensors (the RGB and grayscale cameras as well as the ToF depth sensor) and visualize their data. Some basic image processing is also available. The package can be found here: `https://github.com/microsoft/HoloLensForCV`

The **ROS** is not a standard operating system per se, but a collection of software frameworks intended to act as middleware when developing robotic applications. It contains hardware abstractions, low-level device control and communication between heterogeneous processes, as well as an open-source collection of most commonly used algorithms and interfaces. It is built for use on Unix operating systems. An important consideration is that ROS doesn't naively provide real-time capabilities, though it is possible to integrate it with real-time code. This was addressed with ROS 2 which offers real-time and embedded code support. Throughout this work only the original ROS will be used. The latest distributions of ROS can be found under: `http://wiki.ros.org/ROS/Installation`.

As already mentioned, ROS possesses numerous software frameworks and here some of the most ubiquitous ones will be mentioned - **RViz** is a highly customizable 3D visualization framework able to visualize objects, robot geometries, paths, joint states, sensor data etc.

MoveIt is a software framework dealing with robot kinematics and dynamics as well as path and action planning. Scenes containing all the objects in the robot's workspace can be loaded and collision free paths planned. Time parametrization, planning constraints, integration of 3D sensors, multiple arm support etc. are all also available.

ros_control is a set of packages that include controller interfaces, controller managers, transmissions and hardware interfaces.

tf2 is a server that keeps track of all the coordinate frames and allows easy transformation between them.

The **urdf** is a robot description language intended to ease the definition of geometries, visual meshes and collision meshes of robots. In addition, objects in the robot workspace can also be defined.

The **ROS Bridge** package provides interfaces with processes running on different machines or operating systems. It is used to support communication between the HoloLens and the desktop computer.

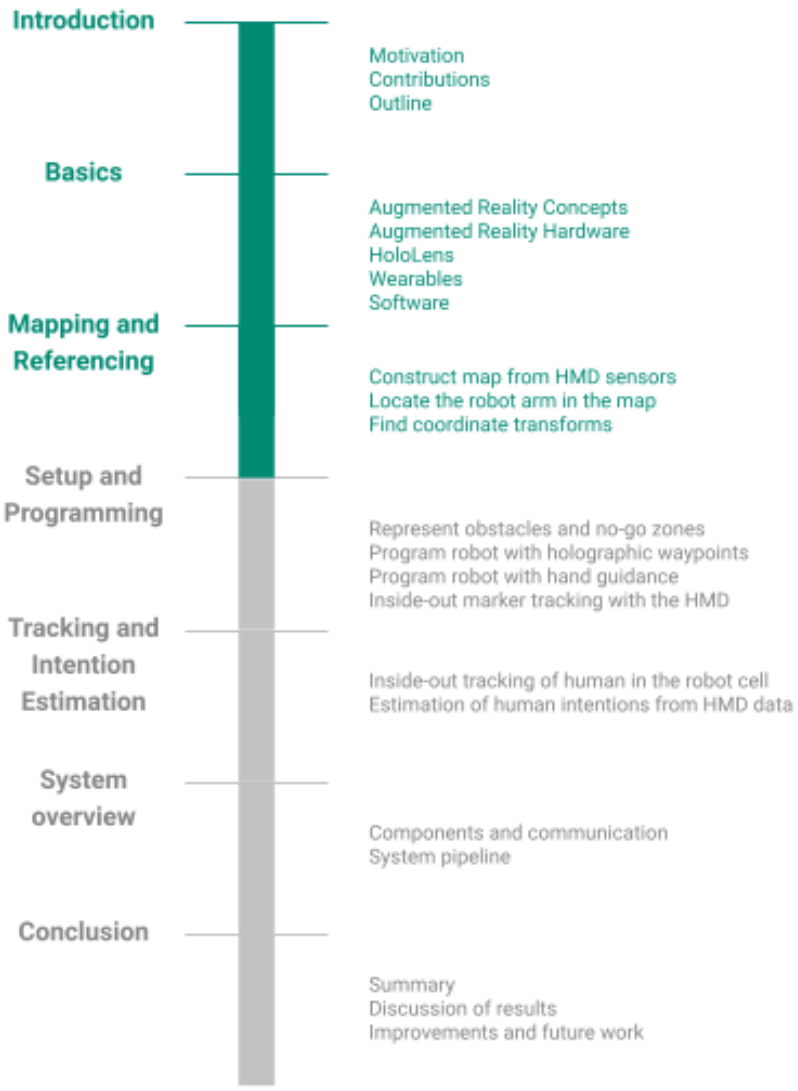
ROS# (ROS Sharp) is a collection of libraries in C# developed by Siemens for interfacing ROS and ".NET" application, particularly Unity. It contains, for example, ROS topics implemented as classes, publishers, subscribers and autonomous generation of robot models in Unity from urdf files. The repository can be found here: `https://github.com/siemens/ros-sharp`

The **Point Cloud Library (PCL)** is a collection of libraries that provides numerous ways to process point clouds and depth information. this includes generating point clouds from stereo and depth sensors, registering different frames, object detection and correspondence matching, outlier removal etc. It can be found for download here: `https://pointclouds.org/downloads/`

2.6 Conclusion

In this chapter we endeavored to answer the question "What is AR?" and how does it relate to similar concepts such as VR and MR. The categorisation of interactions between robots and humans using MR has also been shortly presented. Next we described how it is possible to implement AR using different technologies, most of them readily available off the shelf, such as HMDs and smartphones. We argued our choice of using a HMD to implement the proposed system. We described the capabilities and hardware of our HMD of choice - the Microsoft HoloLens. We then briefly discussed how other wearable technology might be used to supplement the capabilities of HMDs. Finally we introduces the software packages and libraries that were ubiquitously used in the development of the components necessary to achieve our vision of a portable end-to-end HRI system. The next section will present the first major building block of the envisioned system, namely how to map the environment using the HoloLens and find a coordinate transformation between the robot's frame of reference and the frame of reference of the HMD.

3 Mapping and Referencing



In the previous chapter we presented the basics of AR as a concept. We then described different possible implementations of AR in hardware. From the available possibilities we chose to use a HMD, the Microsoft HoloLens. We then quickly went through the standard software packages used to develop HRI based on HMDs.

This chapter will cover methodologies of obtaining a map of the environment in the form of a point cloud from the sensors of the HoloLens. This map is not only of use for the HMD, but can also be shared with the industrial robot as an additional sensory input. Knowing the location of both the robot and the HMD in the joint environment, as well

as the coordinate transformation between them, is paramount for any robust interaction modality.

First, we will describe how to obtain a usable map of the environment from the HoloLens. This map will then serve as the basics for the referencing, workspace setup and robot programming blocks. Focusing then on referencing, we will give a very short overview of how this problem is solved in the wider robotics community. Then we will present the state of the art in referencing between an HMD and a robot to have a baseline and show how the methods which will be presented here differ from other published methods.

We will then present several algorithms we developed for referencing and describe different tests to estimate the quality of our referencing algorithms.

3.1 Mapping

Mapping refers to the ability of constructing a usable map of the device's environment. Maps can take many different forms, they may be 3D or 2D, they may be discrete, they may contain semantic information, etc. In this work maps will be 3D representations of locations of solid objects in the environment - walls, floors, tables, robots and so on. This may take the form of a point cloud, basically a collection of points each belonging to a solid object, or a voxel grid which represents the environment as a grid of cubes, with each cube either containing a solid object or not.

Mapping is the first step in our system as the map of the environment is the basis for the coordinate transform and workspace representation used in virtually all the later components. The user walks around the workspace of the industrial robot (while the arm is static) to build up the map of the environment. We also assume the scene is completely static at this point. This process can be repeated multiple times, but once finished, no other step will change this basic map. An example of a workspace of a robot can be seen in Fig. 3.1.

3.1.1 HoloLens Mapping Capabilities

In Section 2.3.1 we have already mentioned that the HoloLens has in-built depth sensors. These sensors are used by the HoloLens to create a 3D mesh representation of the environment. This mesh is accessible using Unity and MRTK (Sec. 2.5). The user may choose how precise the generated mesh should be, that is the number of triangles, however the process of creating the mesh is not open. Indeed since the release of the HoloLens in 2016 until the research mode was available in 2018, this was the only way to access the map of the environment. The research mode opened-up the raw sensor data of the HoloLens (except from the IMU data), allowing the depth stream to be used by the researchers to create point clouds from scratch. It also allows the use of the images of the four environmental cameras. The sensor data can be accessed by using the HololensForCV software package described in Section 2.5.

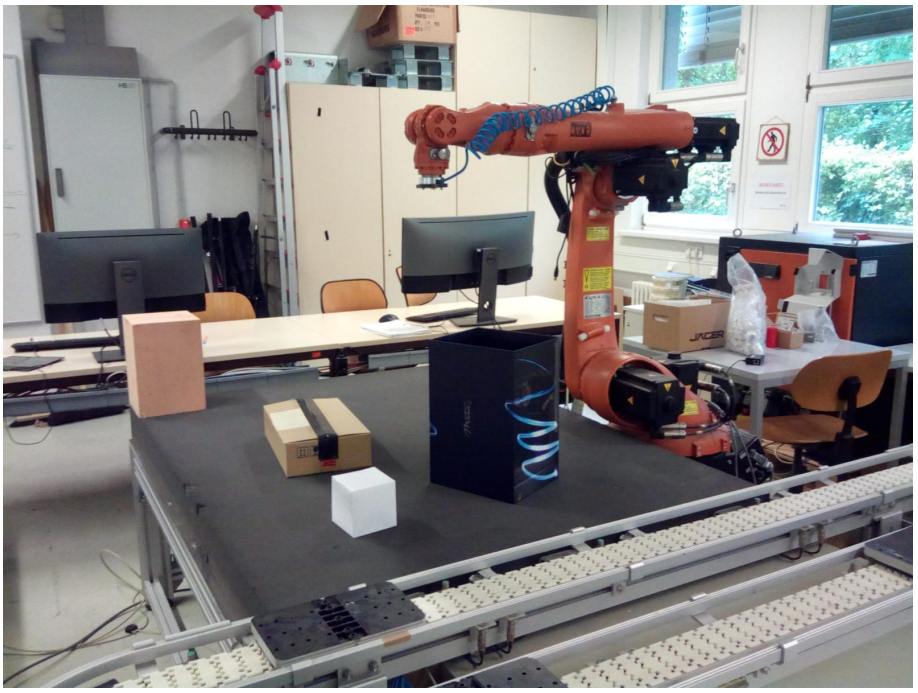


Figure 3.1: An example of the workspace of a robot. A similar setup with the robot, table and conveyor has been used in other tests throughout this work.

3.1.2 Spatial Mesh Sampling

The first approach uses the mesh of the environment already generated by the HoloLens. The user would walk around the robot's workspace and the HoloLens will build the mesh automatically. Once the mapping is complete, the entire mesh of the environment is read from the HoloLens' memory. Randomly a mesh triangle is chosen, weighted by the size of the triangles. The probability p_i of triangle i to be chosen is the area of triangle i divided by the sum of the areas of all the triangles in the mesh, as shown in Eq. 3.1. Then, using barycentric coordinates, a random point within the triangle is selected and saved to the point cloud. The number of iterations of this process, and therefore the size of the resulting point cloud, can be chosen. By making larger triangles more likely to be selected, one obtains a more uniform sampling of the whole area of the mesh. The resulting point cloud is then filtered with voxel grid filtering to obtain a uniform point density.

$$p_i = \frac{A_i}{\sum_{j=1}^{j=N} A_j} \tag{3.1}$$

This method of obtaining the point cloud was quickly replaced by the one based on the direct depth sensor data once it became available, as it allows greater flexibility and control of the filtering to generate a better point cloud. However, this method may still be of use on devices that do not allow direct access to depth sensor data, or do not have a depth sensor altogether but still generate a spatial mesh.

3.1.3 Reconstruction from the Depth Stream

After Microsoft allowed access to the depth stream with the research mode, the raw depth data could be used and the point cloud generated directly. The depth sensor of the HoloLens provides two depth streams, the short-throw depth stream, with 30 frames per second update rate and a range of 0.2-1 meters, and a long-throw depth stream, with 1-5 frames per second update rate and a range of 0.5-4 meters.

Combined with the localisation capabilities of the HoloLens, the different depth frames can be fused into a single point cloud of the environment. We registered the point data of different frames to the main point cloud using the HoloLens' own localisation as the initial guess and ICP [36] to refine the guess. It was found, however, that ICP does not significantly increase the precision (see Table 3.3). Therefore the registration step may be skipped.

We then discard points that are below the minimum cut-off distance to eliminate points that may belong to the user's hand, and above the maximum cut-off distance to eliminate low quality points. The maximum cut-off distance was experimentally determined to be 3.3 meters (Fig. 3.2). The minimum cut-off distance was taken to be one meter, around the reach of the user's arms. Therefore we can use only the long-throw stream and discard points further than 3.3 meters.

The resulting point cloud is down-sampled using voxel-grid filtering to ensure uniform point density. It is then filtered with an outlier removal filter, removing any point that had less than 9 neighbours in a radius of 5 cm, and smoothed with MLS [37]. Finally Random Sample Consensus (RANSAC) [38] plane detection is used to detect planes and map all the points near the plane to the plane itself. This improves the resolution of objects on floors and tables.

3.1.4 Reconstruction from Stereo Camera Pair

The HoloLens also possesses four greyscale environmental understanding cameras, which form two stereo image pairs. The two front most cameras are quite well positioned to extract a stereo image pair.

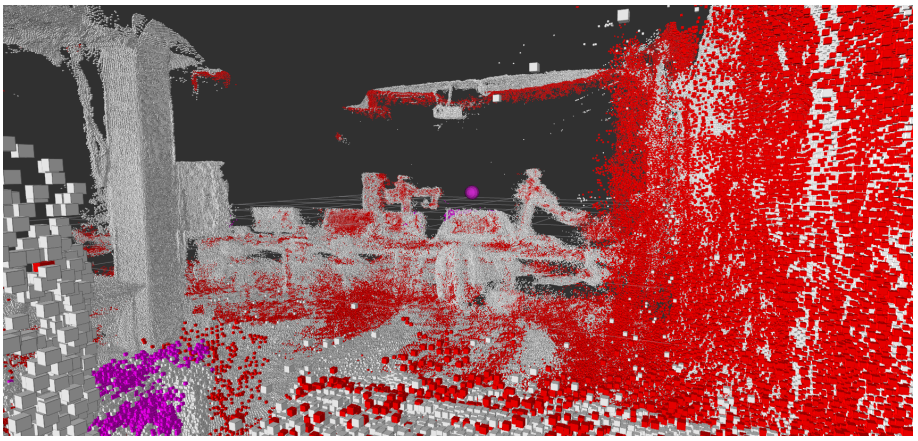
To get a depth image we need to calculate a disparity map from the two images. We define disparity as:

$$disparity = x - x^{'} = \frac{Bf}{Z} \tag{3.2}$$

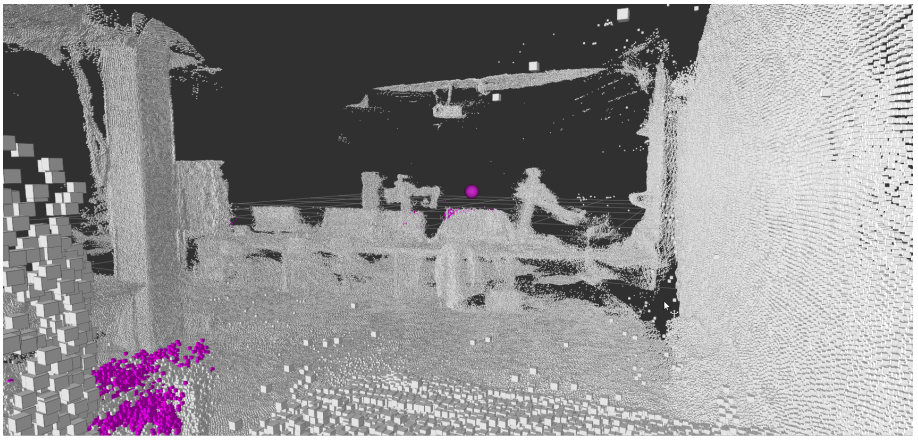
Where B is the distance between the cameras, also known as baseline, f is the focal length of the cameras (assumed to be the same), Z is the distance of a 3D scene point X from the cameras, while x and $x^{'}$ are the projection of the 3D point into the camera plains respectively. One can see that if we can find the two projections of the scene point while knowing the baseline and the focal length, we can calculate the distance of the scene point. The task now is to determine which points in the two images correspond to same scene points. OpenCV offers two methods of achieving this task, Block Matching [39] and the Semi-Global matching [40]. We used the latter together with a Weighted Least Squares filter to improve accuracy.

Issues were unfortunately found with this approach. The disparity map calculation is computationally expensive. Secondly there were issues in synchronising the two cameras. The images would normally be one frame apart. A two thread system was implemented, with each thread reading data from one camera and saving it in a 5 bin buffer. The buffers were compared to find two images with the minimum timestamp difference, however this did not solve the problem. A comparison between a point cloud reconstructed from depth sensor data and one generated from the disparity map of the two frontal environmental cameras is visible in Fig. 3.3. As can be seen, in the central area the stereo approach generates a robust and dense point cloud, indicating that the algorithm may be feasible.

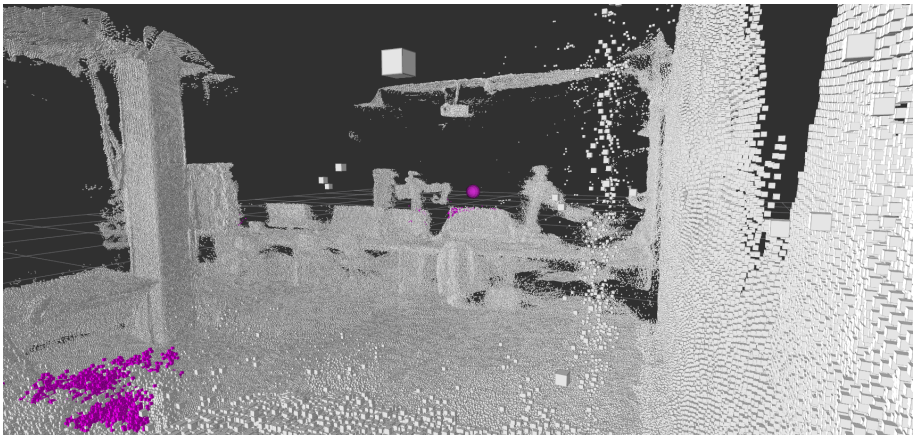
Although this approach did not work on the HoloLens due to frame synchronisation issue, it is still worth considering for future applications. Newer HMDs such as the Apple Vision Pro abandon the use of depth sensors in favour of a large number of conventional cameras. In such cases, especially if the device does not provide a mesh of the environment, this approach should be seriously considered.



(a)



(b)



(c)

Figure 3.2: Distances were measured at the time the depth frame was captured and do not represent the distances from the current viewpoint. (a) The mapped point cloud with points more than 3.3 meter distant from the depth sensor in red. Points at less than 1 meter are marked in purple; (b) Point cloud with points more than 3.3 meter distant removed. One can see the presence of sparse outliers that can be filtered out; (c) Point cloud with points more than 3.4 meters away removed. The outliers are much denser requiring more aggressive filtering which may degrade the quality of the inliers.

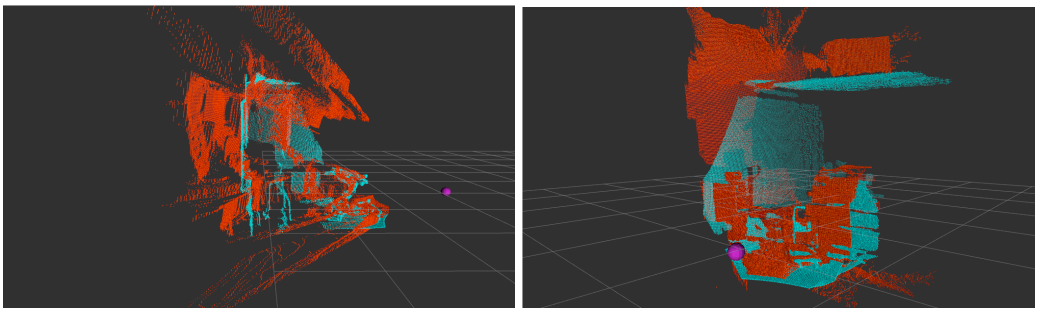


Figure 3.3: Point cloud constructed from a disparity map (shown in orange) compared to a point cloud constructed from depth sensor data (shown in cyan). One can see large errors caused by improper synchronisation of the frames. The picture on the right shows a best case scenario and the potential of such a stereo approach

3.1.5 Mapping precision

Using the depth sensor data to map the environment was the most sensible solution. In 2.3.1 it was mentioned that the HoloLens' localization accuracy is around 2cm, depending on the specific environmental conditions. It is to be expected that the lower boundary for the accuracy of a map generated by the HoloLens in motion would also be similar. Here we present experiments intended to test the accuracy of the depth sensor as well as the generated map itself.

To test the influence lens distortion, a flat surface was positioned at 1 and 2 meters respectively from the HoloLens' depth sensor. The HoloLens was rotated so that one of the selected five pixels(center, top, bottom, left, right) lies on the surface. Afterwards the HoloLens remained stationary. A total of fifteen consecutive depth frames were taken for each pixel and each distance for a total of 150 measurements. The standard deviation of the depth measurement fluctuations around the average was found to be 3 mm and the maximum fluctuation around the average 5 mm. Thus it was concluded that the error due to distortion is minimal.

Next, to test the accuracy of the depth sensor, the experiment was repeated by taking the center 5 pixels and averaging their distances over 5 frames. The experiment was setup and conducted five times to average out possible human error. The results can be seen in Table 3.1. One can see that the maximum error at 1 m is 7.5 mm and at 2m is 8mm. The accuracy of the depth sensor seems adequate for constructing robust maps.

Table 3.1: The observed averaged depth values for each repetition of the depth sensor accuracy experiment. Each value is the average of 5 pixels over 5 frames.

	Repetition 1	Repetition 2	Repetition 3	Repetition 4	Repetition 5
1 Meter	0.998400	1.007512	0.998728	1.005440	1.003424
2 Meters	2.011248	2.006576	2.005984	2.008064	2.000224

Next we took scans of the Robot hall in the Intelligent Process Automation and Robotics group building (Engler-Bunte-Ring 8, 76133 Karlsruhe) using a Faro Focus^S laser scanner with 1 mm precision as the ground truth. We compared it to a point cloud generated by the HoloLens. The HoloLens' point cloud was tested with four different combinations of using registration or not and using the post-processing step or not. ICP was used for registration, while the post-processing step consists of MLS smoothing and RANSAC plane detection and projection. First, we selected the parts of the environment where the two scans overlap (see Fig. 3.4) and measured the average Euclidean distances and the Hausdorff distances, the greatest distance between two closest points in the two point clouds, of the four combinations. The results are presented in Table 3.2. One can see that apparently the post-processing step introduces a bigger error. The large Hausdorff distance can likewise be attributed to left-over discrepancies in the two point clouds either as a result of missed holes or the fact that the point clouds are taken at slightly different time.

Table 3.2: The average Euclidean distances and the Hausdorff distance between the laser scan, ground truth point cloud and the HoloLens' point cloud

	Without ICP, not postprocessed	With ICP, not postprocessed	Without ICP, postprocessed	With ICP, postprocessed
Euclidean distance $[m]$	0.040128	0.040742	0.061583	0.062344
Hausdorff distance $[m]$	1.052818	1.054748	1.172284	1.169257

To get a better estimate of the precision of the two point clouds we used the CloudCompare software. CloudCompare gives the percentile distribution of distances between the two point clouds and therefore offers a much better insight into the quality of the point cloud generated from the HoloLens. The results are shown in Table 3.3. One can see that the best performance is the mapping without ICP and with post-processing. In this case 75% of points have an error of 3.6 cm or lower. A visual comparison of the four point clouds to the ground truth can be seen in Fig. 3.5.

Table 3.3: The percentiles of the distances of each spatial map combination to the laser scan.

Percentile	Without ICP, not postprocessed	With ICP, not postprocessed	Without ICP, postprocessed	With ICP, postprocessed
10th $[m]$	0.00591	0.00629	0.00588	0.00513
25th $[m]$	0.01177	0.01254	0.01135	0.01099
50th $[m]$	0.02192	0.02503	0.02150	0.02582
75th $[m]$	0.04105	0.04573	0.03673	0.04183
90th $[m]$	0.06447	0.06994	0.05275	0.06057

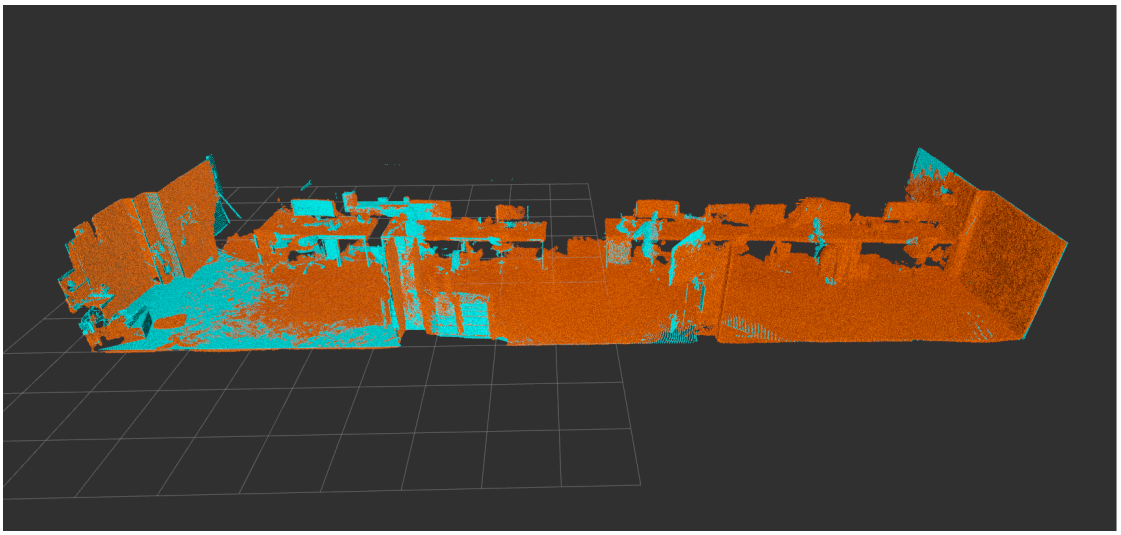
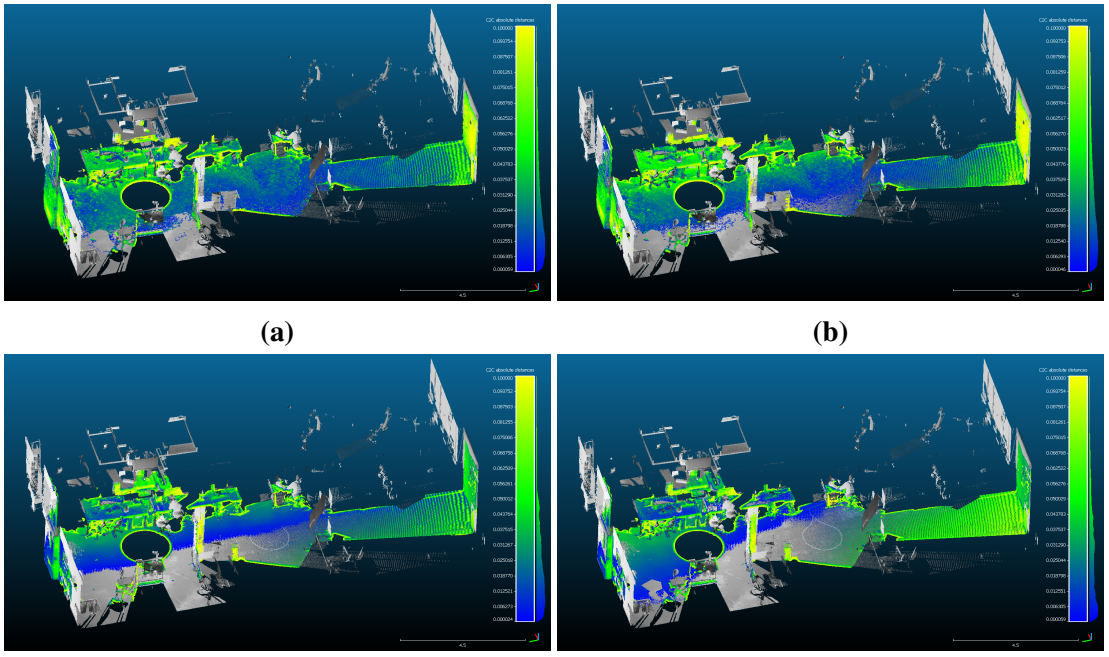


Figure 3.4: The overlapping segment of the laser scan - cyan; and the HoloLens point cloud - orange; used to calculate the average Euclidean distance and the Hausdorff distance in experiment 3.



(a) **(b)**

(c) **(d)**

Figure 3.5: Comparison of the HoloLens point cloud with the ground truth obtained via laser scan;(a) Point cloud without ICP registration between frames and without the post-processing step of MLS smoothing and RANSAC plane detection and projection;(b) Point cloud with ICP registration and without postprocessing; (c) Point cloud without ICP registration but with post processing; (d) Point cloud with ICP registration and with post-processing. Grey points represent an almost complete overlap, blue points have lower error. One can see that the point cloud c) has overall the smallest total

3.2 Referencing and Coordinate Systems

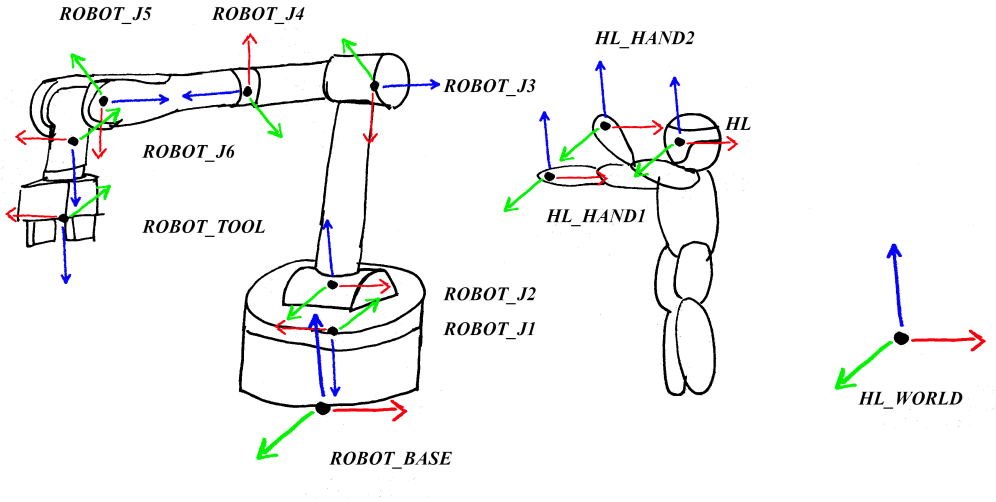


Figure 3.6: The main coordinate systems of the proposed system. One can see they are mainly split between the HoloLens dependent coordinate systems and the robot dependent ones.

Referencing refers to finding the coordinate transform between objects. In our case it is between the current HoloLens position and objects of interest in the environment. This is an important step as referencing is used to properly present and overlay virtual information tied to an object using AR as well as calculating the distances to the objects.

The number of coordinate systems varies based on the application. For our proposed system the main coordinates of interest are shown in Fig. 3.6. They are:

1. The HoloLens World Coordinate System - this is the Pose of the HoloLens when the application starts on the device (HL_WORLD)
2. The Current Pose of the HoloLens (HL)
3. Pose of the robot base (ROBOT_BASE)
4. Pose of each robot joint (ROBOT_J1, ..., ROBOT_J6)
5. Pose of the robot end-effector e.g. gripper
6. Pose of the user's hands (HL_HAND1 and HL_HAND2)

The transformation between the current and world coordinate systems of the HoloLens is done by the device. The user's hand can likewise be tracked by the device when in the FoV of the HoloLens' depth sensor i.e. in front of the user's gaze direction. On the PC side the robot will keep track of the transformations between the base and the joints and end-effector. Thus the key task is to find the transformation between the robot base and the HoloLens' world coordinate system.

Unity, which was used to develop the applications on the HoloLens, uses a left handed y-up coordinate system, while all the applications running on the PC in ROS use the standard right-handed z-up coordinate system. Thus the pose of the HoloLens and the hand

detection trough the HoloLens cameras need to be properly converted before coordinate transformation with the other parts.

On the ROS side the transformations between the coordinate systems of the robot joints are automatically tracked based on the robot model and the joint states published by the robot controller. We use the tf2 package to keep track of all the transformations inside our system. When we determine the fixed transformation between the HoloLens world coordinate system and the robot base at the start of our application, we save it with the tf2 package. Then we can publish the converted HoloLens pose directly to tf2 which will handle all the transformations needed.

3.2.1 State of the Art

Referencing methods may be classified according to two criteria. The methodology used to obtain the coordinate transform and the number of times the coordinate system transform is calculated. The later can either be continuous, meaning the coordinate transform is re-calculated and updated every time relevant sensor data becomes available, or one-shot, meaning the referencing is calculated once at the beginning of the interaction.

Manual referencing, the simplest option, requires the user to position the hologram above the desired object by hand. Such an approach is tedious if precision is required and even then it is often not precise enough. An example of manual referencing can be found in [41]. In [42] there were two modes of referencing using Microsoft's HoloLens, either a completely manual placement or through the use of the spatial mesh. In the later the user places the cursor on a predefined point on the spatial mesh of the robot, namely the middle of the robot base, and clicks to overlay the hologram. This makes the manual positioning easier and more precise as it's constrained on a surface, however it still may produce errors especially if the mesh is coerce at the specified point. Manual referencing is always one-shot.

Most approaches use visual markers [43] [44] [45] [46]. Markers can be either QR codes, ARUCO markers or other 2D visual identifiers such as a sticker or a distinctive texture. In [47] QR markers on a mobile robot were used for an AR overlay over a fixed camera data stream. In [48] a Baxter robot was used, which possesses an in-built screen on which a QR tracking code was displayed. Unlike other marker methods, this requires no additional calibration as the screen is part of the robot model and the transformation is known. Obviously such an approach is restricted to robots with a screen. In [49] YOLO, a deep neural network, was used to generate 3D bounding boxes around objects. However, an ARUCO marker was still used to transform the coordinate systems between the HMD and a robot. These methods require the placement and calibration of visual markers before any kind of interaction can take place. As this is undesirable for our system, other ways of referencing were required. Visual markers can provide both one-shot and continuous referencing, with the later being most common.

Methods relying on external trackers such as [50] are quite precise. In [50] an additional IR marker motion tracking system is used for increased precision and easy referencing between the HMD, in this case a HoloLens, and an object to be spray painted. Although precise, such methods are constrained to specially setup robot cells. This method always provides continuous referencing.

Object matching methods do exist. Commercial systems such as Vuforia and Visionlib offer 3D referencing based on 2D camera data through edge-detection. Specifically, in [51], Vuforia was used to reference 3D objects based on 2D features, placing invisible 3D models and giving the impression of a virtual character interacting with real objects. Such methods still fail for rotationally symmetric objects or objects with few features, such as the cylindrical robot bases. Ostantin *et al.* [52] use a modified ICP algorithm to match the environment map created by the HoloLens and one created by a mobile robotic platform to obtain the coordinate transform. In another paper Ostantin *et al.* [53] use the map generated by the HoloLens to find the robot for automatic referencing, They remove the walls and floor via RANSAC and use DBSCAN [54] to cluster the remaining points. The volume of each cluster is compared to the volume of the model and those clusters with similar volumes are selected. Then the model is registered to each cluster via ICP and the cluster with the smallest error is selected. Though not explicitly stated the error is most probably measured as the average distance between the model point and the closest point in the registered cluster. Mišeikis *et al.* [55] proposed a robot localization and state estimation system based on Convolutional Neural Networks (CNNs). Two CNNs are employed, the first one generates a mask over the RGB image and localizes the robot. The second one uses the output of the first to detect joints and estimate their state. Though not deployed on a HMD, it still has the potential of working on one to provide continuous referencing.

3.2.2 Proposed Referencing Methods

Given the state of the art, concepts for three different methods were developed. The first would require user the user to roughly reference the robot manually after which the guess would be refined by object matching algorithms. The second would rely solely on the map of the environment and object detection algorithms, and no user input would be required. While the previous two methods were one-shot, the third method would be a continuous referencing method.

For all methods we assume we have a 3D robot model, which is almost always the case. We can sample the surface of the 3D model in a similar fashion as described in Sec. 3.1.2 to obtain a point cloud for matching. The first two methods require a static robot to be referenced, while the third method requires access to real-time robot joint states.

The goal is to provide a precise referencing method that doesn't put additional mental load on the user and is general enough to work with different types of robots.

3.2.2.1 Referencing by Refining the User Input

After obtaining the map of the environment as described in Sec. 3.1, the user is asked to position a holographic cube in the base of the robot, and rotate the cube roughly toward the Tool-center point (TCP) of the robot. We call this cube the seed cube. After the user positions the cube, a registration step is performed to refine the guess. Firstly a spherical region around the seed cube is defined and the points outside the sphere discarded. The radius is selected to be twice as large as the largest side of the bounding box of the model. This is quite general as the largest side is usually the height of the robot which are almost exclusively positioned parallel to the ground (but not always). MLS is used to upsample

Figure 3.7: The referenced robot as displayed to the user on the HoloLens alongside the seed hologram and the spatial mesh.

the remaining point cloud. To ensure uniform density of points in both the model and the scene point cloud, a voxel grid filter with the same edge size is run on both point clouds. Next the model is positioned at the location of the seed cube. We test both ICP [36] and Super4PCS [56] to perform the final model matching. As we know the pose of the seed cube in the HoloLens world coordinate system and know the transformation between the seed cube and the final model match (which we assume is the location of the robot), we can transform poses at will between the robot coordinate system and the HoloLens coordinate system.

The user guess is very important as matching algorithms such as ICP have a problem of getting stuck in local minima, needing a good first guess for a good model matching.

As we will see in Section 3.2.3 this method provides very good precision combined with an execution time of about a few seconds (as different computers with different configurations were connected to different robots, it is hard to give an exact number). The mental load on the user is much smaller than other manual methods seen in the state of the art section. The guess does not have to be precise, therefore the user load is minimal.

In Fig. 3.7 one can see the spatial mesh generated by the HoloLens, the seed cube that the user places and the final holographic overlay after the algorithm refined the user guess.

3.2.2.2 Automatic Referencing

Another option is to remove user input altogether. The proposed automatic referencing pipeline is based on 3D descriptors. Descriptors are a way to encode 3D spatial information of an object into a feature space. These features can then be more easily compared and do not suffer from the same issues of being stuck in local minima. That said matching algorithms such as ICP are always used to refine the guess of these algorithms. Descriptors can be global or local. Local descriptors select particularly interesting zones on the 3D object to be found (e.g. rough surfaces, places where curvature changes rapidly etc.). The process is repeated over the whole scene. Then the features of the specific regions are compared between the object and the scene. Geometric constrains between the points on the object and those found in the scene need to be maintained for consistent matching. Global descriptors meanwhile describe the entire 3D shape or geometry of the object. The scene needs to be divided into different objects or clusters using a clustering algorithm. For each cluster a global descriptor is calculated and the descriptors between the clusters and the objects matched. The pose of the most similar cluster is then the pose of the object.

We tested three different descriptors - Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram (OUR-CVFH) [57], Viewpoint Feature Histogram (VFH) [58] and SHOT [59]. OUR-CVFH is a semi global descriptor, it calculates local features but also introduces multiple, repeatable coordinate systems such that an additional step of implementing geometric constrains for matching is not necessary. VFH and SHOT are local descriptors. OUR-CVFH requires clustering, while VFH and SHOT do not, although it would require an additional registration step based on geometric constrains.

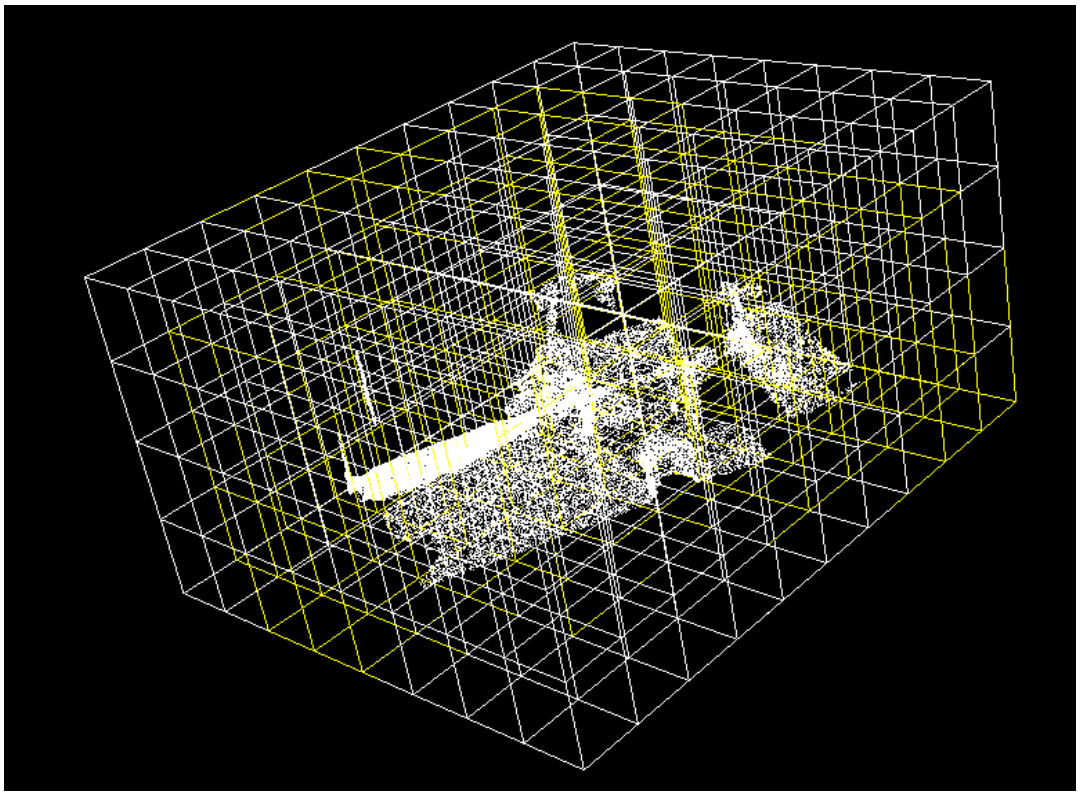
The pipeline is as follows. First, we calculate the descriptors of the robot model point cloud. The robot model can easily be converted into a point cloud using the mesh sampling method described in Section 3.1.2. Next the ground plane from the scene point cloud is removed using RANSAC. Next we slide a box over the scene point cloud. The box is a square with the edge length equal to the largest edge of the robot model bounding box. The box is moved by half it's length first in every axis over the entire scene point cloud. The boxes containing too few points are discarded. The rest have their descriptors calculated. When the box finished sliding over the entire point cloud, the three boxes with the best descriptors are taken as the possible candidates for the location of the robot. For each the robot model point cloud is placed in the center and an ICP registration algorithm used to match the two pointclouds. This is repeated four times for each box for robot model rotations of 0°, 90°, 180° and 270°. The registration with the smallest RMS error between the robot model points and the box points is considered the match. This pipeline allows us to use both local and global descriptors, as well as skipping the tedious matching of geometric constrains.

In Fig. 3.8 one can see the process of automatic referencing. First, the point cloud is searched with sliding boxes (a). If the box contains fewer than three points it is discarded (this can be set much higher to speed-up the algorithm significantly). The yellow boxes were the one having three or more points. For each box, features are calculated and matched to the model (b). In this figure we give an example of the best match between the SHOT features of the model and one of the boxes. One can see the box contains the robot. If one uses the geometric constrains as an initial guess for the ICP, one ends up with the dark blue robot as the match (c). Clearly the initial guess wasn't good enough for ICP. Instead for the selected box we initialize the robot model point cloud in four different orientations and run ICP on all of them, taking the best result as the final match. Such method produces the red robot, which fits the robot in the scene.

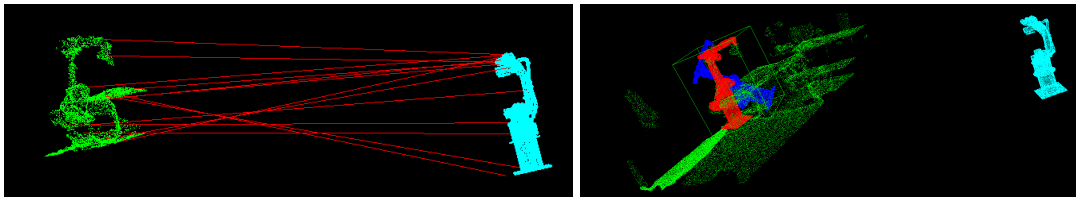
Automatic referencing doesn't require any user input, thus there is no additional mental load for the user. However it adds additional processing time that increases significantly with larger point clouds (up to 2 minutes, though as we mentioned this could be reduced). Also, as we will see during the tests, there was no single descriptor or set of hyperparameters that worked well with all robot types. Though in the author's opinion the refining of the user's guess is a superior method both in time and precision. this method might be used at least for a first guess to reduce mental load even further.

3.2.2.3 Continuous Referencing

The two previous referencing methods reference the robot only once at the start of the interaction. This is prone to error accumulation due to the slight and unavoidable drift in the localisation of the HoloLens. A superior solution would be an algorithm that references the robot continuously whenever possible. We already discussed the possibility of using



(a)



(b) **(c)**

Figure 3.8: Automatic referencing process. a) The point cloud is divided into boxes, boxes with enough points are marked in yellow. b) SHOT feature correspondences between the model and the point cloud in the bounding box with the best match. c) The light blue robot is the model, dark blue is the final transformation using correspondence grouping of SHOT features, red is the final transform using the 4 step ICP rotation method.

the stereo camera pair for mapping the environment in Sec. 3.1.4. Based on the work presented by Popović *et al.* [60], we envisioned a pipeline where we could reconstruct the map, extract the points belonging to a robot in motion and use those points to perform continuous referencing while the robot is in motion performing collaborative tasks.

A stereo image pair would be captured and the disparity map calculated. Based on the egomotion of the HoloLens we can predict the next disparity map. The disparity map in the next frame would be compared to the predicted one. As the predicted disparity only takes into account the egomotion of the HoloLens, mismatches between the two would indicate objects whose motion cannot be explained by the motion of the HoloLens, like a moving robot. The robot model would be modified based on real-time joint data and matched to the points generated form this difference of disparity maps, providing continuous tracking and referencing of moving robots.

As stated before, unfortunately, we were unable to properly synchronize the cameras on the HoloLens (not even offline due to frame reading issues), meaning this approach could not be implemented as envisioned. It is still worth considering especially for devices lacking depth cameras. It would also remove any referencing error due to drift.

3.2.3 Tests and Results

As mentioned in the introduction, we would like our referencing method to be precise, work with different types of robots and be easy to use i.e. not require too much effort from the user.

The first batch of tests were performed for the method of refining the user input. Tests were performed using a Microsoft HoloLens and a KUKA KR-5 ARC robot as the desired reference target. We used the mesh sampling method to obtain the point cloud.

3.2.3.1 Refining user Input - Parameter Selection

First, we tested the influence of parameter selection on the outcome of the algorithm. The algorithm pipeline together with the relevant parameters can be seen in Table 3.4. One can note the large number of different parameters. For simplicity we chose two combinations of triangles per cubic meter and the number of point samples per mesh:

- Normal map: 1,000 triangles per cubic meter and 16,000 samples per mesh. 1,000 triangles per cubic meter is the highest number where the spatial mapping is stable enough to be displayed without lag. Together with 16,000 samples per mesh it only takes a short time (approximately 3 s) to sample and send the point cloud.
- Dense map: 1,240,000 triangles and 256,000 samples per mesh. In this configuration it already takes approximately 1 min to sample and send the point cloud.

For each parameter we tested five different values and ran all possible combinations to find the best parameter combination. During the testing it was found that the the segmentation algorithm required for the Super4PCS, due to it using global geometric information, failed to segment the robot from the nearby table. Less rigorous tests with a nearby KR-6 robot proved that when the robot can be segmented out of the scene Super4PCS has good performance, although very high variations based on parameter selection. The parameter

Table 3.4: The steps of the proposed referencing algorithm, the algorithms used for each step and the parameters of each used algorithm.

Step	Algorithm	Parameters
Spatial Mesh Generation	Inbuilt	triangles per cubic meter
Point Cloud Generation	Spatial Mesh Sampling	number of points
Point Cloud Filtering	MLS	search radius upsampling method upsampling radius
Registration	ICP	maximum iterations maximum corespondance distance
	SUPER4PCS	delta overlap sample size maximum time

Table 3.5: Statistics of the conducted parameter tests of ICP, Super4PCS in combination with MLS. The statistics are based on RMS of the distances between the closes points in the matched robot and the original scene point cloud, in millimeters (mm). Execution time was not tested, however ICP performed noticeably faster each time.

Algorithm	Size	Min (mm)	Max (mm)	Mean (mm)	Standard Deviation (mm)
ICP	dense	1.18	455.98	3.93	17.75
ICP	normal	1.64	14.41	3.19	1.68
Super4PCS	dense	0.40	94.72	17.40	29.01
Super4PCS	normal	0.15	67.36	10.31	7.71

tests were performed on a single captured spatial mesh to avoid variations in the mesh itself. The test results can been seen in Table 3.5. The RMS distance between the closest points in the two point clouds was used as a metric for the precision of the referencing. The best and worst parameter combinations for the ICP on the dense and normal cloud sizes can be seen in Table 3.6.

3.2.3.2 Refining User Input - Influence of User Precision

Next we conducted tests with the best identified ICP parameters and the normal point cloud size. Twelve point clouds with a user guess each were taken. Each user guess was

Table 3.6: Parameter test results of the ICP tests. All parameters listed as - did not have an effect on the result.

	Max dist.	Max iter.	MLS method	Search r	MLS param
Best dense	1,10,50,100	500	voxel grid	0.05	0.05
Worst dense	0.1	50	voxel grid	0.005	0.5
Best normal	0.1	500	no mls	-	-
Worst normal	0.1	50	voxel grid	0.005	0.1

Table 3.7: The mean, minimum, maximum and standard deviations in millimetres of the conducted tests. The first row represents the 12 original human guesses. The ICP-rotation and ICP-translation are the tests conducted by rotating and translating the original user guesses respectively

	mean(mm)	min(mm)	max(mm)	σ (mm)
User guess	27.53	3.65	138.87	44.52
ICP - all	4.91	3.22	14.42	1.39
ICP - rotation	5.90	3.22	14.21	1.90
ICP - translation	4.74	3.22	14.41	1.20

then rotated in steps of 18° to test the influence of imprecise rotation of the seed hologram. The influence of rotation on the ICP can be seen in Fig. 3.9. Keeping the original rotation, the seed algorithm was translated in a 1m volume around the initial guess with a step of 0.1m. In Table 3.7 one can see the precision of original user guesses, the precision of the ICP refinement when the seed algorithms were rotated, translated, and the average statistics of all cases. The ICP refinement performs much better and with less deviation than a purely manual method, yet it is robust to rough user guesses. Computation time was not measured in these tests.

3.2.3.3 Refining User Input - Real World Error

In the previous experiments we have seen that the precision of the referencing is in the millimeter scale (between 4 and 5 millimeters usually). However this measures the distances between the points of the model and map. How does the referencing look like in the "real world" namely the difference between where holograms are placed in the HoloLens world coordinate system and where the robot sees them in ROS, namely the robot coordinate system.

The user performs the referencing as previously described and adds another hologram cube to the scene. The cube is rotated and moved to be positioned as exactly as possible in the corner of the table in the scene. The real position in the world is known as the table, together with the robot is part of a modeled 3D object. The resulting position of the cube is received from the HoloLens and visualized in the RViz environment. We compare the values of where the cube should be if positioned correctly and where the cube position is when sent from the HoloLens.

In Table 3.8 the resulting coordinates of the cube are shown. The target values for the cube position are 1.2, -0.85 and 0.906 m for the x, y and z value respectively. The deviations from the target coordinates are also shown in the table as well as the total Euclidean distance in the last column. The task was performed by the same user. After every fourth execution of the experiment, the spatial mesh created by HoloLens was reset and a new mapping and referencing step performed.

We considered whether the batches of four tests between which the spatial mesh was not reset show a correlation. For each of the four test series, the mean distance error and the corresponding standard deviation are calculated. The mean distance error and standard derivation are depicted in Fig. 3.11.

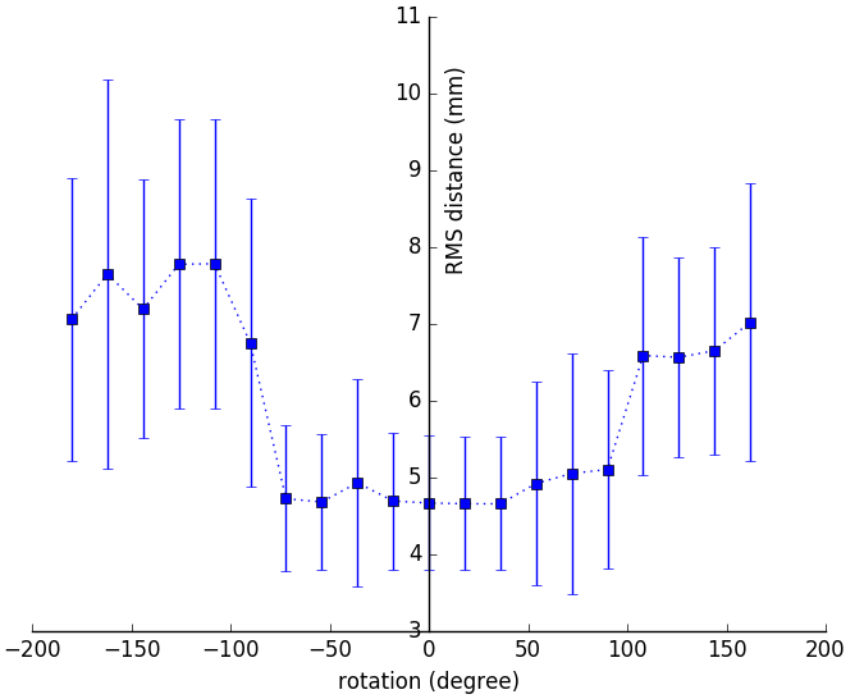


Figure 3.9: The influence of rotation on the registration algorithm. One can see that the algorithm is robust to rotational errors of the seed hologram, meaning that the user guess doesn't have to be overly precise. It is also evident that the error is not mirrored due to the robot's asymmetry.

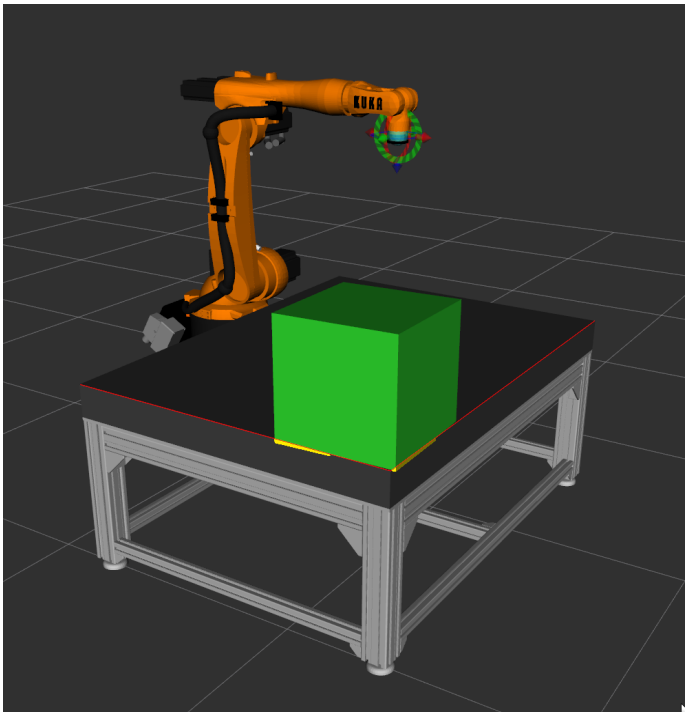


Figure 3.10: Example result of experiment in RViz. Although the user tried to position the cube to be perfectly aligned with the edges of the table - highlighted with red lines, one can see deviations -highlighted yellow areas.

	x [m]	**y** [m]	**z** [m]	**Δx** [m]	**Δy** [m]	**Δz** [m]	**Distance Error** [m]
1	1.1977	-0.8298	0.8573	-0.0023	0.0202	-0.0487	0.0528
2	1.1878	-0.8293	0.8865	-0.0122	0.0207	-0.0195	0.0309
3	1.2112	-0.8181	0.8919	0.0112	0.0319	-0.0141	0.0366
4	1.2115	-0.8212	0.9232	0.0115	0.0288	0.0172	0.0355
5	1.1906	-0.8230	0.8791	-0.0093	0.0270	-0.0269	0.0393
6	1.1785	-0.8340	0.8773	-0.0215	0.0160	-0.0287	0.0393
7	1.1996	-0.8458	0.8962	-0.0004	0.0042	-0.0098	0.0107
8	1.2158	-0.8603	0.8603	0.0158	0.0206	-0.0457	0.0526
9	1.1694	-0.8708	0.9173	-0.0306	-0.0208	0.0113	0.0387
10	1.1900	-0.8713	0.9033	-0.0100	-0.0213	-0.0027	0.0237
11	1.1785	-0.8645	0.8814	-0.0215	-0.0145	-0.0246	0.0357
12	1.2116	-0.8488	0.8877	0.0116	0.0012	-0.0183	0.0217
13	1.2224	-0.8313	0.9075	0.0224	0.0187	0.0015	0.0292
14	1.2287	-0.8225	0.8941	0.0287	0.0275	-0.0119	0.0415
15	1.2409	-0.8163	0.9314	0.0409	0.0337	0.0254	0.0588
16	1.2252	-0.8197	0.9154	0.0252	0.0303	0.0094	0.0405

Table 3.8: All the measurements of the positional deviation of the user placed cube between the HoloLens coordinate system and the robot coordinate system as visualized in RViz

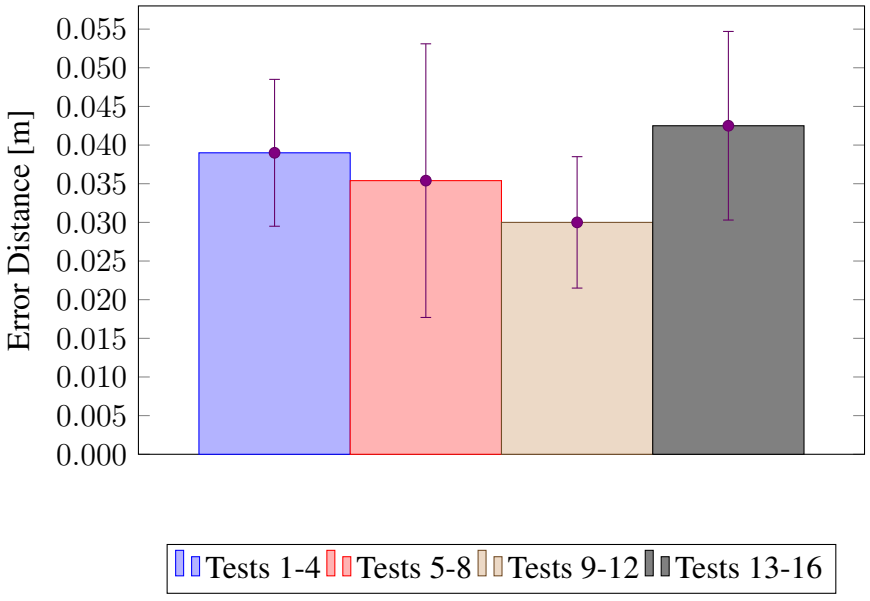


Figure 3.11: The mean distance error and the standard error deviations of the four batches of tests

There seems to be no strong correlations between the results of each subset. The standard deviations are bigger than the differences of the average displacement in each subset. The overall mean displacement for all tests is 0.0367 m with a standard deviation of 0.0122 m.

Obviously there is an user error component in the placement of the cube. To measure the user error another set of experiments was conducted. This time the user is asked to position the cube at the origin of the HoloLens world coordinate system as pictured in Fig. 3.12. The axes and location of the origin are displayed. No referencing or mapping takes place and this experiment purely shows user positioning errors.

In Table 3.9 the resulting coordinates of the cube are shown. This times only the coordinates in the Unity system are considered. The target values for the cube position are 0.2, 0.2 and 0.2 m for the x, y and z value respectively (to account for the displayed axes). The deviations from the target coordinates are also shown in the table. The last column shows the total Euclidean distance error for each test. The task has been performed by the same user. After every fourth execution of the experiment the HoloLens was restarted resulting in a different placement of the coordinate system in the real world.

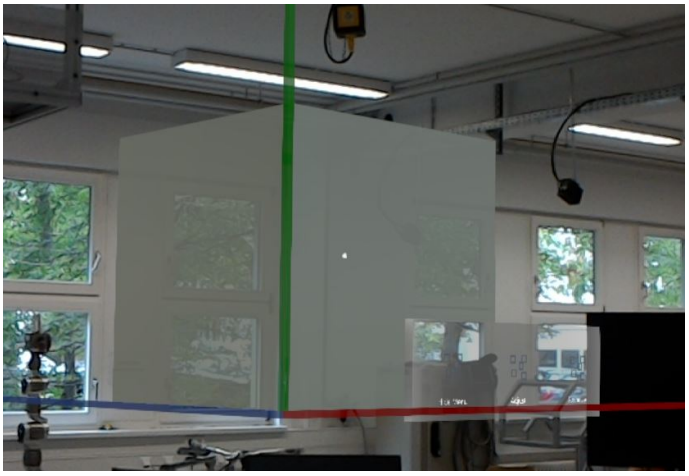


Figure 3.12: The user error test with the visualized HoloLens world coordinate system.

We again test whether the batches of tests between which the spatial mesh was not reset have a correlation. For each of the four test series, the mean error distance and the corresponding standard deviation are calculated in Fig. 3.13.

Again there is no evident correlations between the results of each subset. The standard deviations are mainly bigger than the differences of the average displacement in each subset. Therefore, all tests are considered together with the overall mean displacement for all tests being 0.0077 m with a standard deviation of 0.0037 m.

This gives us the possibility of finding the total referencing error, considering also the hologram stability error that was estimated to be 5.83 mm in [61].

$$e_{reg} = e_{exp1} - e_{holo} - e_{user} = 3.67\ cm - 0.77\ cm - 0.58\ cm = 2.32\ cm \tag{3.3}$$

The total referencing error appears to be over 2cm. An important error source that is intrinsically linked with referencing is localization. As mentioned in Sec. 2.3.1, this

	x [m]	**y** [m]	**z** [m]	**Δx** [m]	**Δy** [m]	**Δz** [m]	**Distance Error** [m]
1	0.2046	0.2026	0.1948	0.0046	0.0026	-0.0052	0.0074
2	0.2020	0.2013	0.1975	0.0020	0.0013	-0.0025	0.0035
3	0.2048	0.1909	0.2027	0.0048	-0.0091	0.0027	0.0106
4	0.1979	0.2014	0.1990	-0.0021	0.0014	-0.0010	0.0027
5	0.2016	0.1969	0.2043	0.0016	-0.0031	0.0043	0.0055
6	0.2091	0.1981	0.2029	0.0091	-0.0019	0.0029	0.0097
7	0.1841	0.2066	0.2007	-0.0159	0.0066	0.0007	0.0172
8	0.2039	0.1941	0.2025	0.0039	-0.0059	0.0025	0.0075
9	0.2003	0.1954	0.1964	0.0003	-0.0046	-0.0036	0.0058
10	0.2048	0.2010	0.2045	0.0048	0.0010	0.0045	0.0067
11	0.2050	0.2012	0.1971	0.0050	0.0012	-0.0029	0.0059
12	0.2009	0.2052	0.2000	0.0009	0.0052	0.0000	0.0053
13	0.2129	0.1994	0.2007	0.0129	-0.0006	0,0007	0.0129
14	0.2002	0.2040	0.2011	0.0002	0.0040	0.0011	0.0042
15	0.1958	0.2003	0.1910	-0.0042	0.0003	-0.0090	0.0100
12	0.1961	0.2026	0.1933	-0.0039	0.0026	-0.0067	0.0082

Table 3.9: The user hologram placment errors of the second batch of experiments.

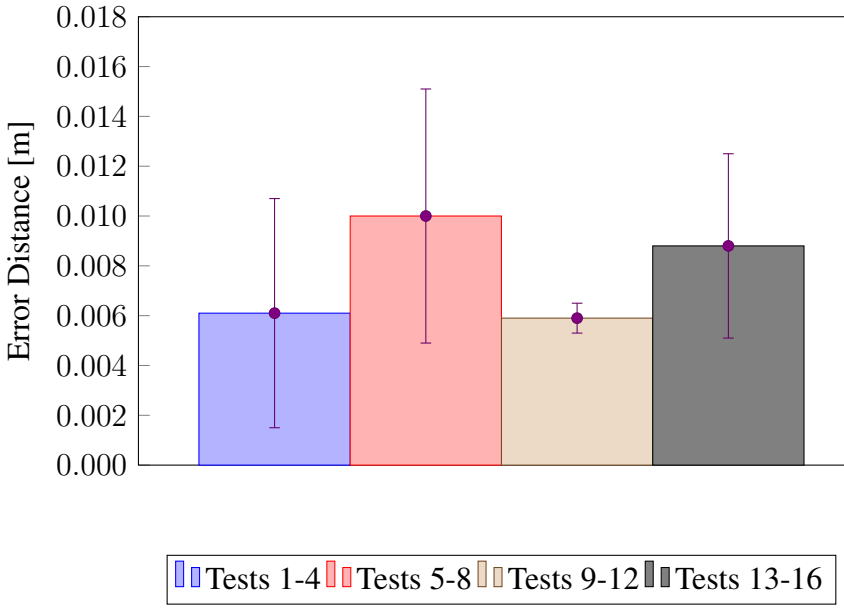


Figure 3.13: The mean distance error and the standard deviations of the four batches of tests

Scene	User Guess RMS error (mm)	Correct	Automatic RMS error (mm)	Correct
0.	2.11	✓	2.10	✓
1.	1.63	✓	1.63	✓
2.	2.21	✓	2.21	✓
3.	1.46	✓	1.46	✓
4.	1.64	✓	1.64	✓
5.	1.38	✓	1.38	✓
6.	1.44	✓	1.44	✓
7.	1.52	✓	1.52	✓
8.	5.24	X	5.22	X
9.	2.32	✓	2.32	✓
10.	1.34	✓	1.34	✓
11.	4.95	X	8.22	X

Table 3.10: RMS error in millimeters of the two referencing methods of refining the user guess and the completely automatic one on the 11 scenes taken for the KR-5 robot. The column labeled "correct" is the result of visual inspection.

and other tests, as well as other papers put the localization error at around 2cm. This appears to be main source of error in the system, as any positioning of holograms in the environment is necessarily subjected to it. It is safe to say that the actual referencing error is overshadowed by the localization error of the HoloLens.

3.2.3.4 Automatic Referencing and comparison

As we have thoroughly tested the referencing based on the user guess, we conducted tests of the automatic referencing in a similar fashion. The key question to answer is whether the automatic referencing brings additional value compared to the manual one in terms of precision (though still the largest source of error is localisation), speed, quality of user interaction, etc.

As the automatic referencing has more than 30 free parameters, a less exhaustive parameter search was performed. We chose to use a 3D Harris Detector [62] to identified points of interest for which SHOT descriptors will be calculated. The model and box point clouds are both filtered to have the same density of 1 point per $2cm^3$. We took 11 different maps featuring the KR-5 robot and 11 featuring the KR-16 robot. Despite the RMS error we inspected every match visually to prove that the referencing was indeed correct. The results are shown in Tables 3.10 and 3.11.

One can see that the two algorithms perform similarly. There were two scenes in both sets where not enough robot points were captured, and both algorithms failed on those. The main difference is that depending on the scene, the automatic referencing has a runtime around a minute, while the algorithm of refining the user guess takes a few seconds.

Furthermore less strict tests with the nearby KR-120, which is quite larger than the previous two robots, show that refining the user guess works out of the box, while parameters had to be modified for the automatic algorithm, making it less general.

Scene	User Guess RMS error (mm)	Correct	Automatic RMS error (mm)	Correct
0.	1.30	✓	1.30	✓
1.	2.87	✓	2.87	✓
2.	9.64	X	9.01	X
3.	2.63	✓	2.63	✓
4.	1.34	✓	1.35	✓
5.	1.63	✓	1.64	✓
6.	12.08	✓	14.28	
7.	1.51	✓	1.51	✓
8.	1.47	✓	1.47	✓
9.	1.48	✓	1.48	✓
10.	1.64	✓	1.65	✓
11.	4.08	X	9.11	X

Table 3.11: RMS error in millimeters of the two referencing methods of refining the user guess and the completely automatic one on the 11 scenes taken for the KR-16 robot. The column labeled "correct" is the result of visual inspection.

3.2.4 Discussion

Based on the tests it was concluded that the referencing is precise enough. Any referencing error is overshadowed by the localization error of the HoloLens which currently presents the biggest bottleneck in regards to precision.

Comparing our methods to the state of the art is rather difficult, as papers do not usually measure the precision of their referencing algorithm. In regards to methodology the best comparison would be the paper of Ostantin *et al.* [53]. The average running time of their algorithm is 23.1 seconds. with a positioning error of 9 mm and orientation error of 3.1 °. These results are comparable with our algorithm, although the orientation error wasn't measured (though it would be encoded in the RMS point distance error). The authors also noted that the algorithm may encounter errors during the clustering step, in which case the user is allowed to set the initial position of the robot, and then run the ICP algorithm only relative to the point cloud around the user-defined point. This error recovery mode is equivalent to our method of refining the user guess.

As we have shown, the user guess does not require precision and, therefore, does not present an increased mental load. Furthermore refining the user guess saves runtime while offering virtually the same precision as the automatic method. Therefore it was decided that refining the user guess shall be the preferred method going forward.

An issue that was not encountered yet may be of relevance is drift. It is expected that the localization error of the HoloLens will slowly increase over time. This would have been solved by the continuous referencing method and may be a topic of future work.

3.3 Conclusion

This chapter presented two key components of the proposed system, mapping of the environment and knowing the location of the objects of interest in a common coordinate

system through referencing. First, we defined mapping and presented the mapping capabilities of the HoloLens. Three different methods of obtaining a map of the environment in the form of a point cloud were presented. Using the raw depth sensor data was found as the best solution and the accuracy of the generated map was tested against a laser scanner, finding that 75% of points in the HoloLens map have a distance of less than 3.7 cm from laser scanner map.

Next, we presented the different coordinate systems for which the transformation has to be found. It was shown that the state of the art of using visual markers or tracking system does not fit with our vision of the system, where we require no modification to the robot cell for the AR-based interaction system. We proposed three methods to obtain the coordinate transform - refining the user guess through ICP, a fully automatic method based on sliding boxes and 3D descriptors, and a stereo camera based method. Sadly the later was impossible to implement on the HoloLens HMD. The other two methods were thoroughly tested and it was shown that the referencing has a millimeter scale precision that is overshadowed by the HoloLens localization error. The vast majority of papers do not test the referencing error. We compared our finding to those of Ostantin *et al.* and have obtain very similar results. It was decided to use the method of referencing by refining the user guess moving forward, as it does not present an increased mental load for the user while having much better runtime and precision.

The mapping part of the system pipeline is visible in Fig. 3.14. To the left are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS. In Fig. 3.15 one can see the components for the referencing, with the division again as described previously.

Two papers relevant for this section were published. In [63] we have described the referencing methods described here as well as provided some more alternative methods that could be used for referencing. In [33] we have described how to map the environment, how precise the map is and how to use the map for setting up the robot working environment, which will be described in more detail in the next chapter.

In the next chapter we will see how to use the map of the environment and the coordinate system transformation to setup the robot's working environment and program the robot itself.

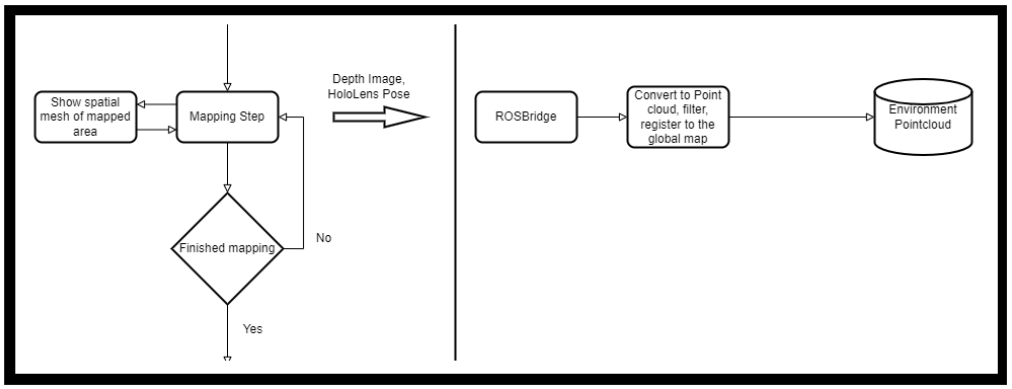


Figure 3.14: The mapping components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS.

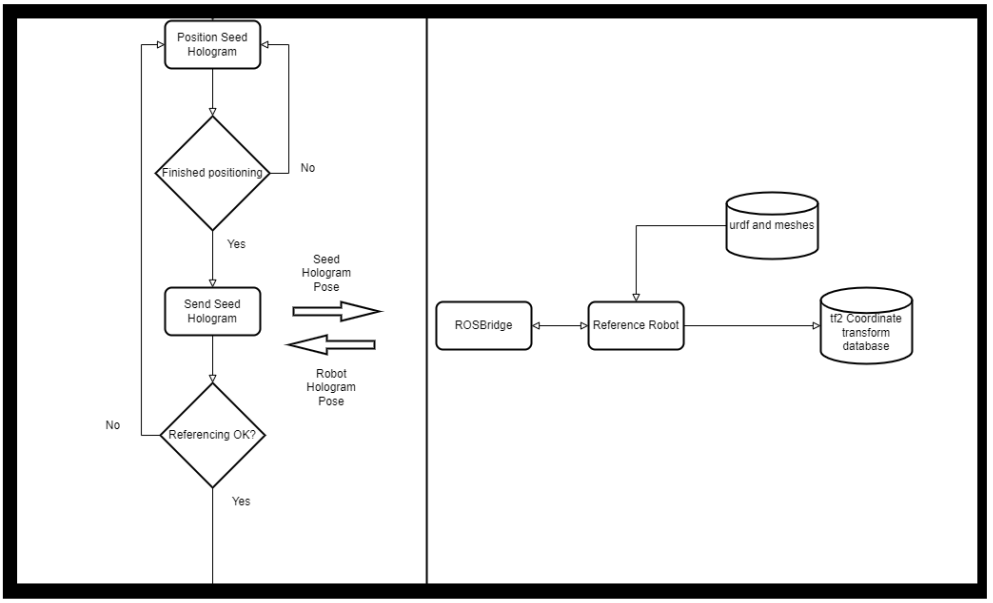


Figure 3.15: The mapping components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS.

4 Robot Workspace Setup and Programming

In the previous chapter we have shown how to use the HoloLens to obtain a 3D map of the environment and how to find the coordinate transform between the robot world coordinate system and the HoloLens world coordinate system.

In this chapter we will show how to use those information to help setup the robot working environment and to program the robot. Let's imagine a scenario where a new robot was bought. The calibration of the arm was performed according to the manual and the robot connected to a PC that will be used to program it. We would like the robot to perform a useful task while not colliding with any objects in the environment. Collisions would not only disrupt the execution of the task, but can damage the robot itself and the objects around it. Telling the robot how its environment looks like is a time consuming an grueling task, that can be much facilitated using an HMD like the HoloLens.

Next the robot needs to be programmed to execute the desired task. Once again classical methods of robot programming are time consuming and unintuitive. In this chapter we will also show the benefits of using the HoloLens to ease the programming of robots.

4.1 Robot Workspace setup

Robots need to know their working environment to avoid collisions while performing tasks. Usually robots and robot cells do not possess sensors to map their working environment, instead only possessing the sensors necessary to perform the desired task.

One can program the robot directly in the real environment, by moving the arm to desired positions and saving those positions as waypoints to avoid collisions with the objects of the environment. The robot will then continuously repeat this trajectory to perform its task. Such an approach, called on-line programming, is time consuming, inflexible and requires the robot to be out of service while the user programs the robot.

On the other hand off-line programming takes place in a virtual environment. The robot can perform its previous program while the new program is developed. Various trajectory planning methods can be used to automatically generate trajectories, making the programming simpler. The biggest downside is that off-line methods require exact representation of the environment, meaning that the virtual and real environment are basically identical. This requires precise 3D models of every object in the environment as well as determining their exact pose in relation to one another. This requires extensive measurements and the setup is again time consuming, even if the programming is made easier. If the environment changes the measurements need to be repeated and models remove or added, making such a method again quite inflexible.

Another part of the robot working environment are safety zones or no-go zones. These are regions in space that the robot can reach but we do not want the robot to move in, either because we do not possess an adequate map, or the obstacles in the region dynamically change or they are accessible by humans and the robot is not equipped for safe human-robot interaction. Safety zones are usually cuboids or spherical. In on-line programming one must input these safety zones, without being able to directly see them, and then perform exhaustive testing with the robot to see if the zones are properly setup. They are much easier to implement in off-line programming, yet, again, these rely on a precise representation of the real environment.

In this section we will describe methods of obtaining an environmental representation using the mapping and referencing algorithms described in the previous chapter. This greatly reduces the time and effort needed to produce collision-free paths for the robots.

4.1.1 State of the Art

Most industrial robot manufacturing companies offer software for off-line programming of robots, such as ABB's RobotStudio or KUKA's KUKA.Sim. These are the de facto state of the art in industry at the moment. However research exists to improve these standard software packages [64].

Neto *et al.* [65] presented a more intuitive offline programming method based on the common CAD package Autodesk Inventor. The user inputs tool coordinates and a program is automatically created. This still requires precise CAD models and calibration. They note that calibration errors are a major source of inaccuracies. According to the authors calibration requires expensive measurement hardware, software and expert knowledge. They also note that external sensing can help mitigate the errors of offline programming.

In [66] a trajectory is auto-generated for a spray-painting task by using range images of the part to colour. The collision-free trajectory generation, however, still required a model of the robot cell.

In the field of AR-based robotics, several approaches exist to plan robot motion in unknown environments. Ong *et al.* [67] use a tracked pointer tool to manually input trajectories and define collision-free volumes. However, no map of the environment is created and only a small part of the total collision-free volume is used. The authors themselves note that alternative methods for generating collision-free volumes should be explored.

Similarly, Quintero et al.[42] use holographic waypoints and B-spline interpolation to plan robot trajectories. The system relies on the user to manually modify trajectories to avoid obstacles. The mesh of the environment generated by the HoloLens is used to define waypoints on surfaces, yet the map itself is not used further.

In [68] the environment of a telepresence robot is mapped to allow the overlay of virtual fixtures - virtual objects for operator assistance. The motion of the robots, however, is guided by the user and no programming was implemented.

4.1.2 Proposed Method

We propose to use the HMD and the mapping and referencing capability to improve upon the state of the art. Our method does not require any CAD data except for the robot

model, and no additional sensors on the robot or in its working environment. This representation of the environment can then be used with off-line programming methods or further with AR-based methods proposed here or in e.g. [42]. In the latter case, when using holographic waypoints, the user would need to place much less waypoints, as high level motion planners such as MoveIt may be used to interpolate collision-free trajectories between the waypoints.

After obtaining the point cloud map of the environment and the coordinate transform described in Chapter 3, all points of the map outside the maximum reach of the robot are removed to speed-up processing. For this a kD-tree [69] representation of the point cloud is first constructed. The kD-tree is a data structure that facilitates radius and nearest-neighbour searches. We then filter all the points further away than D_{reach} from the (0,0,0) point, which is taken as the base of the robot.

We will represent the workspace of the robot with a voxel grid. Basically, the voxel grid discretizes space into cubes, similar to how pixels are used to discretize 2D space and images. In our map each voxel may be either free, meaning the voxel is not part of an obstacle i.e. object, or occupied, meaning it is part of an object with which the robot may collide. One could also assign probabilities of voxels being occupied but in our case, where we want to avoid collisions at all costs, we apply a binary occupied/non-occupied approach. The voxel i.e. cube has an edge size of D_{leaf}. In ROS one may use the OctoMap package to deal with voxelization of the environment. The OctoMap package uses a voxel-based occupancy grid - each voxel has a probability of being free space or an obstacle. Furthermore OctoMap incorporates an octree-based map compression. An octree is a hierarchical data structure for spatial subdivision in 3D. Each node in an octree represents the space contained in a cubic volume - a voxel. This volume is recursively subdivided into eight sub-volumes until a given minimum voxel size is reached. Further information can be found in the paper by Hornung *et al.* [70].

The map point cloud is down-sampled with a voxel-grid filter with voxel size D_{leaf}, which is the size of the voxels in the OctoMap. For each point in the model point cloud of the robot, a nearest-neighbour k-d tree search is performed to remove all the robot points from the map. An outlier removal filter is used to eliminate any robot points that may have been left over by the previous step. Finally the resulting point cloud is used to generate an OctoMap voxel grid representation of the occupancy as seen in Fig. 4.1.

The generated voxel occupancy grid is sent back to the HoloLens where a user may visualize and edit the occupancy grid. This step allows the user to correct any errors made by the algorithm if they exist. It also allows the user to remove occupied voxels from parts of the environment where contact with the environment is needed to perform the robot's task. Finally, the user may also add safety zones in situ, drastically reducing the setup and testing times. In Fig. 4.2 the overlaid robot model, the rendered OctoMap, and the set up of the safety zones can be seen.

When the user is done, the safety zones and the edited OctoMap are sent back to the computer. As the user can freely move and add voxels through AR on the HoloLens, these voxels must be snapped back to the voxel grid. These OctoMap environmental representations can be saved, loaded in MoveIt and edited with the HoloLens as many times as necessary. Likewise, one could save different voxel grids and safety zones depending on the task for future uses. A representation of an edited map and safety zones in RViz can be seen in Fig. 4.3.

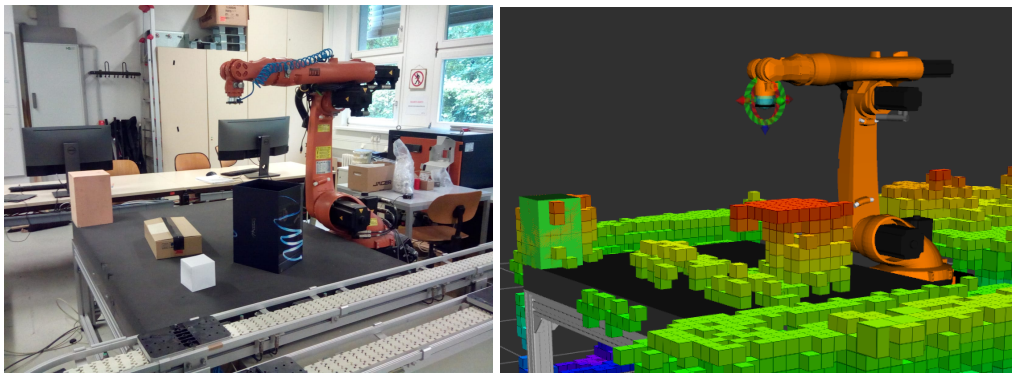
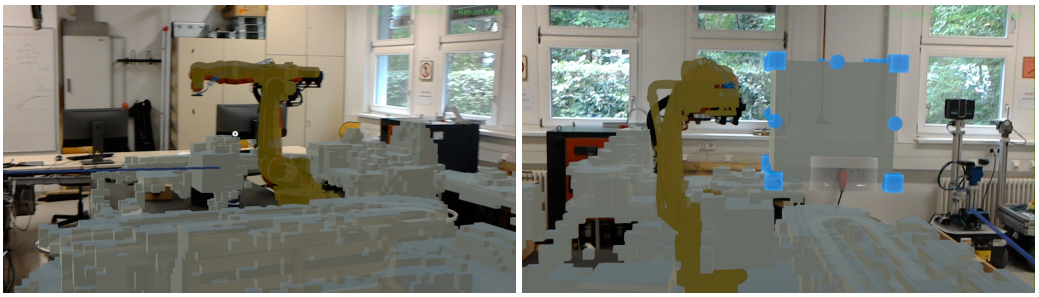


Figure 4.1: The real scene which was mapped with the HoloLens (left) and the resulting OctoMap after the point cloud map was converted into a voxel grid, together with the 3D model of the robot and table, as seen in RViz.



(a) **(b)**

Figure 4.2: (a) Visualisation of the voxel grid representation of the environment in the HoloLens. The individual voxels can be added, moved and removed; (b) Adding safety zones in situ using the HoloLens. Such definition of safety zones are more intuitive and faster than classical input in offline robot programming.

4.1.3 Tests and Results

To evaluate the quality of the OctoMap generated from the HoloLens' point cloud we placed a wooden 20x20x30 cm cuboid on the front-left of the table. We assumed the worst case scenario of using the sampled environmental mesh of the HoloLens. Khoshelham *et al.* [71] found that the average global error of the HoloLens' environmental mesh is around 5 cm, which is much higher that the map we were able to generate and test in Chapter 3 that has an average error of 2.15 cm. We therefore also used a D_{leaf} size of 5cm. This gives a minimum of 96 and a maximum of 160 voxels belonging to the target cuboid, depending on how well the grid edges align with the cuboid edges. We counted the total number of false-positives, i.e. the voxels that are detected as occupied by the object that are in fact not, and false-negatives, i.e. voxels detected as free that are in fact part of the object. The results presented in Table 4.1 show that on average there are 9.58 false-negatives (6-10% total voxels) and 61.33 false-positives (39-65% voxels) with 12 point clouds tested. Worth noting is that the false-negatives are much more critical as they can cause crashes while false-positives only slightly limit the collision-free volume. Also worth noting is that some false-negatives are hidden behind false-positive voxels or near the table and are therefore unreachable.

We also carried more than 50 tests where we gave a desired goal pose to the robot and let MoveIt plan a collision free trajectory in the cluttered scene shown in Fig. 4.1. If a collision does occur it will be due to the errors in the voxel representation of the environment. Although the vast majority of the tests had no collision, there were indeed a few edge cases were a collision happened when objects are positioned diagonally to the table. This is due to small location and mapping errors combined with the discretization direction of the workspace.

Table 4.1: The number of false-positives and false-negatives in the 12 OctoMaps tested. The object contains between 96-160 voxels.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	$\varnothing$	σ
false positive	83	60	67	71	52	75	78	61	41	35	58	55	61.33	14.48
false negative	6	16	10	5	5	8	14	7	12	15	11	6	9.58	3.99

To eliminate these rare edge cases a simple method of padding the voxel grid was implemented. for each outside face of an edge voxel, meaning occupied voxels bordering non-occupied voxels, voxels of half the leaf size, D_{leaf}, were added. Multiple padding levels can be added, each being composed of voxels half as small as the previous one. One can however note that already with 2 levels the collision objects are enlarged by 75%. An illustration on padding levels can be seen in Fig. 4.4.

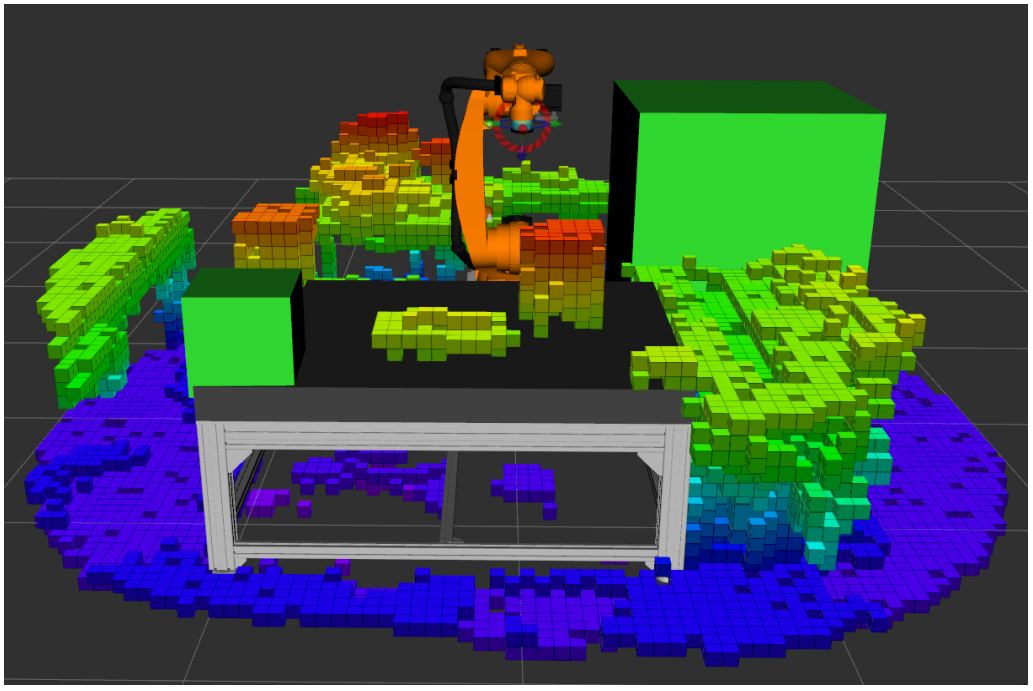
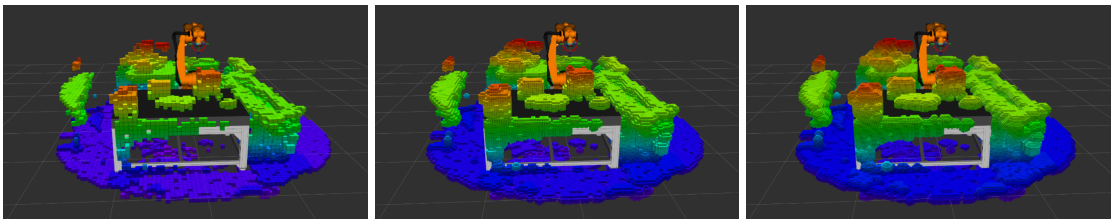


Figure 4.3: The edited voxel grid and added safety zones as visualised in RViz. To note is that the table in this application was part of the robot description file. If a CAD model exists and the robot should interact with that part of the environment, adding it to the robot model will filter out the unwanted voxels automatically.



(a) (b) (c)

Figure 4.4: The padding process to remove the edge cases that might result in collisions. (a) The original OctoMap with 5cm voxels; (b) First level padding with 2.5 cm voxels; (c) Second level padding with 1.25 cm voxels on top of the level 1 padding.

4.2 Robot Programming

We have already dove somewhat into robot programming in the previous subsection, as programming and workspace setup are intrinsically connected. In on-line programming the user moves the robot directly, usually through a control pad, a part of the robot control box, and saves the desired trajectory points and tool behaviour to be later replayed by the robot. Such a method is time consuming, requires expert knowledge and needs to be repeated whenever the task or the environment changes. The robot is out of use for the duration of the programming. Off-line programming needs a precise model of the environment but may leverage motion planning and other software to facilitate programming. The robot may also continue working while the new program is developed. The programmer however is also decoupled from the real physical surrounding of the robot.

In the past decade a new class of robot emerged, the so called CoBots [3]. These robots possess Joint-Torque Sensor (JTS) and are lightweight enough to work safely alongside humans. The sensors detect any collision and stop the robot. Alternatively the JTS may be used to perform impedance/admittance control that allow the robot to control it's position and exerted force based on the force input from the environment. Such controls allows these robots to be programmed by hand guidance, also called walk-through programming. This is likewise an on-line programming method, however, the user may simple grab the arm itself and move it to perform tasks. The tools on the robots usually also possess buttons to activate them, meaning programming becomes much more intuitive.

Other methods also exist. One can for example use programming blocks which encode robot behaviour from simple movement to sensor-driven behaviours. Using learning by demonstration, one can also teach the robot how to perform a task instead of directly programming it. Such methods rely on machine learning principles. More advanced machine learning methods may allow the user to even use speech to tell the robot what to do.

Robot programming trough the use of AR and HMDs is one of the most researched topics combining AR and robotics. This is due to modern HMDs' ability to provide numerous interaction and visualization modalities to help the user program robots.

4.2.1 State of the Art

Since a comprehensive review of all the proposed methods for programming robots would be a massive undertaking, as programming the robot is such a core feature, we will endeavour to represent the general landscape of the research. This should hopefully highlight the current issues and present alternatives to the methods proposed here. We can direct the reader to survey papers, such as [72], [73] and [74]. As classical online and offline programming methods have already been described, we shall focus on newer and different programming methods. Some of these concept have been implemented in our work (e.g. hand guidance) while others serve as a brief overview of alternative approaches.

4.2.1.1 Hand Guidance

As mentioned in the opening of the chapter, CoBots possess JTSs that allows them to be programmed by hand guidance - merely grabbing the robot and showing them physically how to perform the task. This intuitive method allows easy teaching of new trajectories, even by lay-users. Such ease of teaching and reprogramming is of extreme importance to expand robotics to small and medium enterprises, as well as in flexible manufacturing paradigms [4]. Hand guidance on industrial robots in the mean time has been marginal, usually requiring external force-torque or other sensors in the robot workspace. Traditional teaching pedants are still the most used tool to program industrial robots, despite requiring extensive training to use and being slow when it comes to reprogramming. Besides industrial settings, hand guiding is also important in medical robotics [75]. Beyond simply easing trajectory teaching, hand guidance is an essential part of kinesthetic learning, a sub-field of learning from demonstration [76]. We will cover imitation learning in Sec. 4.2.1.3.

As we would like our approach to work on portable systems and both with CoBots and standard industrial arms, it is important to see if there have been other approaches to achieve hand guidance that do not require additional sensors and that work with industrial robots.

Moe *et al.* use a Microsoft Kinect and a smartphone-based accelerometer to perform hand tracking to guide the end-effector of an industrial robot [77]. This approach, however, was limited to 5-Degrees of Freedom (DOF). Furthermore by just driving the end-effector one cannot make use of extra degrees of freedom that redundant joints may provide. It still provides a good proof of concept, as the system would be entirely portable were it not for the Kinect.

Lee *et al.* proposed a generalised method of hand guidance by torque control, based on the dynamic model of a robot, the motor current, and a joint friction model [78]. The approach requires experiments to determine the friction model of each joint, as well as possessing a dynamic model. Furthermore the robot is confined to using a torque controller. Finally the external force needed to move the end-effector was found to be 1.23-4.83 times greater than approaches based on JTS. Ideally the force would be close to zero, which is not the case even for JTS methods.

A similar approach was taken by in [79], requiring a dynamic model and friction model identification, using a voting system and admittance control to move the robot. These two factors, as with the previous approach, prevent it to be a truly "plug-and-play" method.

In [80] a method without friction model identification was used with only a feed-forward term for partial friction compensation. The force required to move larger robots unfortunately became too large for human operators, due to uncompensated friction.

Teaching robot trajectories in virtual reality is also possible, however that requires either exact modelling of the workspace, or additional sensors to track its status in real time, which we would like to avoid.

4.2.1.2 Graphical Programming

Graphical programming refers to the use of graphical elements to represent higher level actions and behaviours of a program. General example include MatLab's Simulink and Microsoft Visual Programming Language. In robotics it has been shown that graphical programming tools ease robot programming both for lay-users [81] and experts [82]. In the latter case it eases interaction between the large amount of software components of complex systems and prevents writing repetitive and therefore error-prone source code.

The specific blocks can be written by experts to be used by lay-users, or they can be generated automatically, for example by using ontological reasoning [83] or by emulating user input in different programming environment [84] creating a general tool that can be exported and implemented on different proprietary software of robot manufacturers.

4.2.1.3 Learning from Demonstrations

The goal of Learning from Demonstration (LfD) is to teach robot skills from human demonstrations. The action may be performed directly by the human and captured via cameras, depth sensors, motion trackers etc. [85] or the human may take direct control of the robot and demonstrate the robot how to perform the action [86]. In the later case we differentiate between kinesthetic teaching and teleoperation [87]. The skills learn need to be generalisable, robust to changes in the environment or robot states. Here notably the difference between kinesthetic teaching and hand guidance is that through hand guidance the robot just repeats the shown motion.

Recently the use of VR has also been demonstrate as a possible data source for LfD. This simplifies the data acquisition process and may be used to implement any of the three methods.

4.2.1.4 Reinforcement Learning

Whereas in learning from demonstration a user demonstrates the correct way to perform a task, in reinforcement learning no examples are given. It is usually connected with deep learning i.e. the use of artificial neural networks with extremely large numbers of parameters that are trained to perform a task. The desired outcome of a task is encoded in the so called reward function [88]. Large number of robots and training time are required for good performance [89]. Another option is the so called sim-to-real, where the training takes place in simulation and is then transferred onto real robots [90]. Reinforcement learning is now days tightly coupled with transfer learning, the ability to reuse knowledge from previous tasks to perform novel tasks with less training [91].

The most recent advances use embodied language models to directly incorporate real-world sensor data into language models to establish the link between words and percepts. For example PALM-E [92] allows the human user to use natural language to command a robot to perform a task, which the robot performs even under disturbances. Such large scale language models also enable general inference.

Reinforcement learning and deep learning require large scale training data, computing power and very often suffer from lack of explainability i.e. if an error occurs it is often quite difficult to properly explain why. Such approaches promise to one day enable autonomous robots, which can be asked to perform tasks and conversed with in a human manner without the need for any sort of programming on the part of the user.

4.2.1.5 AR-based Approaches

AR assisted robot programming is perhaps the most researched sub-field of AR-based HRI. The earliest research used camera and screen [8] [6]. However as technology moved forward more advanced systems that allowed the robot and user to be in the same environment were developed.

One particular research direction is the use of varied intuitive input devices to facilitate robot programming. These devices are tracked by external tracking systems, most notably using infra-red (IR) markers. *Hein et al.* [93] have shown that such devices allow even untrained people to solve robot jogging, trajectory input and surface interaction error-free. Such devices can also feature additional sensors, such as a force sensor in the tip, to help with input. This approach has often been combined with projectors to provide AR visualization [24] [94] [67]. This was a popular approach before the advent of commercial HMDs, however it still remains a popular choice, even being combined with AR of XR HMDs [95]. The main advantage of such a system is the tracking precision offered by the motion tracking system. This also allows tracking of various tools to ease programming. Reinhart *et al.* [24] also list trajectory editing, interaction with virtual menus and the digitising of object surfaces - generating 3D models of unknown objects on the fly, as interaction options of such systems. The biggest disadvantage is that such systems are static and require extensive setup (less so when HMDs are used instead of projectors).

Another approach that was quickly developed after the introduction of the HoloLens HMD is the use of holographic waypoints [42]. The user can use either the gaze or manipulate the waypoints via hand tracking. Once the user is finished, an interpolation step based on B-splines constructs the trajectory of the robot.

For pick and place tasks, one may also select the object or location for picking and then select the location to place the object, such as in [46]. A higher level motion planner is then used to calculate the trajectory (it is not specified which planner was used, most likely MoveIt).

A general benefit of using HMDs in robot programming is the option of in situ virtual execution [46] [42]. The user can view the entire execution of the program in the robot workspace but with a virtual holographic robot instead of the physical one. This allows inspection and error correction without the risk of collisions and damage to the robot itself.

AR approaches are also often combined with the other robot programming methods described earlier. They are made easier to use and more intuitive through the use of AR

and HMDs. Mateo *et al.* combine AR and graphical programming [44]. Liu *et al.* use the HoloLens to patch missing knowledge gained through imitation learning [48]. Luebbers *et al.* use the HoloLens to ease the input of constrains and visualize learned policies during constrained learning from demonstration [96].

4.2.2 Proposed Methods

We developed three different methods to program the robot via HMDs. The first one uses holographic waypoints similar to [42]. The second seeks to emulate hand guidance by using the HoloLens hand tracking capabilities. The third one adds the option to use tools, such as a smart pen, to ease the robot programming. Though classically such tools were tracked via IR marker tracking systems [94] [67], we achieved inside-out tracking using the HoloLens' IR emitters, which are part of the depth sensor, to illuminate the IR markers on the tool.

4.2.2.1 Using Waypoints

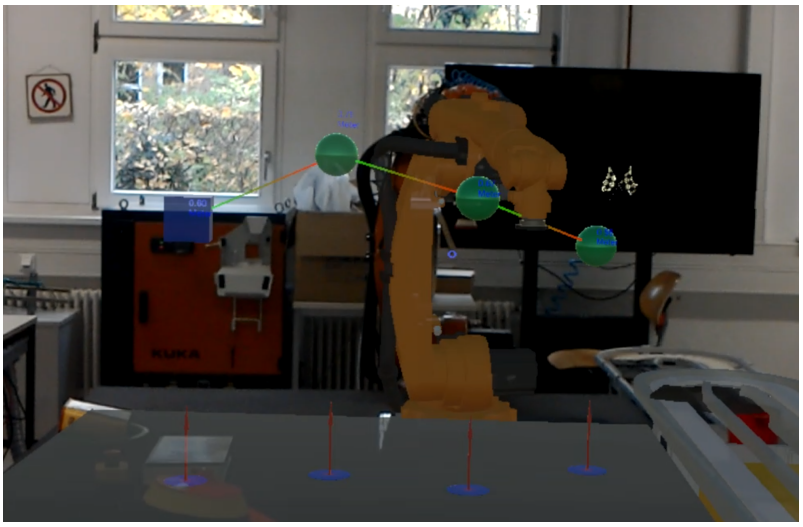


Figure 4.5: First person view from the HoloLens. The waypoints, their position in the robot base coordinate system, and their projection on the table (which was part of the robot model in this case) can all be seen. A virtual robot is overlaid over the real one and used to test the validity of the planed trajectory.

With the advent of the HoloLens, using HMDs to define holographic waypoints was a natural low-hanging fruit. We started working on such an approach before Quintero *et al.* [42] published their paper in 2018. Our approach took quite a similar form. The user can use voice commands to interact with the system, adding and removing waypoints and requesting the trajectory planner to plan a path through the waypoints. The main difference is that we use MoveIt high level path planner to plan our trajectories instead of relying on interpolation as in [42]. This combined with the cell setup allows the user to input much fewer waypoints to achieve a collision-free trajectory. We also visualize the location of the waypoints in the robot base coordinate system as well as the option to project the position of free-space waypoints to the surfaces below. This is very useful e.g. to position the robot above the object to grasp in pick-and-place tasks. After [42] was

published, we also implemented surface waypoints i.e. waypoints that are bounded to the spatial mesh. This allows programming the robot with tasks that require surface contact like milling or welding. In Fig. 4.5 the first person view from the HoloLens is visible. One can see the waypoints and their projection as well as the virtual robot overlaid on top of the real one. When the planning is done, the virtual robot executes the trajectory and the user can check if the generated trajectory is satisfying.

4.2.2.2 Hand Guidance

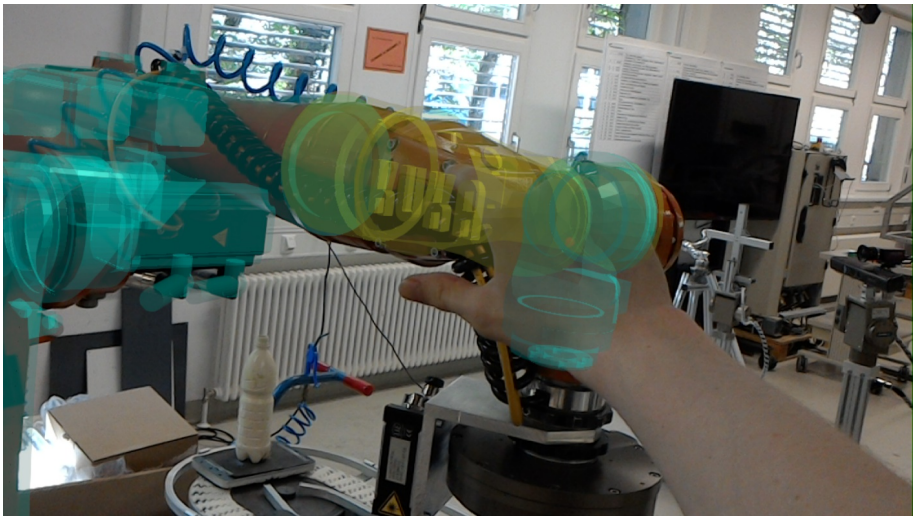


Figure 4.6: First person view of the hand guidance modality from the HoloLens

We wanted to leverage the hand tracking capabilities of the HoloLens to enable general hand guidance independent on the type of the robot and not requiring sensors on the robot or around it. The proposed approach requires only the urdf of the robot which describes its kinematic chain. Unlike the other methods mentioned in the state of the art, our method doesn't require any special setup or parameter identification. The user does not need any force to move robot, it can be used with different robot motion controllers and the sensitivity of the movement can be adapted on-the-fly.

After the robot was successfully referenced, resized convex meshes around the virtual robot links define the area in which hand movements translate to robot movements. An example of such a convex mesh is visible in Fig. 4.7. The size of the area can be freely modified online through a holographic menu. We use the *Hold* gesture, defined in the HoloLens' specifications, to virtually push and pull the robot links. When the user's arm is in the action area of a specific link, the link in question turns yellow on the overlaid robot model. When the user actively moves the link, the link turns orange.

Also worth noting is that the virtual robot can be moved or resized. This allows hand guidance from a safe distance also with large robots that would usually be too big to perform this type of hand guidance.

When the user makes the *Hold* gesture inside the mesh of link *j* whose parent joint at position $s_{j,t}$ has a rotation axis $a_{j,t}$, the positions of the hand in the current frame h_t and the position of the hand in the previous frame h_{t-1} are projected unto the plane defined by $s_{j,t}$ and $a_{j,t}$ as per (4.1):

$$p_{j,t} = h_t - \frac{h_t \cdot a_{j,t}}{\|a_{j,t}\|^2} a_{j,t}, \quad p_{j,t-1} = h_{t-1} - \frac{h_{t-1} \cdot a_{j,t}}{\|a_{j,t}\|^2} a_{j,t} \tag{4.1}$$

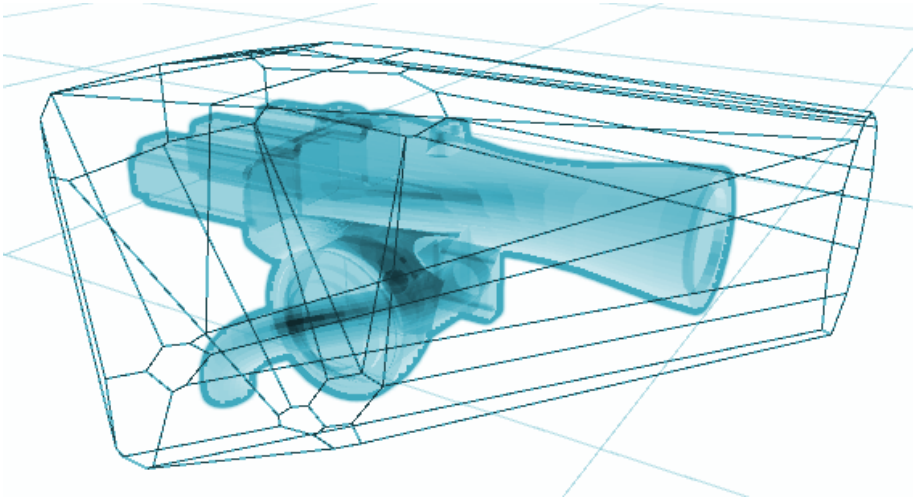


Figure 4.7: The convex mesh around the link defines the area in which hand movements are translated into robot movements.

The normalised vectors of the projection in relation to $s_{j,t}$ are:

$$v_{j,t} = \frac{p_{j,t} - s_{j,t}}{\|p_{j,t} - s_{j,t}\|} \quad , \quad v_{j,t-1} = \frac{p_{j,t-1} - s_{j,t}}{\|p_{j,t-1} - s_{j,t}\|} \tag{4.2}$$

The angle between the two vectors, and therefore the change in joint angle is

$$\Delta\theta_{j,t} = \arccos\left(v_{j,t-1} \cdot v_{j,t}\right) \cdot \text{sign}(a_{j,t} \cdot (v_{j,t-1} \times v_{j,t})) \tag{4.3}$$

The new desired joint angle is therefore

$$\theta_{j,t} = \begin{cases} \theta_{j,t-1} + \mathrm{K}\Delta\theta_{j,t} & \theta_{j,min} \leq \theta_{j,t} \leq \theta_{j,max} \\ \theta_{j,t-1} & \text{otherwise} \end{cases} \tag{4.4}$$

Where K is a tunable motion scaling factor.

This angle update however covers only part of the hand motion, to get the full motion we need to propagate the remaining motion down through the kinematic chain. We rotate h_{t-1} around $a_{j,t}$ centred in $s_{j,t}$ by the angle $(\theta_{j,t} - \theta_{j,t-1})$

$$r_{j,t-1} = q_{j,t} \cdot (h_{t-1} - s_{j,t}) \cdot q_{j,t}^{-1} + s_{j,t} \tag{4.5}$$

Where $q_{j,t}$ is the quaternion representing the rotation around $a_{j,t}$, $q_{j,t}^{-1}$ is it's conjugate, and the $\cdot$ represents the Hamiltonian product. The quaternion $q_{j,t}$ is defined as

$$\begin{gathered} \hat{q_{j,t}} = (\cos\left(\frac{\theta_{j,t} - \theta_{j,t-1}}{2}\right), \sin\left(\frac{\theta_{j,t} - \theta_{j,t-1}}{2}\right) a_{j,t}), \\ q_{j,t} = \frac{\hat{q_{j,t}}}{\|\hat{q_{j,t}}\|} \end{gathered} \tag{4.6}$$

The same computation is then reiterated for each joint down the kinematic chain until $r_{j,t-1} = h_t$ or there are no more joints left. In the latter case the complete motion of the hand is not reproduced by the robot due to kinematic constrains. In Fig. 4.8 one can see a graphical representation of the equations.

The vector of joint position updates is then sent to ROS where any controller can be selected through the ros_control package. If the controller in question is Cartesian, the desired joint values can be converted to the desired end-effector pose via forward kinematics.

Given that in the majority of cases dragging the end-effector may be preferable, a holographic sphere can be placed around the end-effector which can be dragged and rotated around the three axis of the world coordinate system. The pose of the sphere then indicates the desired pose of the end effector. For control in joint-space the inverse kinematic solution is used that minimizes the total change of all the joint angles. In our case we use the MoveIt framework to perform both forward and inverse kinematics as needed.

Such an interaction system would become even more intuitive provided full hand skeleton tracking was available. Newer devices such as the HoloLens 2 already offer such a modality. Alternatively the RGB camera of the HoloLens 1 can be used in conjunction with e.g. [97] to provide robust skeleton hand tracking even with surface contact and occlusion.

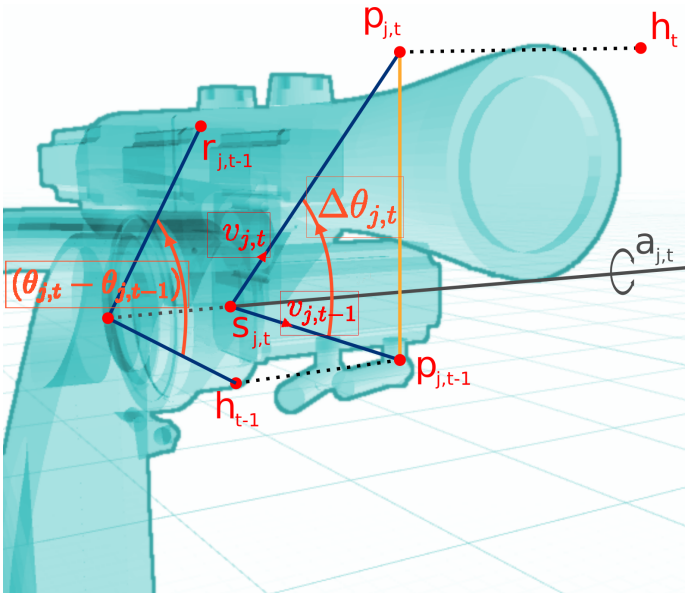


Figure 4.8: Graphical representation of (4.1) - (4.6). The hand motion between the two consecutive frames has been vastly exaggerated for the sake of clarity

4.2.2.3 Tracking Tools

It is sometimes beneficial to use tools to help input robot motions, especially when surface contact or the amount of force applied need to be specified. Researchers at the Intelligent Process Automation and Robotics Lab developed such tools and showed that they indeed ease robot programming [93]. Smart input devices such as the smart pen can be used to easily define a machining process for industrial robots [98]. As already mentioned the downside is that the robot workspace must be equipped with a marker tracking system,

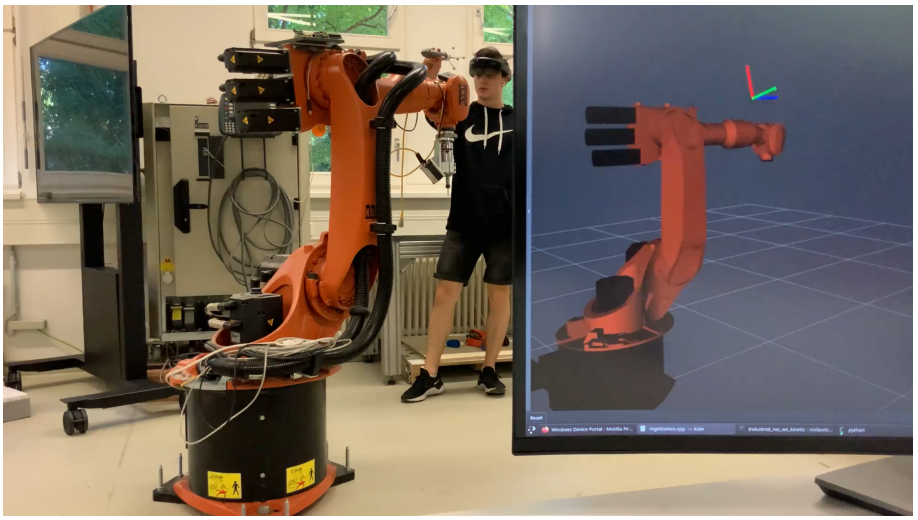


Figure 4.9: The user wearing the HoloLens and holding the tracked input device equipped with IR-markers. The input device is tracked purely through the on board sensors of the HoloLens. One can also see the live tracking in RViz on the screen of a desktop PC running ROS.

which makes the approach immobile and requires extensive setup and calibration. In our case the marker tracking system used IR markers.

To alleviate these issues, we proposed to use the same tools, however the tracking would be performed by the HoloLens itself. This is enabled by the depth sensor of the HoloLens, which uses IR projectors to illuminate the environment, with the depth camera measuring the time it takes for specific points to be reflected back. This depth measuring principle is called ToF. The projectors however can also be used to illuminate the IR markers of the smart tools. Specifically, we use the short-throw reflectivity stream to track IR markers illuminated by the HoloLens' depth sensor.

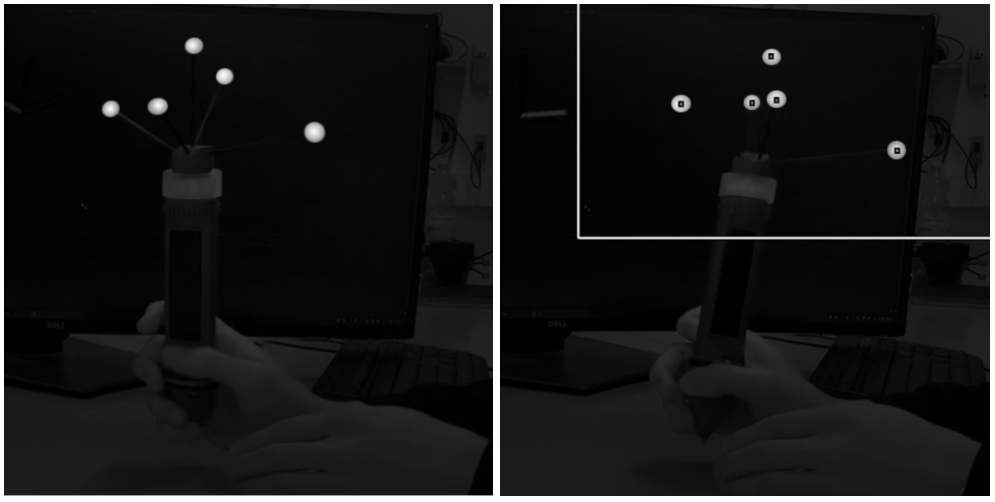
After undistorting the frame we apply a threshold filter, as the markers are highly reflective. The threshold filter is applied so that only pixel values higher than 250 remain, since the marker pixel values are very close to 255. This will eliminate interference from other reflective surfaces. The image size is reduced by half for blob detection to save on processing speed. The centre pixels (x_i,y_i) of each blob are then mapped into 3D space using the pinhole camera model:

$$m_i = \left(\frac{Z_i \cdot x_i}{f}, \frac{Z_i \cdot y_i}{f}, Z_i \right)^T \tag{4.7}$$

Where f is the focal length of the HoloLens depth camera, and Z_i is the distance of the i-th blob centre from the camera, obtained by mapping the centre pixel of the blob to the depth stream of the same frame.

The distance matrix (D_R) of the set of detected markers M_R is created and the distances compared to the distance matrix of the model (D_M). Each distance matrix element d^{ij} holds the distance between the points m^i and m^j. As there will inevitably be a reconstruction error, a tolerance threshold δ is introduced. Two distances are considered equal if:

$$| d_R^{pq} - d_M^{ij} | < \delta \tag{4.8}$$



(a) **(b)**

Figure 4.10: (a) The original, distorted reflectivity stream. One can notice the IR markers illuminated by the HoloLens' depth senor are quite visible. (b) The detected position of the blobs and the defined region of interest for the next frame.

Where d_R^{pq} is the distance between two reconstructed markers, m_R^p and m_R^q and d_M^{ij} is the distance between two model points, m_M^i and m_M^j. A simple diagram with three points is visible in Fig. 4.11.

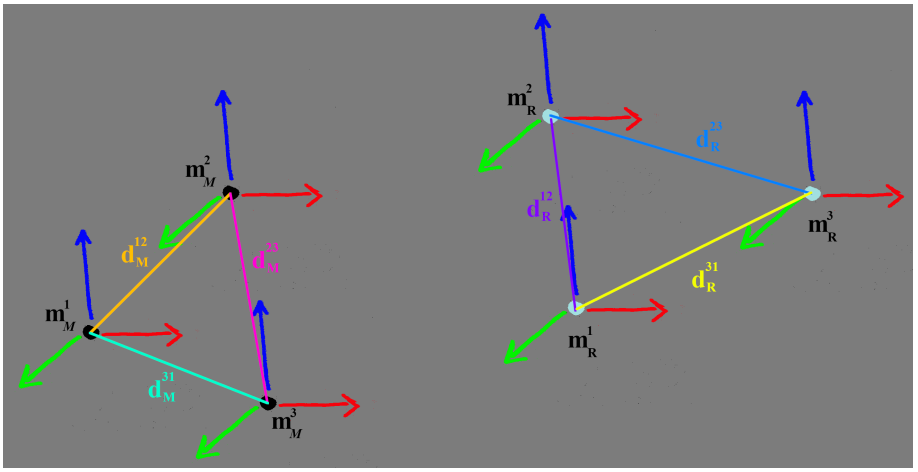


Figure 4.11: Left - a visual representation of model parameters d_M^{ij}, m_M^i and m_M^j; Right - representation of reconstructed parameters d_R^{pq}, m_R^p and m_R^q.

After each of the found markers has been assigned to a correspondent in the model, the rotation and translation between the two rigid bodies has to be determined.

The main approaches are through the use of orthonormal rotational matrices [99] and the Singular Value Decomposition (SVD) of the covariance matrix [100]. *Umeyama* ([101], *Kanatani* [102] and *Challis* [103], improved the previous methods, especially by fixing a flaw where noisy data sometimes caused the rotation matrix to have a determinant of -1 and thus become a reflection rather than a rotation. Our algorithm is based on the work

from *Kanatani* [102] and *Challis* [103] who both independently expanded on the method proposed by *Arun et al.* in [100].

Given a point x_i in the set M_R and point y_i in the set M_M ($i = 1, 2, ..., N$ and $N \geq 3$). The transformation between them is:

$$y_i = Rx_i + t \tag{4.9}$$

Where R is a 3x3 rotation matrix and t is the translation vector.

The least-squares problem of finding R and t is equivalent to minimising the following expression:

$$\frac{1}{n}\sum_{i=1}^{n}(Rx_i + t - y_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(Rx_i + t - y_i)^T(Rx_i + t - y_i) \tag{4.10}$$

This can be further simplified by eliminating t as an unknown variable. We compute the mean vectors of both point sets $\overline{x}$ and $\overline{y}$. The vector t can then be computed as:

$$t = \overline{y} - R\overline{x} \tag{4.11}$$

Substituting t in Equation 4.10, we get:

$$\frac{1}{n}\sum_{i=1}^{n}(Rx_i - y_i + \overline{y} - R\overline{x})^T(Rx_i - y_i + \overline{y} - R\overline{x}) \tag{4.12}$$

We define two new point sets by translating the sets y_i and x_i so that the means $\overline{x}'$ and $\overline{y}'$ of these two new point sets are located in the origins of the two reference frames:

$$x'_i = x_i - \overline{x}, \quad y'_i = y_i - \overline{y} \tag{4.13}$$

Substituting these two new vectors in Equation 4.12, we get:

$$\frac{1}{n}\sum_{i=1}^{n}(y'_i - Rx'_i)^T(y'_i - Rx'_i) \tag{4.14}$$

By expanding and reducing the equation above, we get:

$$\frac{1}{n}\sum_{i=1}^{n}({y'_i}^T y'_i + {x'_i}^T x'_i - 2{y'_i}^T Rx'_i) \tag{4.15}$$

By using the following equivalences:

$$\{Rx'_i\}^T y'_i = {y'_i}^T Rx'_i \tag{4.16}$$

$$\{Rx'_i\}^T Rx'_i = {x'_i}^T R^T Rx'_i = {x'_i}^T x'_i \tag{4.17}$$

Thus, minimising Equation 4.10 is equivalent to maximising:

$$\frac{1}{n}\sum_{i=1}^{n}({y_i'}^T R x_i') \tag{4.18}$$

Rearranging and summing this, gives the following to maximise:

$$\frac{1}{n}\sum_{i=1}^{n}({y_i'}^T R x_i') = tr\left(R^T \frac{1}{n}\sum_{i=1}^{n} y_i' {x_i'}^T\right) = tr(R^T C) \tag{4.19}$$

Where $tr()$ is the trace of a matrix and C is the correlation matrix computed as:

$$C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})(x_i - \overline{x})^T = \frac{1}{n}\sum_{i=1}^{n} y_i' {x_i'}^T \tag{4.20}$$

Using SVD one can decompose the correlation matrix $C = UWV^T$ where U and V are orthogonal matrices and W is a diagonal matrix containing the singular values of C. By substituting the SVD of C into Equation 4.19, we get:

$$tr(R^T C) = tr\{R^T U W V^T\} = tr\{V^T R^T U W\} \tag{4.21}$$

We now define a new matrix Q as:

$$Q = V^T R^T U \tag{4.22}$$

Thus we now have to maximise:

$$tr(R^T C) = tr(QW) \tag{4.23}$$

Since W is a diagonal matrix, the result of Equation 4.23 is only influenced by the values along the main diagonal of Q. Thus maximising Equation 4.23 becomes the problem of maximising the values on the main diagonal of Q.

As V, R and U are orthogonal the same must hold for Q. The Euclidean vector norm of the main diagonal of Q must be equal or less than 1. Therefore, in order to maximise Equation 4.23, Q has to be the identity matrix.

Going back to Equation 4.22 and substituting Q with I:

$$I = V^T R^T U \rightarrow RV = U \rightarrow R = UV^T \tag{4.24}$$

This solution fails in certain cases when the determinant of R becomes -1, which makes it a reflection not a rotation. This has been resolved by *Kanatani* and *Challis*. After computing the SVD of C, we can maximise $tr(R^T C)$ (Equation 4.19) if:

$$R = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(UV^T) \end{bmatrix} V^T \tag{4.25}$$

The method will no longer fail in the cases where $det(UV^T)$ is -1 as R will have a determinant of $+1$.

Two main methods are used to increase the frame rate. Firstly based on the positions of the markers in the previous frame, a region of interest is defined to reduce the number of pixels in the blob detection phase as seen in Fig. 4.10(b).

Secondly, we interpolate the position and orientation in between frames. We use simple linear interpolation of position assuming constant speed. For orientation, we use SLERP [104], a method for linearly interpolating between quaternions. As the movement markers on the object are tied to the motion of the human hand, the system is not highly dynamic and thus such simple linear interpolations prove adequate.

On a desktop PC with an Intel Core i7-4790K (4 Cores, 8 Threads @ 4.0 GHz), 16 GB RAM and a Nvidia GTX 750 Ti running an Kubuntu 16.04 (64-bit) OS, the tracking runs at 46 FPS in static conditions and 41 FPS in dynamic conditions. The entire workflow split into programming jobs is visible in Fig. 4.12. The computing job runs on two threads, while the prediction and streamer jobs each run on one.

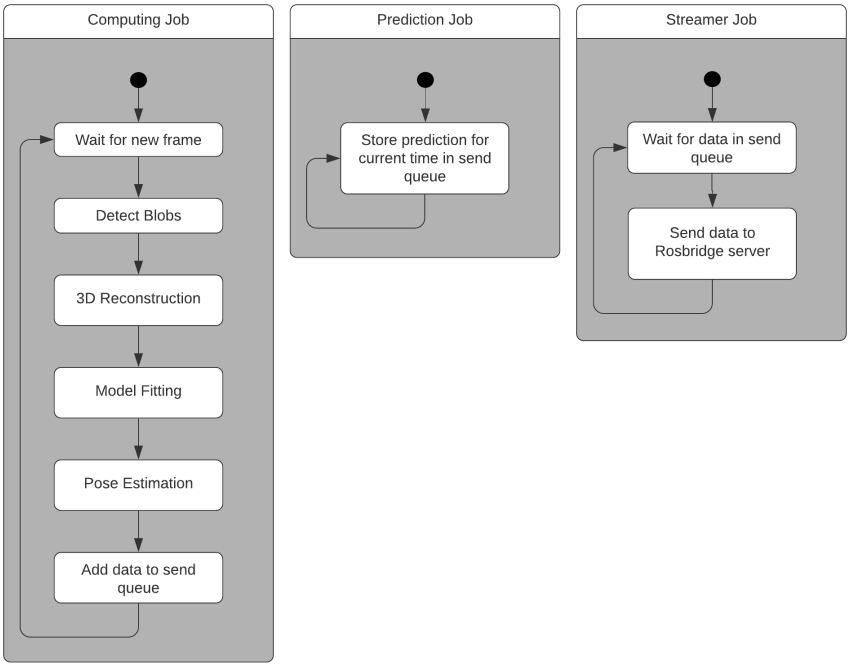


Figure 4.12: The workflow of the marker tracking algorithm is split into three jobs or processes - The Computing Job in charge of detecting markers in reflectivity frames and computing the input device's pose. The Prediction Job that interpolates the object's position and orientation in between frames. And the streamer job that streams the tracking data to the desktop computer running ROS. The Computing Job being computationally expensive runs on two threads, while other jobs run each on one.

4.2.3 Tests and Results

This section presents the tests and the results of the technical evaluations for each of the three proposed programming methods - waypoints, hand guidance and tool tracking.

4.2.3.1 Using Waypoints

Other papers already showed the user benefits of using such an approach. Quintero *et al.* [42] conducted a user study with ten participants. They showed that programming robots through holographic waypoints takes less teaching time and produces better results. They also found that the workload is decreased however mental workload is increased based on the NASA-TLX questionnaire. In a follow up paper based on an improved version of the system presented in [42], Chan *et al.* [105] found compared the results of a pleating task between performed purely by a human, assisted by a robot controlled via waypoints with a HMD or assisted by a robot and controlled by a joystick. It was found that using the robot to assist in the task decreases task completion time and physical load. It was also found that task completion time was reduced when using the HMD instead of the joystick. However, contrary to the findings in [42], there was no significant difference in the physical and mental loads between the joystick and HMD methods. It was also found that the participants utilized the robot more when using the HMD compared to the joystick, though the authors note that the novelty of AR may have skewed this result.

It is also worth mentioning the work of Frank *et al.*. Though they used an hand-held tablet for AR interaction and only indicated picking and placing locations, they had a large number of participants (40 in total). They employed their own 11 statement questionnaire with a 5-point scale for each statement, from strongly disagree to strongly agree. It was shown that the AR interface proved easy to use, was intuitive and was considered useful for completing the task. However no baseline on the execution of the task with a classical method was presented.

As the programming method is well researched and tested, we only tested the generation of collision-free trajectories already described in the robot workspace setup section (Sec. 4.1.3).

4.2.3.2 Hand Guidance

Technical tests of the hand guidance were performed using a scaling factor of $\mathrm{K} = 1$ and a Reflexxes interpolated joint position controller [106]. Moving each link, we compared the positions of the end-effector to the hand position, as well as tracked the desired joint state provided by our algorithm, the control signal of the Reflexxes controller, and the actual joint state of the robot. The robot used was an industrial KUKA KR-5 ARC manipulator with no JTS. In Fig. 4.13 and Fig. 4.14 one can see the example where link 3 is moved. The end-effector, and therefore link 3, motions follow closely the hand motions. One can also see that, although the hand commands suffer from shaking as they were unfiltered, the inertia of the robot filters out this signal and the joint states are smoother than the commands. That said a tunable band-stop filter, or signal smoothing for small, shaky hand motions could be a helpful addition.

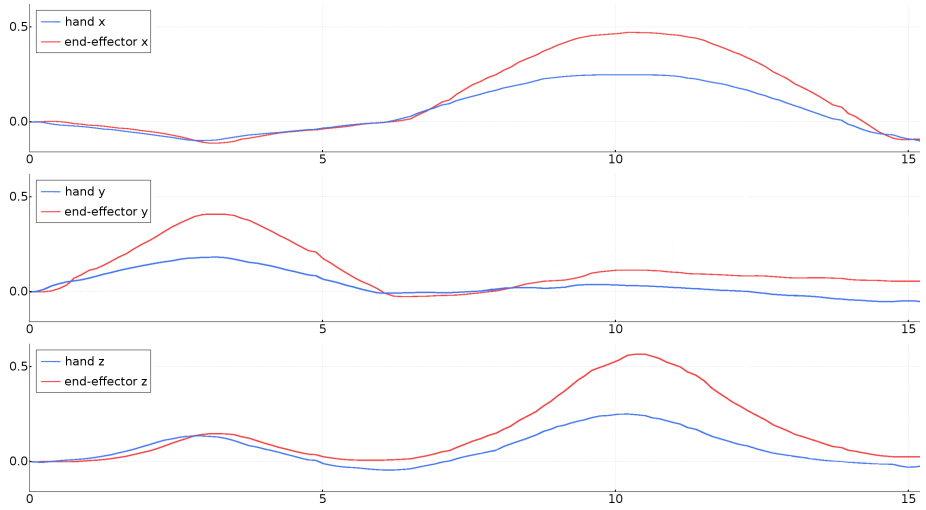


Figure 4.13: The x,y and z coordinate movements of the hand virtually pushing and pulling link 3 compared to the end-effector position. One can see that the end-effector movements follow closely the hand movements.

In Fig. 4.14 it is visible that the desired joint values follow closely what one would expect from our command strategy, with the base joint (joint 1) and elbow joint (joint3) receiving the majority of move commands. The non-zero move command from the shoulder joint (joint 2) is due to the fact that when $s_{3,t}$, h_t and h_{t-1} are collinear, the projection angle is zero, and therefore the next joint, joint 2, needs to perform the movement.

Finally we tested if there were any errors in replaying the trajectory captured by the hand motion. This is mainly to prove that there is no lost data when capturing joint states with their time stamps, or any unexpected stochastic behaviour in the system. In Fig. 4.15 one can see the comparison between the robot joint angles of the captured trajectory and the replayed trajectory. It can be seen that there are no deviations between them.

4.2.3.3 Tool Tracking

We performed two groups of tests. A static one where both the HoloLens and the tracked input device are static and a dynamic one where both the HoloLens and the input device move. To get directly the precision of the tracking we skipped the referencing step using the point cloud. We took 1000 static samples before each test to find a transformation that minimises the root mean square error between the ground truth position of the input device obtain via the ART-3 tracking system and the transform of the input device position in the HoloLens coordinate system.

For measuring the precision in static conditions, the input device and the HoloLens were both placed in stationary positions one arm-length away. After calibrating the world coordinate of the ART system, we gathered 5.000 samples of pose data for the pen pose seen from the HoloLens and the ART tracker.

To measure the precision in dynamic conditions, we moved randomly through the room again and gathered 10.000 samples of pose data in 5 experiments of 2.000 samples each. The referencing was performed at the start as previously described.

The static tests have shown an average absolute positional error of 1.9 mm and an average absolute angular error of $0.37°$. In the dynamic case an average absolute positional error of

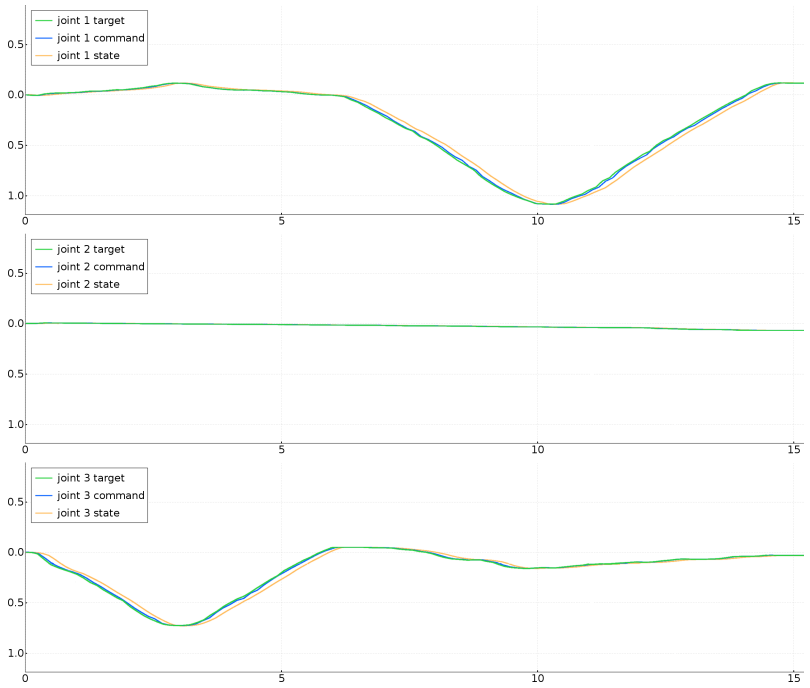


Figure 4.14: Angles of the three joints down the kinematic chain from link 3. The target is calculated by our algorithm, the command is the value the Reflexxes controller sends to the robot. The state is the actual angle of the robot joint. The controller doesn't distort the calculated target. The slight lag in the joint states is due to inertia.

22.1 mm and an average absolute angular error of $3.87°$ were observed. Fitting a Gaussian distribution to the positional errors gave the median of the positional error of -0.1 mm and the standard deviation of 30.7 mm. The angular error distribution had a median of $0.22°$ and a standard deviation of $5.39°$.

Interesting to see is the non-Gaussian distribution of the stationary tracking error. This is most likely due to the fact that the main error contribution is the IMU drift. Therefore the quality of the tracking was based purely on the average absolute error. On the other hand, while moving, independent error sources such as IMU drift, visual odometry errors etc. add to a Gaussian distribution as per the central limit theorem. Thus the error in the dynamic case does follow a Gaussian distribution. The biases found in the error distribution can then be used as ad-hoc corrections to the obtained tracking data.

The main contribution to the tracking error in the dynamic case seems to be the localisation error of the HoloLens, as is noticeable from the difference between the static and dynamic cases. In most cases the user will not move significantly while inputting trajectories, thus the error is expected to usually be much closer to the static case than the fully dynamic case.

Compared to four other state of the art commercial tracking devices (OptiTrack Flex3, Qualisys ARQUS and MIQUS and the Vicon Vantage), the static position error is four times higher than the average of the four devices (0.5mm) while the dynamic one is significantly worse at 45 times higher. The achieved frame rate however is only 2.5 times lower than the average of 100 frames per second.

In [107], *Bérard et al.* tested human input precision from various devices. It was shown that mouse and stylus type devices have a human input precision of around 0.5 mm while

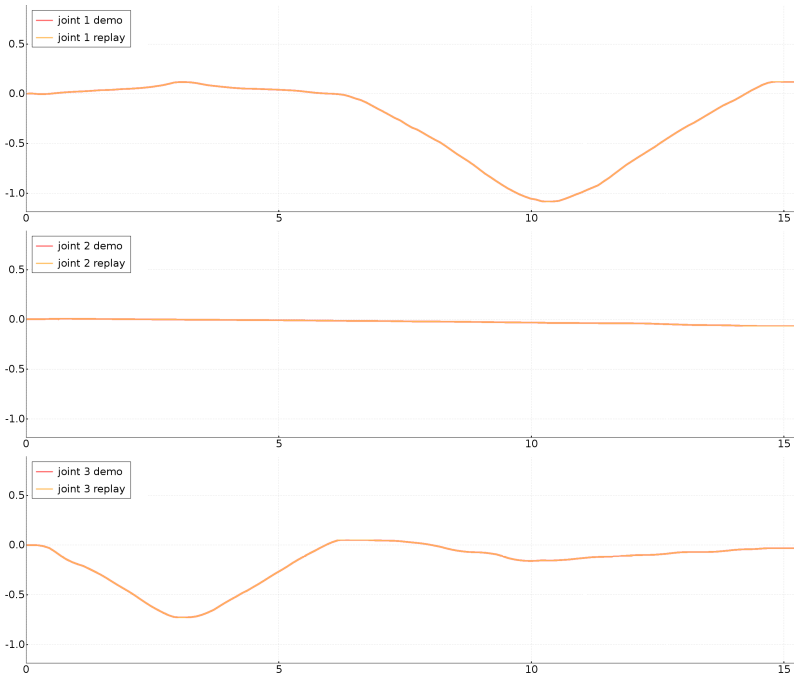


Figure 4.15: The joint values of the robot while inputting the desired trajectory compared to the joint values while the trajectory is replayed. One can see that the replay function doesn't introduce any errors - no data is lost in the saved trajectory compared to the execution and no unexpected behaviour is observed when executing safed trajectories.

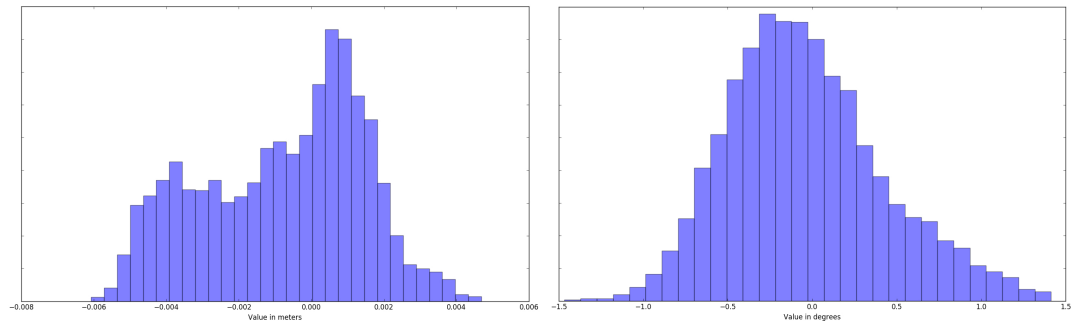
free-space devices have a precision of 5 mm. Thus the precision of the tracking in free space without significant HoloLens motion is adequate. Assuming that when the input device contacts a surface the precision will be approximately same to the mouse or stylus, the tracking precision will need to be improved in those cases.

There are ways to further reduce the tracking error to achieve the necessary precision. The most obvious one is to use newer generations of HMDs, like the HoloLens 2, which have improved tracking capability. Additionally the use of sensors on the robot itself may improve accuracy dramatically. In [98] it was shown that a laser line sensor on the robot can be used to vastly improve the accuracy of user input to achieve sub-millimetre precision. The input was given through the same input-device tracked here.

4.3 Conclusion

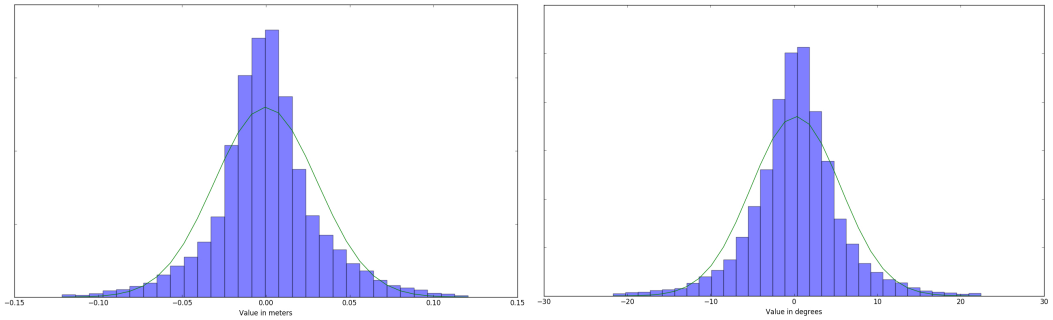
This chapter built upon the previously developed capabilities of mapping the environment and referencing the robot via a HMD. We presented the current issues with setting-up the workspace of robots. We have shown that the current methods are slow, expensive and require expert knowledge. We proposed an approach where the map of the environment is converted into a voxel occupancy grid. We have shown that this allows a quick setup of the workspace without expensive equipment. This allows higher level motion planners to plan collision-free paths, greatly easing the programming task.

We also reviewed the current robot programming paradigms and showed how the use of HMDs can offer more intuitive ways of programming robots. We implemented three



(a) **(b)**

Figure 4.16: Tests with moving IR markers and a static HoloLens. The ground truth is provided by an ART-3 tracking system: (a) Distribution of the positional error of 5000 data points. The average absolute error is of 1.9mm (b) Distribution of the angular error of 5000 data points. The average absolute error is of $0.37°$. To note is the non-Gaussian distribution of error due to the IMU drift being the major component of error.



(a) **(b)**

Figure 4.17: Tests with moving IR markers and a moving HoloLens. The ground truth is provided by an ART-3 tracking system:(a) Distribution of the positional error of 10000 data points. The average absolute error is of 22.1mm (b) Distribution of the angular error of 10000 data points. The average absolute error is of $3.87°$. In the dynamic case the error follows a Gaussian as per the central limit theorem due to multiple independent error sources.

different programming options. The first one uses waypoints and the voxel occupancy grid to plan collision-free paths. The position of the waypoints in the robot base coordinate system and the projection of the points to the surface beneath are quality-of-life additions. The second method sought to expand the hand guidance method to any robot, with or without JTSs or other external sensors. The third method is meant to ease the use of smart programming tools by implementing inside-out tracking on the HMD instead of relying on immobile marker tracking systems that need extensive setup and calibration.

Of particular note in this chapter is the marker tracking error of 2.21 cm in the case when the HoloLens moves compared to the static case in which the error was in the millimeter range. During tests no referencing was used. This was another proof that the localisation of the HoloLens itself has an error of around 2cm. When programming the robots, this error will need to be mitigated. One possible solution is the one presented by Hartman *et*

al. in [98], where the sensor on the robot is used to correct the error of the user's input. Alternatively the localisation precision of the HMD needs to improve considerably.

The part of the system pipeline for the robot environment setup is visible in Fig. 4.18. To the left are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS. In Fig. 4.19, 4.20 and 4.21 one can see the components for the programming via holographic waypoints, hand guidance, and smart tool tracking respectively.

We published several papers relevant to this chapter. In [33] we tested the mapping capabilities of the HoloLens, as mentioned in the previous section, and also showed how to use the map to generate a voxel occupancy grid from workspace setup. In [108] and the follow up paper [109] described the implementation of the HMD-based hand guidance system. In [110] we described the general system architecture for HMD-based HRI systems, including workspace setup, referencing, programming and interaction. In [34] we described the inside-out tracking via the HoloLens of tools equipped with IR markers.

In the next chapter we will present methods of safely interacting with the robot during the execution of its task. Tracking of the human coworker inside the robot's workspace is paramount to achieve this. Another important aspect is deducing the intentions and targets of the human coworker. The same methodologies allow us to implement collaborative tasks both with lightweight CoBots and standard industrial robots.

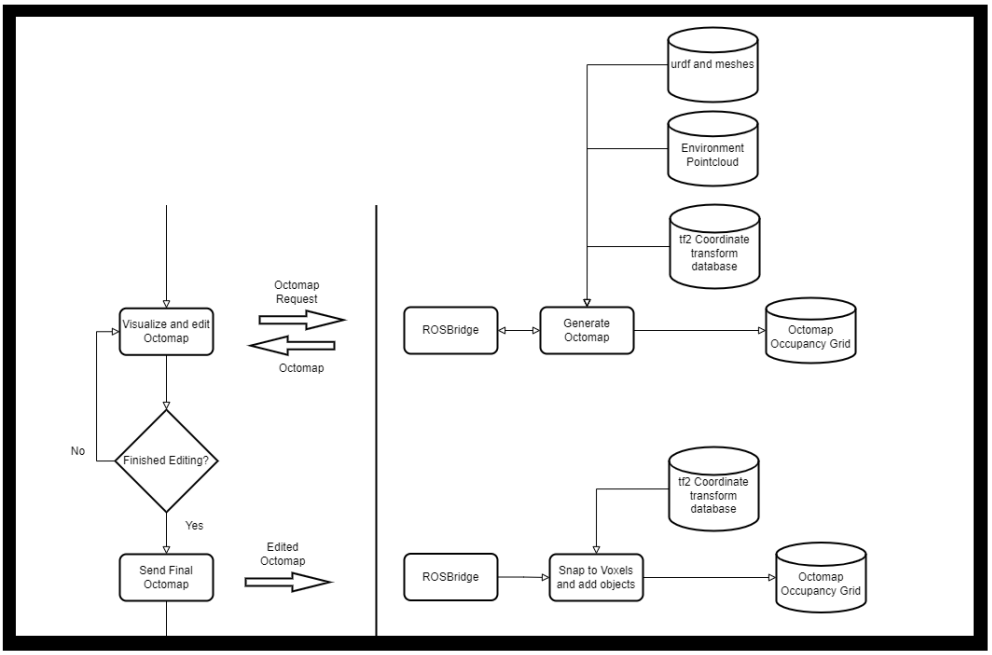


Figure 4.18: The setup components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS.

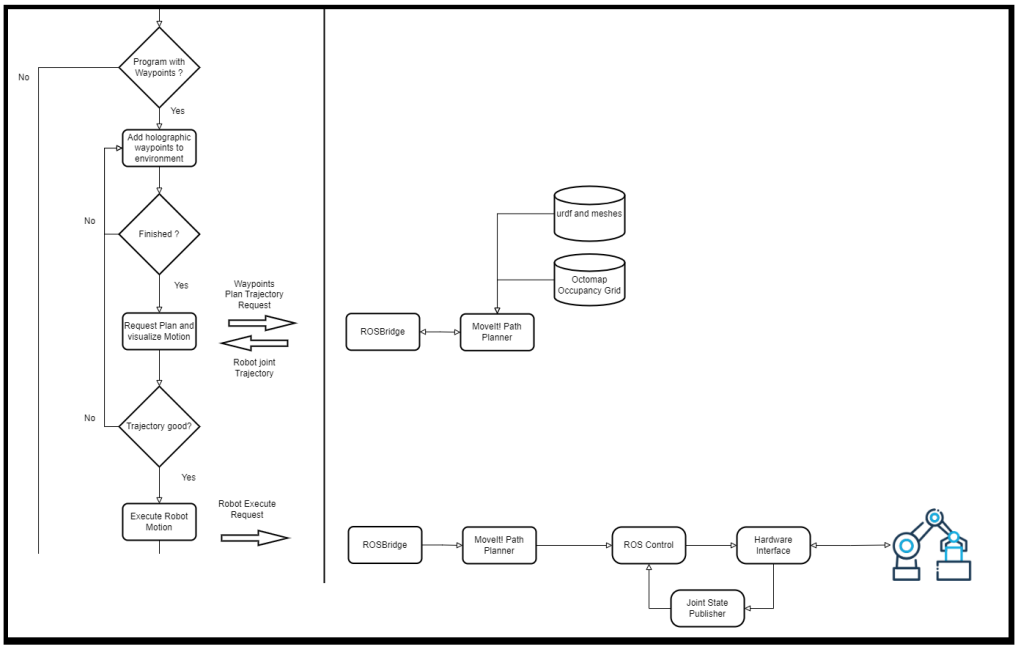


Figure 4.19: The mapping components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS.

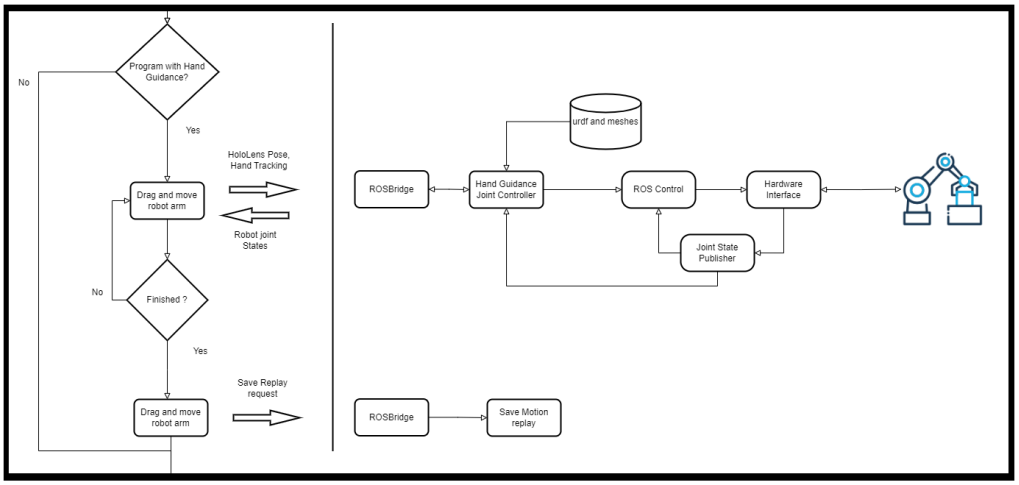


Figure 4.20: The mapping components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS.

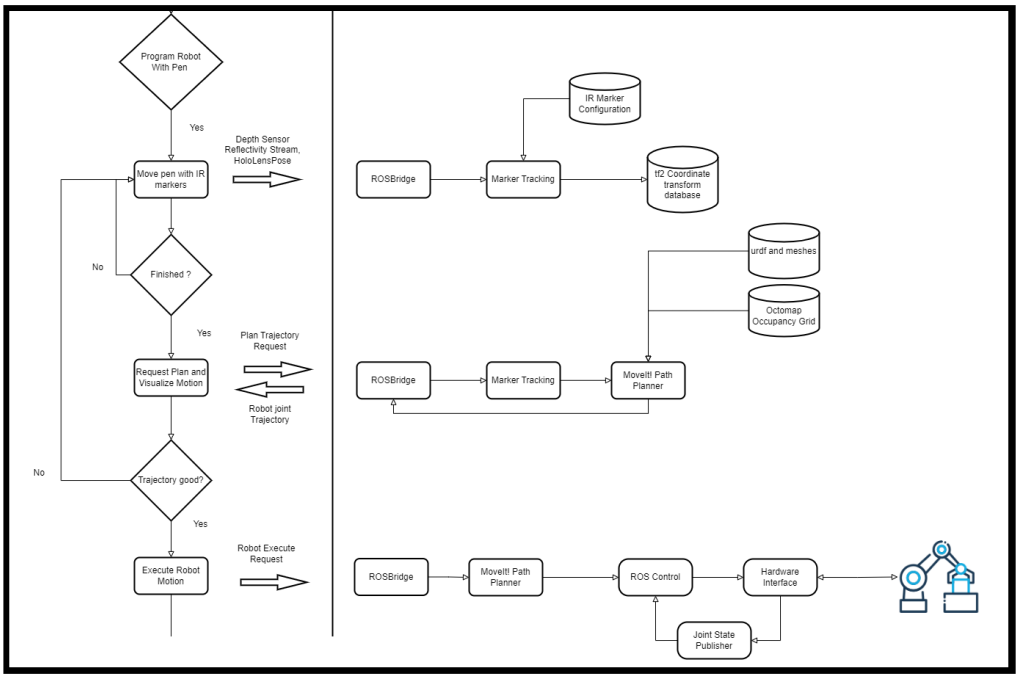


Figure 4.21: The mapping components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS.

5 Tracking and Intention Recognition

So far we have seen how the use of HMDs can facilitate robot setup and programming. A particularly important field in robotics is HRI. As robots get more widespread it becomes natural for them to interact and collaborate with humans. While at present the vast majority of robots are used in industry and isolated from humans, there is a strong push to reverse that, as human-robot teams often perform better than purely robots or humans. The human coworker can help the robot with their higher cognitive skill while the robot movements are faster, more precise and can easily carry heavy loads.

The main issue is ensuring that the robot doesn't injure the human during the interaction step, as most industrial robots are heavy and do not possess adequate sensors to detect collisions. Additional methods of ensuring the safety of the human need to be taken. This usually comes in the form of speed and separation monitoring - the position of the human is tracked, the speed of the robot is decreased as the human gets closer. The robot may be stopped if the human gets too close. The position of the human is tracked through external sensors (mounted in the robot workspace) such as cameras, laser sensors or pressure plates. We shall than once again endeavour to make a mobile, easier to use system with the same capabilities.

Once the safety of the human coworker is ensured an important component in any team, namely knowing the intentions of the other team member, needs to be addressed. Humans have a good sense of the intention of their human coworkers. Robot teams likewise can share data to facilitate their collaboration. Most humans do not have an intuition of the intentions of the robot. Likewise the robot needs an algorithm to try to predict the human's intention from sensor data. HMDs offer the display capabilities to properly visualize the robot's intention to the human, while possessing enough sensors to allow an algorithm to evaluate the intentions of the human.

5.1 Person Tracking

To ensure the safety of the human in the robot workspace knowing their position is paramount. This allows the usually heavy and rapidly moving robot to slow down around the human or to completely avoid the space surrounding the human themselves by replanning its trajectory. Finally we may stop the robot if the human gets too close. To allow the robot to perform its task as efficiently as possible, the safety zone around the human should be minimized as much as possible, while still ensuring their safety.

This section describes the current methodology of safely tracking people inside the working environment of the robot. These methods use sensors mounted around the workspace

of the robot and are therefore immobile and usually require extensive calibration to work properly. We propose a completely wearable method that does not require extensive calibration and is of course completely mobile.

5.1.1 State of the Art

The use of laser sensors or laser curtains is the de facto standard in industry at the moment [111]. The use of cameras, usually with redundant cameras and ToF sensors is also quite popular. Such systems include the commercially available PILZ *SafetyEYE* as well as the system presented in [112]. Vogel *et al.* used a projector and a multiple camera system to detect safety-zones violation [113]. Based on joint states and velocity, the projector makes a light curtain on the table around the robot. The cameras capture the contour and compare it to the predicted visible contour. If the two do not match a safety violation is detected and the robot stops. This can be thought as a reconfigurable laser curtain based on the robot state.

In [114] pressure plates were installed in the floor, with a projector only indicating the safety zones to the human coworker. The plates allowed discrete person tracking, meaning that multiple safety levels can exist in the system. Corrales *et al.* [115] used a motion capture gyro-suite to track joint rotations of a skeleton model of the human. The translation of the human however accumulated significant drift and soon became imprecise. To mitigate this a UWB-based tracking system was implemented for transnational tracking, rendering the system static.

The use of Deep Learning (DL) allowed the use of simple, cheap camera systems to track people, be it in the form of bounding boxes [116] or full skeleton tracking ([117], [118]). Although simpler and cheaper than previous methods, the system still requires multiple, fixed cameras to deal with occlusions and redundancy. Another thing to consider is the intrinsic safety issues of pure machine learning approaches. The classical CNN approaches suffer from a lack of explainability. Namely if an error occurs, it is hard to pinpoint exactly what caused the error. Likewise it may be hard to determine if the approach will work in all cases or when it might fail except through extensive testing.

Hoang *et al.* [119] use the HoloLens to setup safety zones manually. An additional automatic cylindrical safety zone is featured around the user and moves with the HoloLens. The size of the cylinder is however unspecified and we would like to avoid unnecessary waste of collision-free space for the robot.

5.1.2 Proposed Approach

Our approach consists of a HoloLens and two IMUs (MPU9250 9-DOF IMU on an Texas Instruments SensorTag CC2650) worn around the user's wrist. The sensor tags provide connectivity to the PC directly via Bluetooth. IMUs are compact enough to be worn by the user without almost any weight increase. They are also very cheap, meaning they would not increase the cost of the system significantly. IMUs measure linear acceleration in 3 axis, angular velocity in 3 axis and usually feature a magnetometer in 3 axis. To measure the pose i.e. the position and orientation, the IMU measurements need to be integrated or summed. This accumulate drift - the pose estimation error grows larger the longer

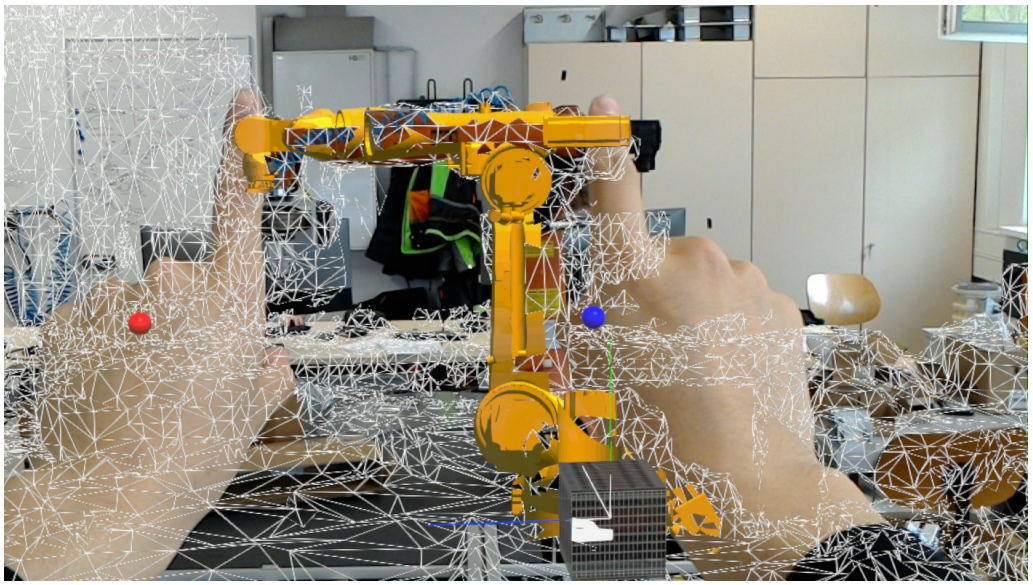


Figure 5.1: The start arm positions. One can see the differentiation between left and right arm, the referenced robot as well as the spatial mesh.

the system is used, like in the approach proposed in [115]. On the other hands SLAM algorithms do not accumulate significant drift [1]. Thus one way to exploit IMUs, while keeping the system mobile, is to pair them with a device that runs a SLAM algorithm, like the HoloLens.

We would like to minimize the safety space around the user, while still avoiding collisions. A body safety cylinder can be used for the head and body, like in [119]. The position of the cylinder is taken from the localization data of the HoloLens. Each of the arms are covered by two cylinders, one for the upper arm and one for the forearm and hand. The pose of the cylinders can be calculated from the wrist tracking data provided by the IMUs. The human shoulder has 3 degrees of freedom, like a spherical joint or three orthogonal rotational joints. The elbow has two degrees of freedom of which one, the rotation of the forearm around its axis can be ignored, as the cylinder is rotationally symmetrical around that axis. This gives four degrees of freedom to be determined, which is easy to calculate given the 6-DOF delivered by the IMUs. The kinematics of the human arm can be seen in Fig. 5.2.

The length of the upper arm and forearm, and therefor the length of the cylinders can be easily determined from the user height [120], though care has to be taken as it may not always be accurate [121]. The height itself is easily obtainable from the HoloLens, as an inbuilt function already estimates the ground plane for more robust localization.

We assume the user already mapped the robot's workspace and performed the referencing step. To start the tracking the user extends both arms in front of the HoloLens and performs the *Hold* gesture. The HoloLens does not differentiate between the left and right hand. We take the leftmost detection as the left hand and the rightmost as the right. We also use this method of differentiation whenever there are multiple hand detections from the HoloLens. This provides a well defined start position. After this simple setup the

[1] Provided the device re-observes and already observed landmark. This is called loop-closure and is the main method to bound drifting. Otherwise errors will still slowly accumulate producing a slight drift.

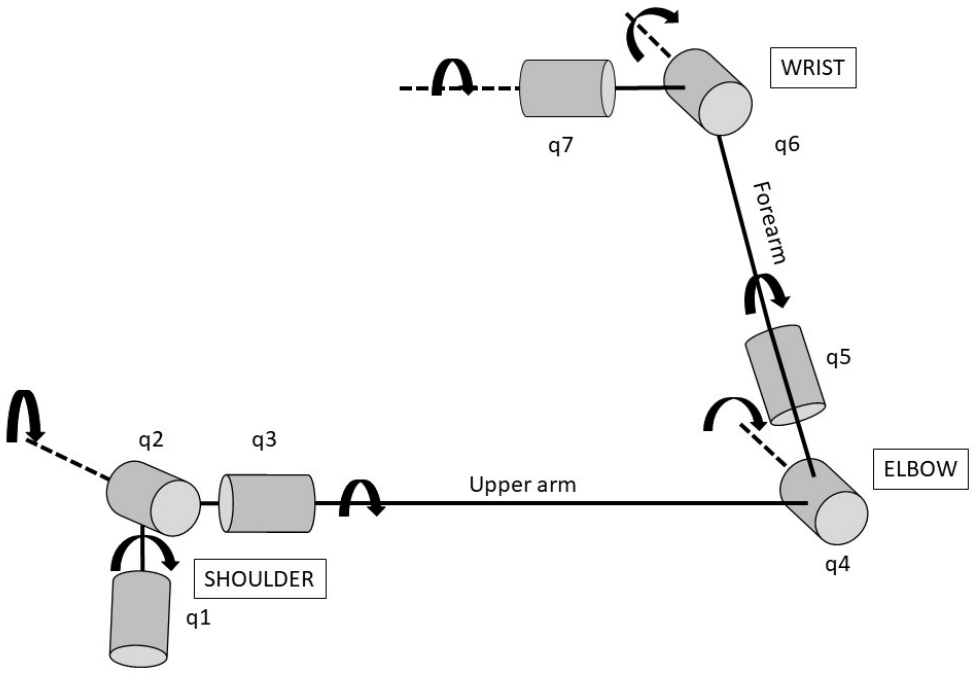


Figure 5.2: Kinematics of the human arm.

tracking system is activated, and the user can go about the robot cell performing their task. The starting position, together with the referenced robot can be seen in Fig 5.1.

To note here is that the HoloLens detects the position of the hand and not that of the IMUs themselves. However as IMUs provide only the relative position and orientation change from the start pose, this difference does not present a problem.

The orientation of the IMUs and the direction of gravity are estimated using the Madwick filter [122]. For position estimation we tested a Kalman filter and a combination of low and high pass filters. In the former we first pass the input through a median filter of window size three. In the later we reduce the measurement noise of the acceleration with a low pass filter. After the first integration giving speed and the second giving position we use high pass filters to partially eliminate drift.

To fuse the measurements from the IMUs and the HoloLens, in case of the Kalman Filter, we add the position as a measurement alongside acceleration. The position measurement is taken as the HoloLens measurement when the hand is in the FoV of the sensor, or as the previous position estimate in case it is not. For the low and high pass filter combination we simply substitute the previous value of the measurement with the HoloLens tracking data when available.

For both methods, first the gravity vector, estimated by the Madwick filter, is subtracted from the acceleration reading.

$$a_k = a_k^{imu} - g_k \tag{5.1}$$

In the case of the Kalman filter, we first run a median filter with window size 3 over the input signal. The filtered signal is then fed to a linear system without a control signal:

$$\hat{x}_{k|k-1} = \mathbf{A}\hat{x}_{k-1|k-1}; y_k = \mathbf{C}x_k \tag{5.2}$$

Where the state matrices were defined as in (5.3):

$$\mathbf{A} = \begin{pmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix}; \mathbf{C} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

The observation covariance matrix $\mathbf{R}$, transition covariance matrix $\mathbf{Q}$ and initial state covariance matrix $\mathbf{P}_0$ are defined as in (5.4):

$$\begin{gathered} \mathbf{R} = \begin{pmatrix} \sigma_{acc}^2 \end{pmatrix}; \mathbf{P}_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_{acc}^2 \end{pmatrix}; \\ \mathbf{Q} = \begin{pmatrix} 0.2 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 10^{-4} \end{pmatrix} \end{gathered} \tag{5.4}$$

Where σ_{acc}^2 is derived from the standard deviation of the IMU listed in the technical manual, in this case 1 mm .

Instead of the Kalman filter, a combination of low and high pass filters may be used to reduce drift. First, a low pass filter is applied on the acceleration vector:

$$a_k = \alpha_{lp} a_k + (1 - \alpha_{lp}) a_k \tag{5.5}$$

A high pass filter to partly mitigate drift is applied together with the summation to get the speed:

$$v_k = \alpha_{hp,v} v_{k-1} + \alpha_{hp,v} a_k \Delta t \tag{5.6}$$

And similarly for position:

$$s_k = \alpha_{hp,s} s_{k-1} + \alpha_{hp,s} v_k \Delta t \tag{5.7}$$

Where, in our case, $\alpha_{lp} = 0.2$, $\alpha_{hp,v} = 0.99$ and $\alpha_{hp,s} = 0.95$.

We regard the hand tracking position of the HoloLens as precise. We transform the hand position as detected by the HoloLens to the IMU reference frame, using the transformation matrix calculated during the calibration step:

$$s_{t,imu} = T_{hl}^{imu} \cdot s_{t,hl} \tag{5.8}$$

In the case of low and high filter combination we merely substitute s_{t-1} in (5.7) with $s_{t,imu}$ from (5.8). In the case of the Kalman filter we expand the system to include position observations. The position observations are taken either as $s_{t,imu}$ when the HoloLens measurement is available, or as the previous a posteriori estimate $\hat{x}_{k-1|k-1}$ when that is not the case. The observation matrix then becomes:

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \tag{5.9}$$

and the observation covariance matrix:

$$\mathbf{R} = \begin{pmatrix} 10^{-4} & 0 \\ 0 & \sigma_{acc}^2 \end{pmatrix}; \tag{5.10}$$

Note that the observation covariance for the position is purposely kept low to indicate the high fidelity of the measurement, be it the HoloLens hand detection or the a posteriori estimate, which are both the most precise measurements of the system.

The results of the tracking algorithm in RViz can be seen in Fig. 5.3. One can see all the different coordinate systems and transforms necessary.

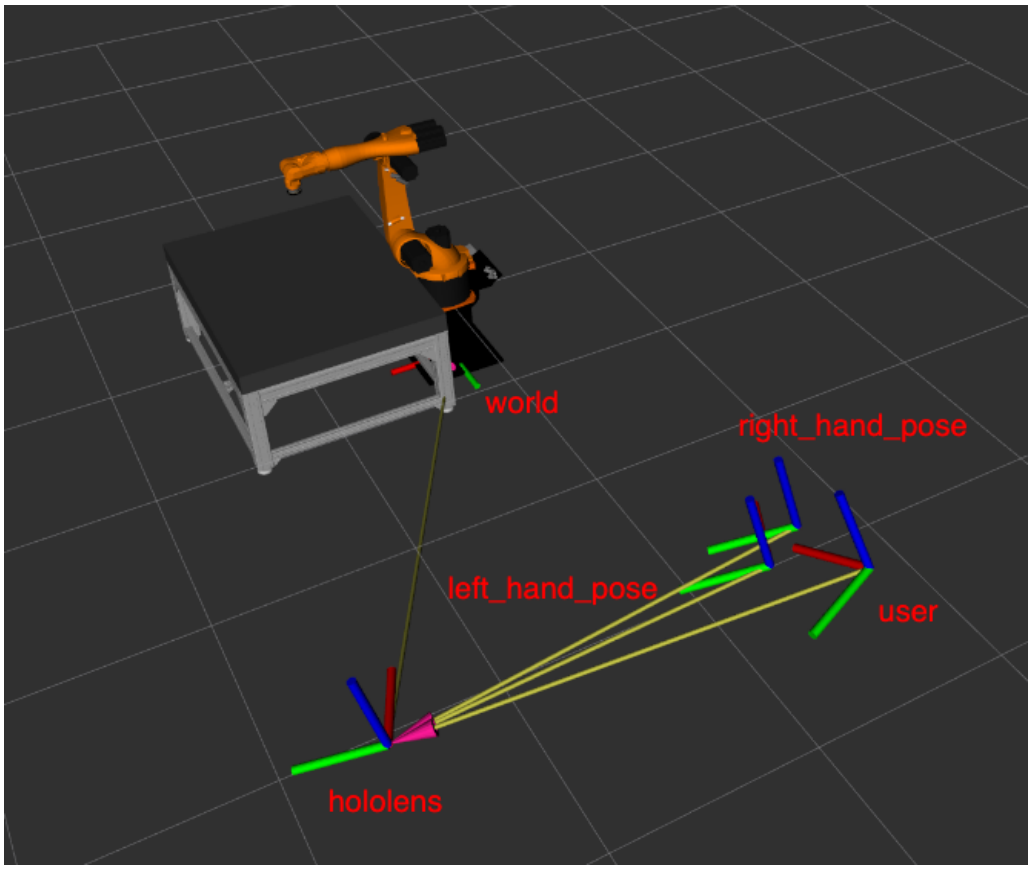
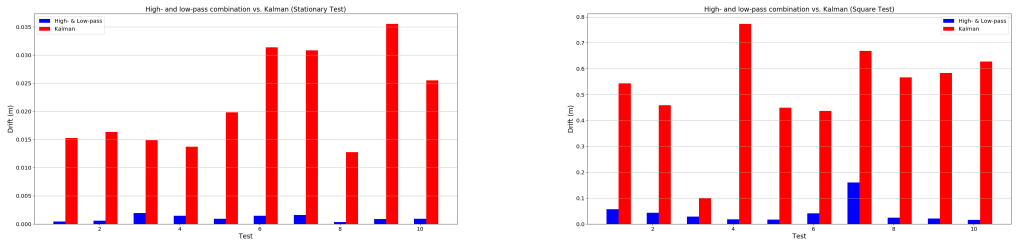


Figure 5.3: The results of the hand tracking in RViz. The world coordinate system of the HoloLens, the current position of the HoloLens, the position of the hands and the referenced robot can all be seen.

5.1.3 Tests and Results

First we conducted two set of tests to determine which filtering strategy, the Kalman Filter or the low pass and high pass filter combination, performs better. In the first set of tests the IMU was left stationary for 20 seconds. In the second set of tests we printed out a 20x20cm square track, over which a person was moving the IMU following the track as



(a) **(b)**

Figure 5.4: Kalman Filter in red, combination of high pass and low pass filters in blue. (a) The error accumulation for 20s while the IMU was completely stationary. (b) The error accumulation when the IMU was moved in a square pattern.

precisely as possible. This would not only give us a dynamic error, but the dynamics would be consistent with the natural movement of a human arm. Each set constituted of 10 repetitions of the same experiment. The results can be seen in Fig. 5.4. One can clearly see that the approach using the Kalman filter consistently performed much worse in all the tests, both stationary or dynamic. Therefore it was decided to use the combination of the low pass and high pass filters in further tests.

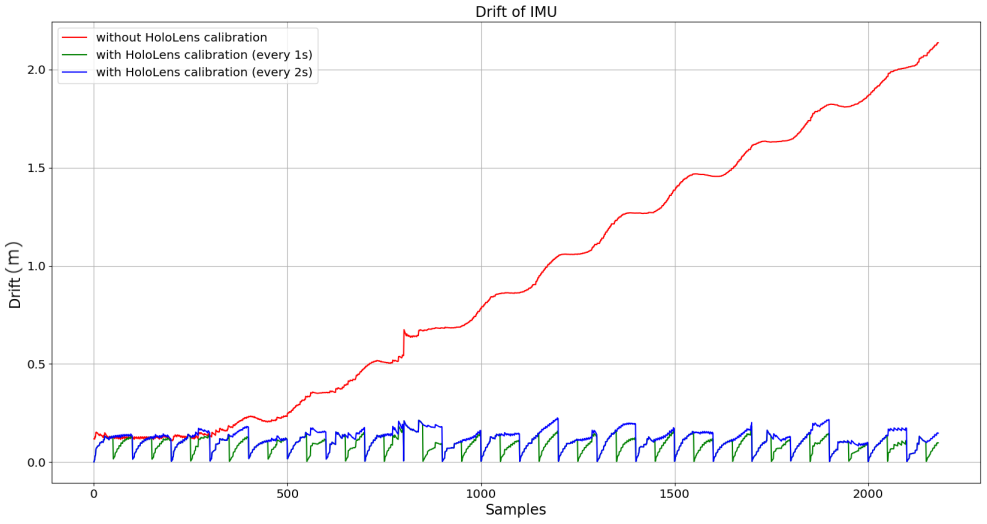


Figure 5.5: The results of the hand tracking with the HoloLens hand detection as baseline. Pure IMU data in red, IMU data with HoloLens tracking provided every second in green and every two seconds in blue. One can see that the error is bounded provided that the HoloLens detects the hand from time to time. If there is no detection from the HoloLens the drift is clearly noticeable.

Next we wanted to test the hypothesis that fusing the HoloLens SLAM localisation data will help bound the drift. As an addendum, we put forward the hypothesis that the more often the hands came into the field of view the less of an error we would have. As we know that the HoloLens SLAM localisation error is bounded and around 2cm, we wanted to see how do the IMUs tracking behaved with HoloLens hand tracking taken as the baseline.

To provide a baseline the hand was held in the FoV of the HoloLens' tracking sensor at all times in the *"Hold"* position. The user may move the hand and walk around, provided the

hand is always visible to the tracking sensor. We artificially fed the HoloLens hand tracking data every 1 second, 2 seconds or never. The results are visible in Fig. 5.5. One can see that without the HoloLens hand detection the error is not bounded and the measurement drifts significantly (red curve). On the other hand, when the HoloLens detection are fed to the sensor fusion, the error remains bounded. The addendum has also been proven, in the sense that the more often the hand is seen by the HoloLens the less the average error (every 1 second in green, every 2 seconds in blue). On the other hand the maximum error is not that different, meaning that the maximum error is not that sensitive to the amount of time between corrections. In Fig. 5.6 one can see an example of the 3D hand movement measured by the IMU with HoloLens corrections (green points) compared to the hand movement detected by the HoloLens itself (blue curve). In this example the HoloLens data was sent to the sensor fusion every 2 seconds.

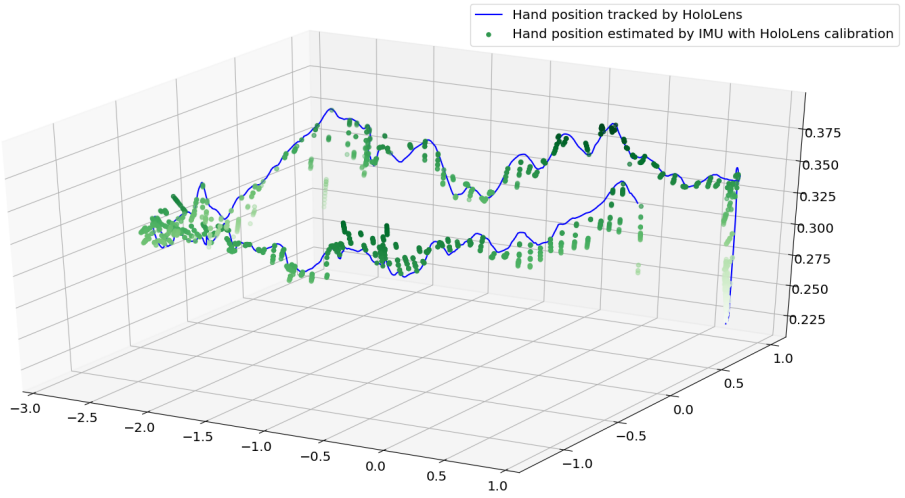


Figure 5.6: The 3D hand movement detected by the HoloLens (blue) and reconstructed from the IMU with HoloLens corrections (green points). The HoloLens hand tracking data was provided to the sensor fusion every two seconds. This served to simulate the hands leaving the FoV of the HoloLens without setting up an external tracking system, and provided a rough base for a qualitative analysis of the proof of concept.

5.2 Human Intention Recognition

Another important element in providing safety to the human coworker, as well as increasing overall efficiency, is the ability for the robotic system to recognize human intentions in real time. Provided the robotic system recognizes the intention and the goal of the human coworker, it has all the necessary information and enough time to plan a path that keeps the human safe and does not interfere with their work, while at the same time retaining optimal movement. Without intention estimation, the robot has to rely on reactive, sensor driven control, which is sub-optimal and may pose the human to more risk. Beyond these basic benefits, intention sharing is a crucial component in any team. Human teams exhibit both implicit and explicit intention sharing. To further development of human-robot collaboration, intention recognition is crucial on both sides.

This section will present HIE based on the HoloLens. The HIE algorithm was developed by Tomislav Petković, a project partner on the project SafeLog. Though the original algorithm was developed for use in autonomous warehouses with Automated Guided Vehicles (AGVs), we will also describe how to adapt the same approach for use in a collaborative scenario with an industrial robot.

5.2.1 State of the Art

Anh and Pereira [123] reviewed the area of human intention recognition research, emphasizing its potential applications in decision making theory.

Lin *et al.* [124] used a fixed camera to observe objects and gestures that a human was performing during preparation of a cup of coffee. The motivation was for a robot to recognize human actions and help the human prepare the coffee by completing other steps in parallel. The intention recognition was based on Markov decision processes.

In [125], Schlenoff *et al.* proposed an intention recognition system in the manufacturing kitting domain. States are represented by a combination of spatial relations along with cardinal direction information. The motivation is to train a robot assistant to help the human coworker assemble a kit with various objects, such as nuts and bolts, that are then provided to workers on the assembly line.

Bascetta *et al.* [126] proposed a human tracking and intention recognition system in an industrial robot cell with an overhead camera. The human is tracked and based on the position and velocities the intention was determined through a Hidden Markov Model (HMM). The cell was divided into 5 areas, with each area being predetermined for collaboration, coexistence or interface. Such a system is intrinsically static and the areas and modes of interaction in each area predetermined, limiting its flexibility.

In [127], intention recognition was used to estimate desired human motion during a teleoperated robot welding task. The user performs the task remotely using VR. The intention recognition is used to smooth the raw user input and provided smoother motions and better welds, than using raw user motions.

A system based on an HMD and intended for human-robot collaborative task planning was presented by Chakraborti et al. [128]. In the presented system, however, the human coworker had to explicitly select and reserve objects it wished to interact with, slowing down task execution and increasing the physical and mental workload of the human worker.

Most of the approaches rely on the motion of the human itself. However in [129] gaze was identified as an important factor at predicting human intentions. The correct human intention while playing the game Ticket to Ride, was estimated 71% of the time, while without it only 47%.

On the other side, showing robot intentions to the human coworker has already been well researched. Dragan *et al.* [130] proposed a method to make robot motions more legible to humans. However this requires adaptation of robot paths, making them less efficient. In [131] proposed the use of AR as a way to communicate robot intentions to the human. Such methods would not require adaptation of the trajectories. Walker *et al.* [132] proposed different AR modalities to communicate motion intentions of a

drone. These include showing the planned trajectory of the drone through waypoints an lines with arrows, representing the drone as a floating eye whose gaze indicates the intention of motion, or through other utilities such as a minimap - a HUD map that shows the current position of the drone and the planned destination, and a pointer to show the current location of the drone.

5.2.2 Proposed Approach

We will first introduce the approach proposed by Petković et *al.* in [32] for estimating human goals based on their motion in an automated warehouse with AGVs. For a more detailed description of the theoretical background, the reader is encouraged to refer to the paper.

The system is described by a HMMs. A HMM is an extension of a Markov Decision Proces (MDP) discrete time stochastic process represented as tuple $(\mathcal{S}, \mathcal{A}, T, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is set of states and $\mathcal{A}$ is set of actions. After an action $a \in \mathcal{A}$ is taken, the system moves from the current state $s \in \mathcal{S}$ to a new state $s' \in \mathcal{S}$. The conditional probability function $T(s, a, s') = p(s'|s, a)$ gives the probability that the system lies in s' after taking the action a in state s. Taking an action yields an immediate reward $R(s, a)$. The goal of the system is to choose the sequence of actions that maximizes the expected total reward $E\big(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_T)\big)$. A discount factor $\gamma \in (0, 1)$ reflects the preference of immediate rewards over future ones. The HMM is an extension of MDP where the observation is also a probabilistic function of the state, meaning that the underlying process is not directly observable and, therefore, hidden.

To encode the environment a Voronoi diagram is used which partitions a plane into regions based on distance to predefined points. Each region corresponding to a predefined point contains all the points closer to the corresponding point than to any other. The Generalized Voronoi Diagram (GVD) is a discrete form of the Voronoi diagram defined as the set of points in free space to which the two closest obstacles have the same distance [133]. The GVD can be constructed from an occupancy grid map, obtained either through an existing floor plan or by mapping the environment vie e.g. SLAM. For estimating the goal of a worker in an automated warehouse, a floor plan is available and therefore used to generate the GVD.

The HIE is divided into an offline and online part. In the offline part the GVD is constructed based on the occupancy grid of the floor plan of the warehouse. It is possible to insert additional nodes at arbitrary locations, but erasing generated ones except for dead ends, is not allowed as it would impede graph's connectivity. Goal nodes are therefore added and dead ends discarded creating a set of nodes $\mathcal{N}$. It is assumed there is a finite number of points of interest for the worker in the warehouse (e.g. the location of the task, exits etc.). The D* algorithm [134] is used to find the optimal paths between each two nodes and all the relative distances are saved in a matrix $\mathbf{F}$, where element $\mathbf{F}_{i,j}$ denotes distance in pixels between nodes i and j.

In the online part, the pose of the worker is tracked via the HoloLens HMD and mapped to the occupancy grid. The pose of the mobile robots is obtained through the Fleet Management System (FMS) of the automated warehouse. The FMS is in charge of storing the statues of each mobile robot as well as send them commands. The robots are represented as mobile obstacles with radius $r = 1\,\text{m}$. If a robot is located on a GVD edge,

the edge is cut from the graph and the relative distance matrix **F** is updated using the D* algorithm. With each pose update from the HMD, if the position on the occupancy grid has changed or the orientation has changed more than $\frac{\pi}{8}$ from the last update, an intention estimation update is performed. First the association vector **c**, which associates the observed worker's pose with each of the nodes in set $\mathcal{N}$, is calculated:

$$c_i = \begin{cases} 0, & \text{an unobstructed straight line exists} \\ & \text{between the worker and the } i\text{-th node} \\ \mathcal{G}(d_i, \sigma^2) \cdot \Phi(\theta_i), & \text{otherwise.} \end{cases} \tag{5.11}$$

Where $\mathcal{G}$ is a Gaussian function with variance $\sigma^2 = 0.005$, obtained experimentally, and d_i is the distance between worker's position and i-th node's location. The Gaussian is modulated with the triangular function:

$$\Phi(\theta_i) = \frac{\pi - |\theta_i|}{\pi^2}, \tag{5.12}$$

Where $\theta_i \in [-\pi, \pi]$ is the difference between worker orientation and the angle at which the worker sees i-th node. The function amplifies the association with those nodes the worker is oriented at, as it is assumed the worker looks at the path it is planning to follow.

The isolation matrix $\mathbf{I}^{n \times g}$, where n is the total number of nodes and g is number of goal nodes, is defined as:

$$\mathbf{I}_{i,j} = \begin{cases} 1, & i = n - g + j \\ 0 & \text{otherwise.} \end{cases} \tag{5.13}$$

The association vector **c** is normalized and the modulated approximate distance vector **d** is obtained by multiplying it further with the distance matrix **F** and isolation matrix **I**:

$$\mathbf{d} = \mathbf{cFI}. \tag{5.14}$$

Each element of vector **d** represents the modulated distance to the respective goal.

To find out if the worker is moving towards or away from the goal,this distance is compared to alternative worker positions and orientations. The difference r between the location of the previous estimation update l' and current worker's position l is calculated. Then, a set of $m = 16$ equidistant points p on a circle around l' with the radius r is generated.

Then, a set of $m = 16$ equidistant points p on a circle around l' with the radius r is generated. For each point $p_i \in p$ and potential worker orientation $\theta \in \{-\pi, -\frac{3\pi}{4}, ..., \pi\}$ the calculation of vectors **c** and **d** is repeated and the result is appended to the potential modulated distances matrix $\mathbf{D}$. Computing the matrix **D** enables the validation worker's motion with respect to states it *could be in*, rather than to the state it *had been to*.

The distance vector **d** and the distance matrix **D** is used to validate worker's actions through the motion validation vector **v**:

$$\mathbf{v} = \frac{\max\limits_{1 \le i \le n} \mathbf{D}_{ij} - \mathbf{d}}{\max\limits_{1 \le i \le n} \mathbf{D}_{ij} - \min\limits_{1 \le i \le n} \mathbf{D}_{ij}}. \tag{5.15}$$

If the worker is moving towards the goal, the corresponding value of **v** will be close to unity, and if it is moving away from that goal, the corresponding value will gravitate to zero. Before further calculations a discrete first-order low-pass filter on **v** is applied in order to reduce the noise influence.

While worker's actions, manifested as moving and turning, are fully observable, they depend on the worker's intentions, which cannot be observed and need to be estimated. A HMM is constructed with $g + 2$ hidden states, where g is a set of G_i known goals in the warehouse. The other two states are $G_?$, indicating that the intention estimation cannot decide which goal the worker is moving to, and G_x which indicates that the worker behaves irrationally i.e. doesn't appear to have a goal. The state G_x may be an indication that the worker is in distress and can be used to send help. On the other hand the worker may be moving to an unspecified goal.The hidden states allow the HIE model to save the intention estimation history as probabilities $P(G_i)$.

First the transition matrix $\mathbf{T}^{g+2\times g+2}$ is calculated:

$$\mathbf{T} = \begin{bmatrix} 1-\alpha & 0 & \dots & \alpha & 0 \\ 0 & 1-\alpha & \dots & \alpha & 0 \\ \vdots & & \ddots & & \vdots \\ \beta & \beta & \dots & 1-g\beta-\gamma & \gamma \\ 0 & 0 & \dots & \delta & 1-\delta \end{bmatrix}, \tag{5.16}$$

Where the parameters have been obtained experimentally as follows: $\alpha = 0.5$, $\beta = 0.1$, $\gamma = 0.05$, $\delta = 0.1$. The architecture and description of the matrix parameters can be seen in Fig. 5.7. These parameters are set based on the desired behaviour of the system, e.g. length of transition between states, how fast the algorithm becomes certain for a specific goal etc.

Every time the worker moves or turns significantly, we estimate the worker intention using the Viterbi algorithm [135], which is often used for solving HMM human intention recognition models [136]. The inputs of the Viterbi algorithm are the hidden states set $S = \{G_1, ...G_g, G_?, G_x\}$, hidden state transition matrix **T**, initial state Π, the sequence of observations **O**, and the emission matrix **B**.

Though the HMM framework generally assumes a discrete set of observations, the validation vector **v** with continuous element values, represents the observation in this case. Therefore an expandable emission matrix **B** is introduced and taken as input to the Viterbi algorithm. Once a new validation vector v is calculated, the emission matrix is expanded with the row $\mathbf{B}'$, where the element $\mathbf{B}'_i$ stores the probability of observing **v** from hidden state G_i. The average of the last m vectors **v** is calculated and the maximum average value ϕ is selected. It is used as an indicator if the worker is behaving irrationally, i.e., is not moving towards any predefined goal. The value of m decides how much evidence is to be collected before the algorithm declares the worker irrational. If the worker has been moving towards at least one goal in the last m iterations ($\phi > 0.5$), $\mathbf{B}'$ is calculated as:

$$B' = \zeta \cdot \begin{bmatrix} \tanh(\mathbf{v}) & \tanh(1-\Delta) & 0 \end{bmatrix}, \tag{5.17}$$

and otherwise as:

$$B' = \zeta \cdot \begin{bmatrix} \mathbf{0}_{1\times g} & \tanh(0.1) & \tanh(1-\phi) \end{bmatrix}, \tag{5.18}$$

where ζ is a normalizing constant and Δ is the difference of the largest and second largest element of **v**.

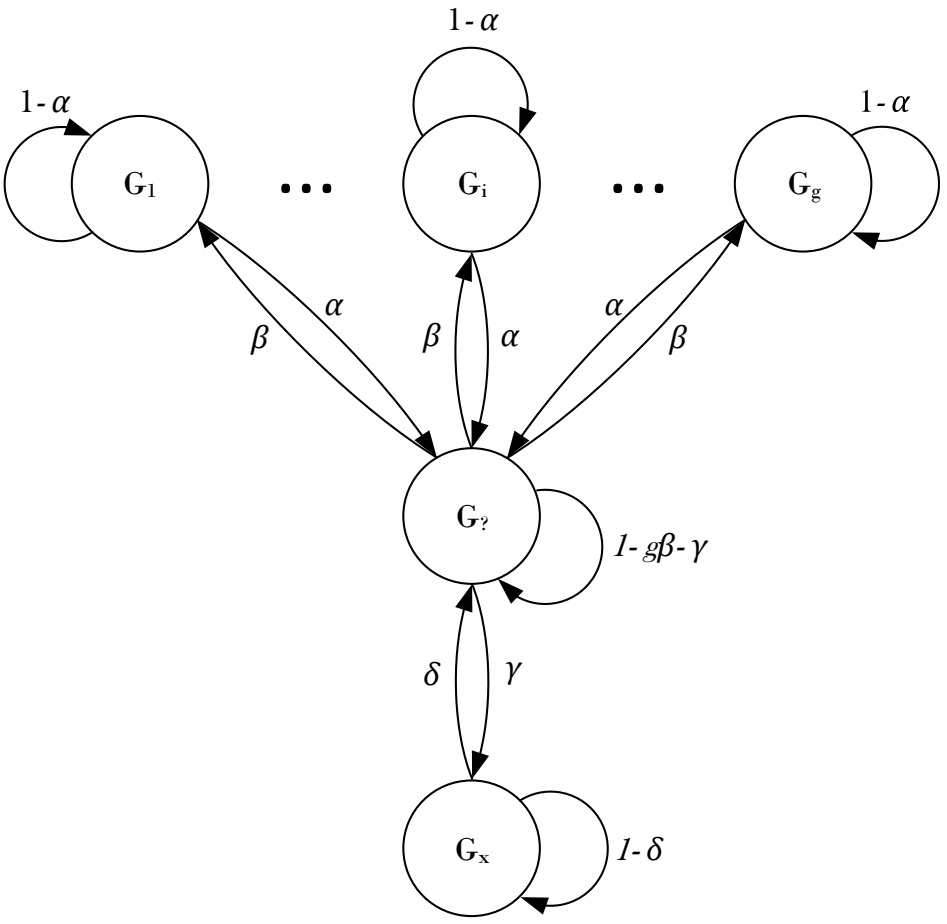


Figure 5.7: HMM architecture for HIE. Worker's change of mind tendency is captured by the parameter α and the parameter couple β and γ set the threshold for estimating intention for each goal location. Increasing β leads to quicker inference of worker's intentions and increasing γ speeds up the decision making process. Parameter δ captures model's reluctance to return to estimating the other goal probabilities once it estimated that the worker is irrational.

Finally, the initial probabilities of worker's intentions are defined as:

$$\Pi = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 \end{bmatrix}, \tag{5.19}$$

indicating that the initial state is $G_?$ and the model does not know which is the worker's intended goal.

The Viterbi algorithm then outputs the most probable hidden state sequence and the probabilities $P(G_i)$ of each hidden state in each step. These probabilities are the worker's intention estimates.

Goals can change during the runtime of the HIE. Therefore it is possible to add and remove goals during runtime. If a goal is removed, the goal is discarded from the calculation and added to the unknown goal intention estimation $G_?$. If a goal has to be added, its intention estimation is set to $\max(\min(P(G_i), 0.1)$ with $0 < i < g$ and the **I** and **T** matrices are expanded.

The maximum number of goals this model can handle is limited because elements of transition matrix **T** must be greater than 0. The only value of matrix **T** that depends on goal number is probability of staying in $G_?$ state and equals to $1 - g\beta - \gamma$. The maximum number of goals is calculated by applying the positivity condition to that expression:

$$g_{max} = \left\lfloor \frac{1-\gamma}{\beta} \right\rfloor, \tag{5.20}$$

For the current value of parameters the model supports a maximum of 9 defined goals. In practice if there are more than 5 defined goals the model will continuously estimate $G_?$ as the most probable state.

For interaction with an industrial robot we assume the human is sitting or standing in front of a table shared with the robot, on which objects are placed in predefined positions. Instead of the position of the human coworker as in the original paper, we consider the position of the hand in relation to goal objects. To simplify the calculations, we assume there is a straight line between the hand position and each goal object. By doing that we can forego the complex path planning step to determine the modulated distance and instead use the euclidean distance to calculate the vector **d** that represents the distance of the hand to each goal.We define an additional 32 points (instead of 16) p_i on a circle around the previous hand position l' and a radius r equal to the distance between the current l and the previous l' hand positions. We calculate the vector **d** for each point p_i and append them to the modulated distance matrix **D**.

Additionally we consider the gaze validation **s** of the HMD. The motivation being that the user is more likely to look approximately towards the goal of the hand motion than towards other goals. The gaze validation is calculated as:

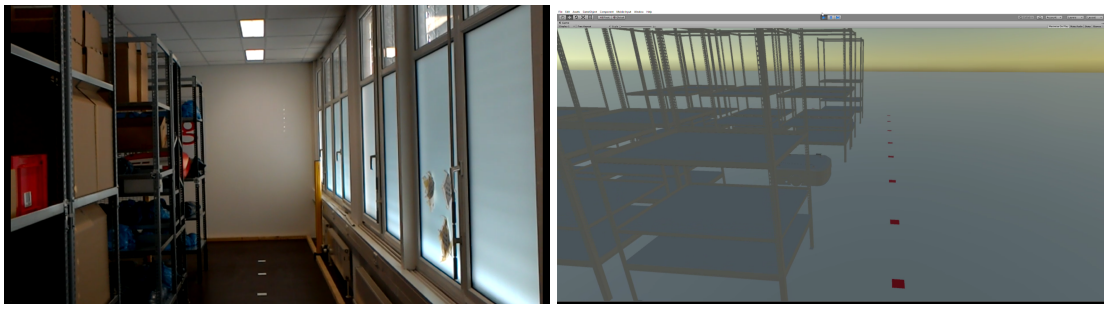
$$\mathbf{s}_i = \begin{cases} \mathbf{g} \cdot \dfrac{\mathbf{o}_i - \mathbf{h}}{||\mathbf{o}_i - \mathbf{h}||}, & \mathbf{g} \cdot \dfrac{\mathbf{o}_i - \mathbf{h}}{||\mathbf{o}_i - \mathbf{h}||} \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{5.21}$$

Where **g** is the HMD orientation in the world coordinate system, $\mathbf{o}_i$ is the position of object i and **h** is the position of the HMD. We expand the motion validation vector **v** as follows:

$$\mathbf{v} = \frac{\max\limits_{1 \leq i \leq n} \mathbf{D}_{ij} - \mathbf{d}}{\max\limits_{1 \leq i \leq n} \mathbf{D}_{ij} - \min\limits_{1 \leq i \leq n} \mathbf{D}_{ij}} \cdot \mathbf{s} \tag{5.22}$$

The rest follows exactly the previously described algorithm.

In regards to communicating robot intentions, the already shown methods of displaying holograms over objects the robot wishes to interact with, or even virtual execution already presented in Sec. 4.2 may be used. Therefore we did not pursue any research in that direction. One interesting addition would be the use of sound cues in parallel to the holograms through spatial sound (virtual sound sources that emulate direction and distance). This method would mitigate the relatively narrow FoV of current HMDs.



(a) (b)

Figure 5.8: First person view of the small test environment as viewed from the HoloLens (a) and as viewed from the player perspective in VR (b).

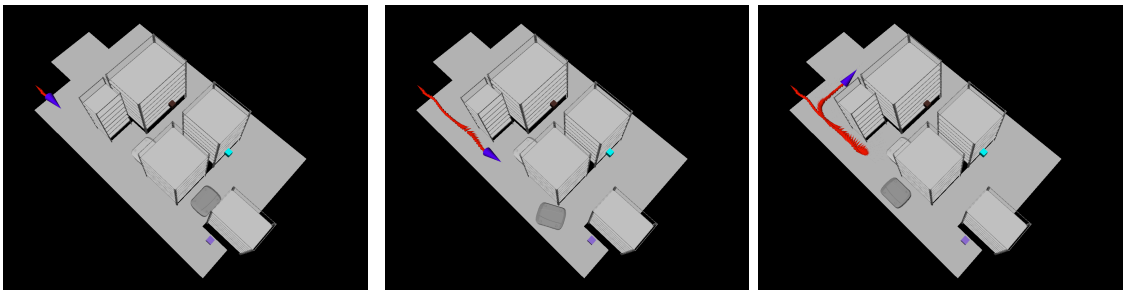
5.2.3 Tests and Results

First we will described the results obtained in the original paper that we co-authored with Petković. This will show that the HIE algorithm delivers precise estimations. We will then proceed to describe the results of adapting the algorithm to interactions with an industrial robot.

In [32], tests were conducted both in a small scale real warehouse, used for system testing, with a user wearing the HoloLens HMD as well as in larger scale virtual warehouses, where a real human user moved around in VR. First, experiments were performed in the real warehouse with the user wearing the HoloLens. A mobile robot was moved in the warehouse by manual control (due to safety reasons). The HoloLens tracked the user's position and orientation and this data was the input to the HIE algorithm. Then the experiments were repeated with the same setup, goals and robot movement but in VR. In Fig. 5.8 one can see the first person views in the real warehouse from the HoloLens as well as in the VR warehouse. These experiments were conducted to test the validity of using a VR environment for larger scale tests.

In Fig. 5.9 one can see a selection of nine key moments of the real world experiment. The same experiment was conducted in VR, except it ends on step (g) as the worker does not have to take of the HoloLens and walk back to the entrance but merely stops the simulation there. The resulting probability outputs of the HIE algorithm, for both the real world and VR experiment, are shown in Fig. 5.10 with the key moments (visualized in Fig. 5.9) marked on the graph.

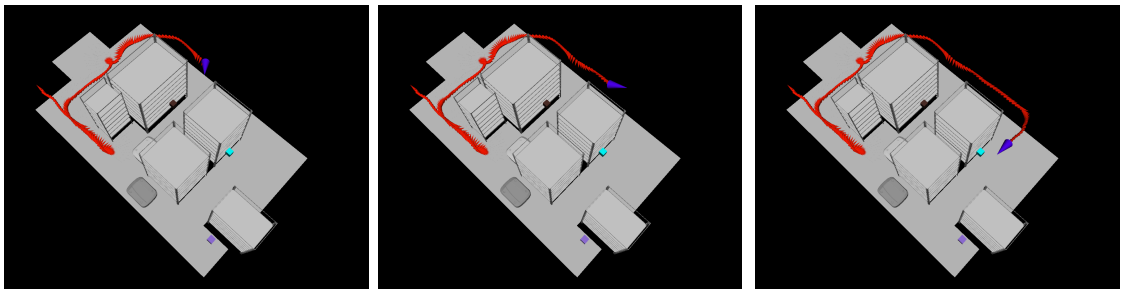
Let us go through the key moments. Step (a) marks the beginning of the experiment. The user then moves directly towards the purple goal, with the HIE estimation giving a high probability that is the user's goal. However, a mobile robot obstructs the path in step (b). As the user didn't immediately react to the obstruction and turned around the HIE marks the user as irrational (as the user's action do not show intention to reach any of the goals). The worker turns around to an alternate path with all three goals being likely (step (c)). The HIE therefore correctly outputs that the exact goal is unknown. At step (d) the worker will either turn towards the brown goal or continue towards the cyan and purple ones. By step (e) it is clear that the worker will not turn towards the brown goal, and the probability for that goal quickly plummets. At step (f) the worker is near the cyan goal, but as the robot blocks the only other way to the purple, the HIE can still not say that cyan is the user's goal. The worker starts turning around in place. The algorithm starts estimating



(a) Initial position.

(b) The worker moves towards the purple goal, but the mobile robot obstructs the path.

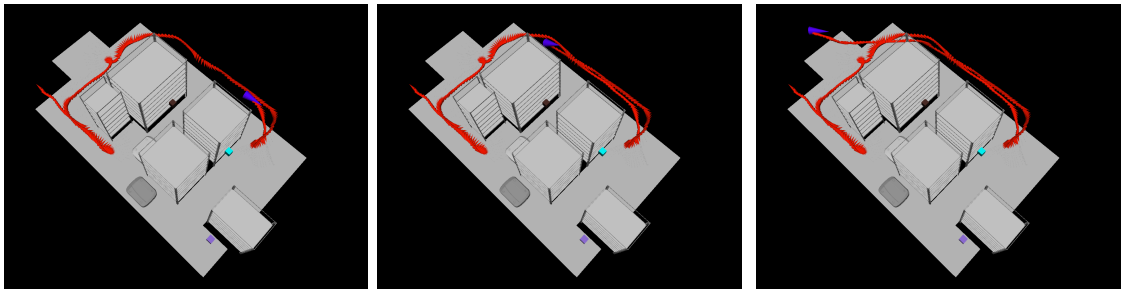
(c) The worker turns and follows the path consistent with going to all three goals.



(d) The worker is on a crossroad. If it turns towards the brown goal it is obvious that it is the goal wants to reach.

(e) The worker does not advance towards the brown goal and continues towards the cyan and purple goals.

(f) The worker stopped near the cyan goal and started turning around in place.



(g) The worker has taken the HoloLens off and is moving towards the brown goal.

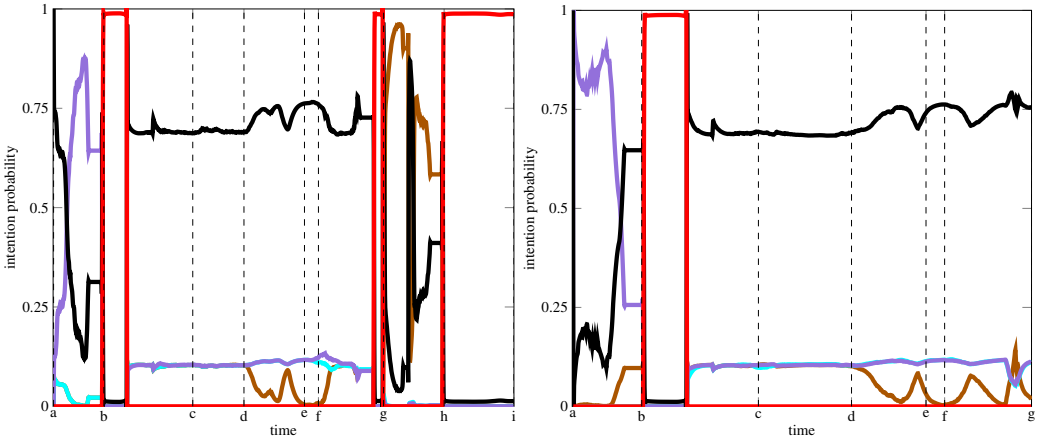
(h) Because the worker passed brown goal and is moving backwards, the model estimates its behavior to be irrational.

(i) End of the experiment.

Figure 5.9: Key moments of the real world experiment in the small-scale laboratory warehouse visualized in RViz.

the worker as irrational as they take off the HoloLens. They start walking back and the estimate for the brown goal rises. At step (h) the worker passed all the goals and the algorithm ones again estimates the user as irrational. The experiment ends at step (i) as the worker exists the warehouse.

One can see that the only noticeable difference between the real-world and VR experiments is at the start. This is attributed that in the real-world experiment the user needs to start the program while wearing the HoloLens and wait for the program to finish loading. This gives a bit of time before the user starts moving. In the VR experiment the user starts



(a) Algorithm's output in real world AR experiment.

(b) Algorithm's output in VR experiment.

Figure 5.10: Comparison of the HIE output between the real-world and the VR experiment. Intention estimations for the three goal locations are labeled with respect to their color in Fig. 5.9 (brown, cyan and purple), the unknown goal state is labeled black and the irrational worker state is labeled red. One can see that the results are similar for both setups.

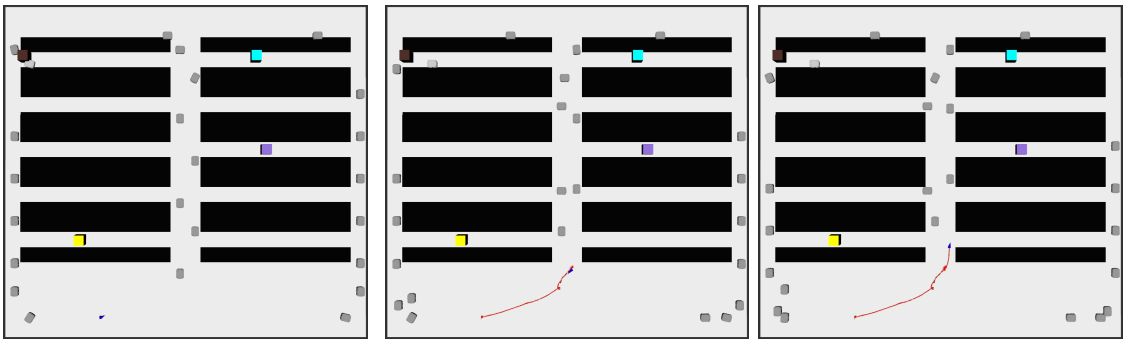
moving immediately towards the purple goal. However the start is semantically equivalent and the rest of the experiment proves that VR is a good surrogate for larger scale experiments.

Next we will present the results from a large scale VR experiment with 24 robots with preset goals and planned paths. We will go through six key moments shown in Fig. 5.11. The resulting goal probabilities with the key moments marked are visible in Fig. 5.12. In step (a) the experiment starts. The worker starts moving towards the central isle but robots block their path (b). As the worker is not proceeding towards any goal in this time-frame, and stands and turns randomly the HIE declares them irrational. The worker can access the yellow goal and, as they once again stand in place and look around, the estimate switches between the yellow goal and unknown goal (c) and (d). The worker starts moving as the path is now clear. The probability for the yellow goal drops and the probabilities for the cyan and purple goals rise as the worker moves closer towards them (e). As the worker turns toward the purple goal and starts moving towards it the probability for purple rises. The worker reaches the purple goal and the experiment ends.

For the tests with the industrial manipulator, we predefined three spatial goals - a green cylinder, a red cube and a blue sphere. They are placed on a table reachable by both an industrial robot and by the human coworker. The first-person view of the setup is visible in Fig. 5.13. The green cylinder is located to the left of the image over the left-most box partly visible in the picture.

The first step was to experimentally test the optimal values of the HMM parameters which were determined as $\alpha = 0.3$, $\beta = 0.05$, $\gamma = 0.05$, $\delta = 0.1$. Please refer to [137] for the argumentation regarding the selection of parameters for this scenario.

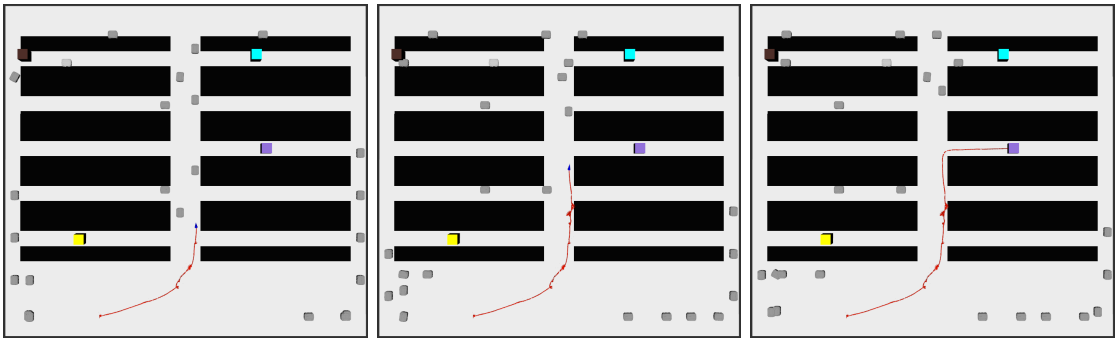
We predetermined three sequences of goals. The first one is going from left to right, trying to grab the cylinder, then moving towards the red cube and then to the blue sphere. The second sequence is cube-cylinder-cube-sphere. The third sequence starts with the



(a) Initial position.

(b) The worker moves towards the yellow goal but a mobile robot blocks the path towards it.

(c) Only the path towards the yellow goal is unobstructed but the worker does not go directly towards it.



(d) The mobile robot blocks the worker's advancement. The worker turns and moves while waiting for the robot to move.

(e) Mobile robot moved and the worker is going towards the purple goal.

(f) The worker is at the purple goal's location which ends the experiment.

Figure 5.11: Key moments of theVR scenario of a larger warehouse with 24 mobile robots, as visualized in RViz. Goal nodes are labeled with colored cubes.

sphere, cube then cylinder, then the user turns around until they turn again to the sphere. The results are visible in Fig. 5.14. One can see that the outputs are as we would expect them to be. When the user turns around completely they are declared irrational as we would expect since no motion towards any goal is detected. When the user is in-between goals the unknown state estimation rises as would be expected, but also the estimate for the neighbouring object. In Fig. 5.14(a) one can see a small time interval where hand tracking was lost, prompting the algorithm to switch to estimating the unknown/irrational states. In Fig. 5.14(b) it can be seen that the transition from cylinder to cube lasts slightly longer than back from cube to cylinder, due to the fact that the algorithm is reluctant to estimate an already visited or skipped goal. One can also see the long transition between the cube and the sphere, as the algorithm prefers the goal that has already been visited two times. This shows that the estimation follows our intuition.

Additionally, we tested simple interactions between an industrial manipulator and the human user. In the first one the robot was selecting goals randomly. Should the goal that the HIE algorithm estimates be the same goal the robot is moving to, it would stop and select a new goal. Additionally the HIE can be used remotely to select goals the robot should move to. This indicates that the HIE may also be useful for telemanipulation.

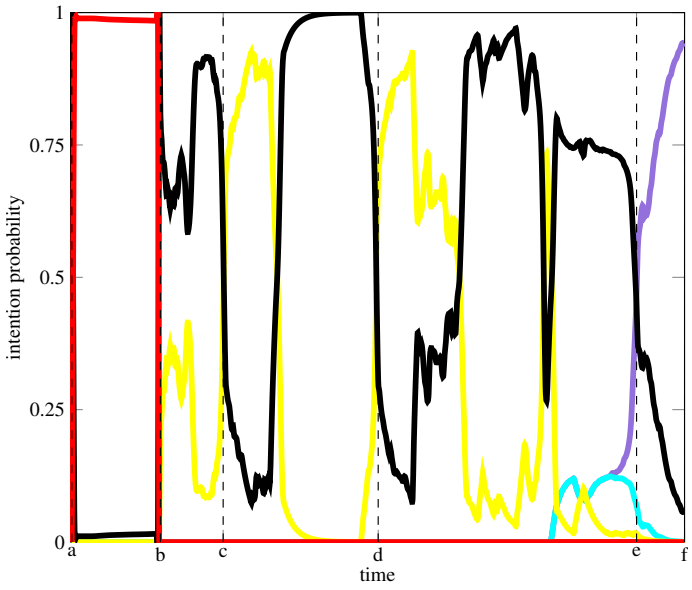


Figure 5.12: HIE output for the VR scenario with 24 mobile robots. Intention estimations for goal locations are labeled with respect to their color in Fig. 5.11, the unknown goal state is labeled black and the irrational worker state is labeled red.

Although the tested environment was very simple, the goal of the test was mainly to demonstrate that the HIE algorithm can be extended to HRI with an industrial robot in addition to its intended use case in automated warehouses. If the environment is more complex with obstacles preventing the human coworker from trivially reaching obstacles, it could be argued that the shared workspace is not properly ergonomically setup. However, even in those cases, one may construct a 3D Vornoi diagram analogously to the original 2D diagram presented before instead of using the simplified straight-line Euclidean distance.

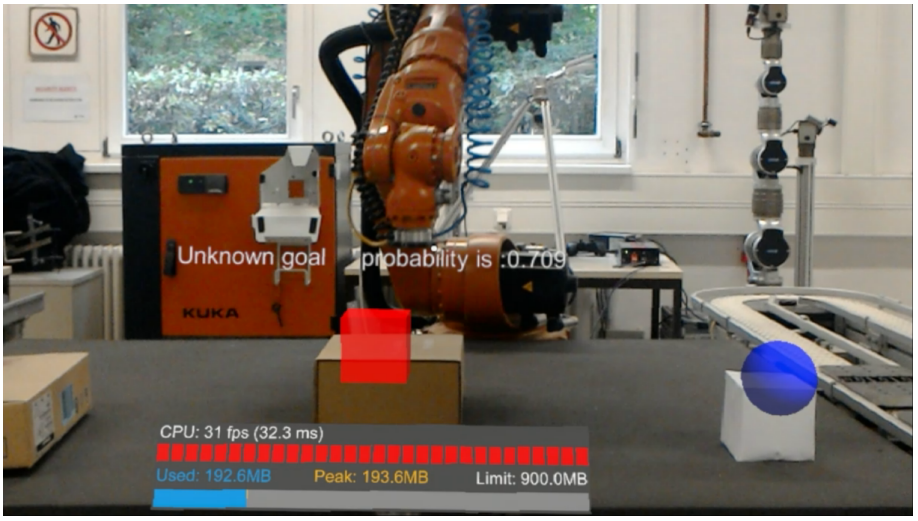
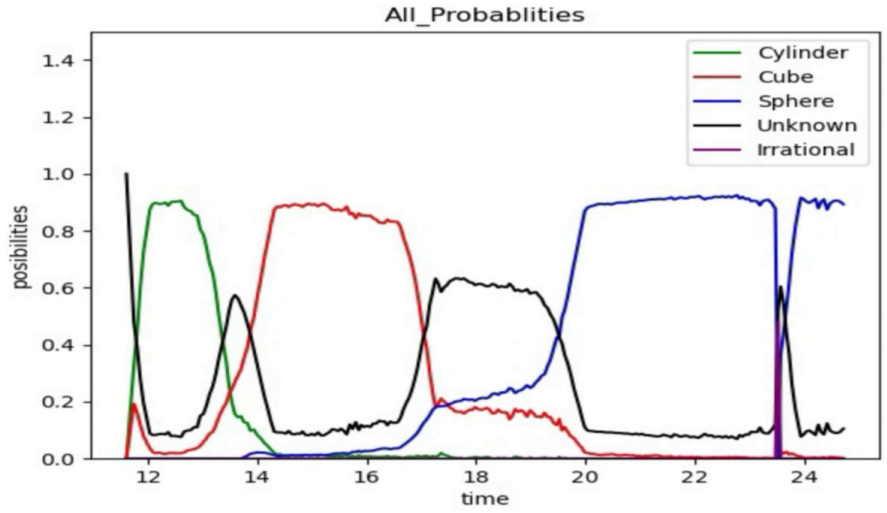
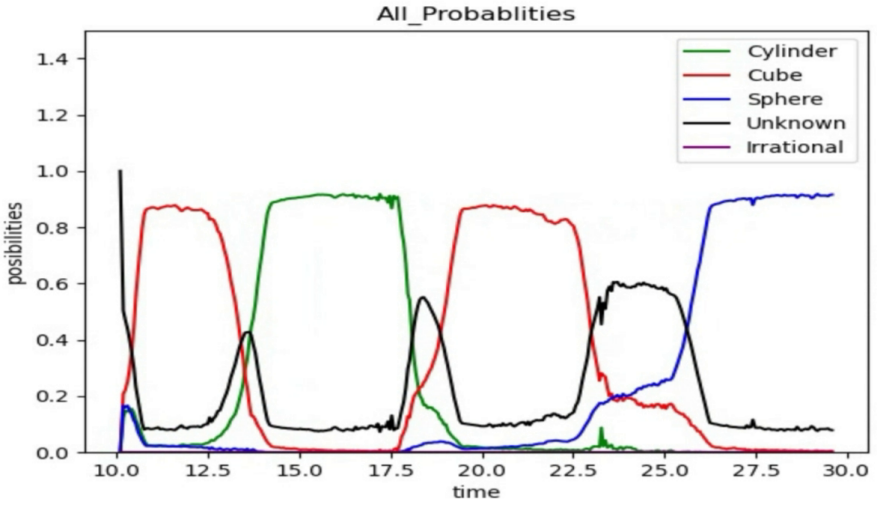


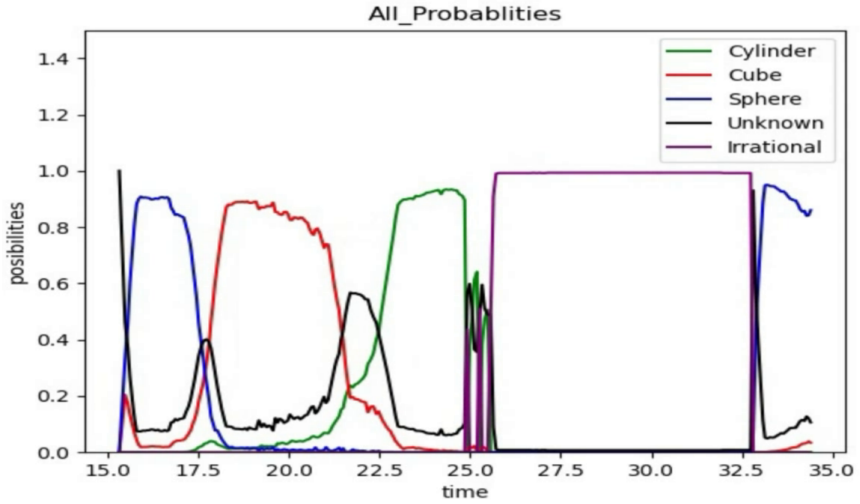
Figure 5.13: The setup for the HIE test with an industrial robot. The goals are predefined as a green cylinder (not visible, left of the image), a red cube and a blue sphere.



(a) Result for the sequence cylinder-cube-sphere



(b) Result for the sequence cube-cylinder-cube-sphere



(c) Result for the sequence sphere-cube-cylinder then turn in place and come back to sphere.

Figure 5.14: Three tests of the HIE algorithm adapted for the industrial manipulator setup

5.3 Conclusion

In this chapter we presented two important tools in keeping the human coworker safe while interacting with industrial robots. The first is a method for inside-out tracking of the human inside the robot cell. The human is protected by a virtual barrier composed of 5 cylinders - one for the head and the body and two for each hand, one for the upper arm and one for the forearm and wrist. We made use of wrist worn IMUs to track the human's arms. The main disadvantage of IMUs is that they suffer from drift, that is, error accumulation over time. For this reason the IMUs where paired with the HoloLens. The HoloLens possesses both hand recognition and localisation via SLAM which does not suffer from drift. This allowed the IMUs' error to be bound and prevented drift.

The other tool is the detection of human intentions. The HIE algorithm was developed by SafeLog project partner Tomislav Petković and used in an autonomous warehouse scenario. The algorithm was also adapted to function in a collaborative setting featuring an industrial robot. The human intention estimation not only provides additional safety, as the robot knows beforehand the actions of the human, but also makes the interaction more intuitive and efficient, as the robot planning can be adapted in advanced, knowing the intention of the human.

During the proposed human tracking approach, we used a very simple left-right hand differentiation that may produce erroneous results if hands cross. Likewise the HoloLens is only able to track the hands of the user if the user makes the *"Hold"* gesture. Finally the hand tracking of the HoloLens provides only position, and not orientation, meaning that the orientation component is not corrected. However the orientation estimation itself is prone to less drift. This is due to the fact that IMUs measure angular velocity, meaning a single integration or summation produces orientation, which accumulates errors significantly slower than the double integration or summation needed to calculate position from linear acceleration. Secondly the orientation estimation can benefit from the magnetometer readings, reducing the drift further.

All of these issues can be alleviated by using more advanced tracking methods which track the entire skeleton of the hand, like the one proposed in [97], or newer devices such as the HoloLens 2 which feature inbuilt hand skeleton tracking.

There are also many other sensor fusion and filtering methods, as presented in [138], that may be more precise and efficient. Improving the sensor fusion and conducting further tests is an important factor to further develop this method in the future.

On the HIE side, using an HMD with eye tracking sensors and hand skeleton tracking would massively increase the landscape of intentions that could be recognized, while increasing precision. Furthermore the position of the objects (goals) could be generated online instead of being predefined by using an object detection algorithm. We have tested the YOLOv4 network [139] for that purpose. The preliminary tests will be described in the next section. Integrating object detection into the intention estimation framework is an important point in further developing this system.

The part of the system pipeline responsible for tracking the human and estimating their intention is visible in Fig. 5.15. To the left are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS.

In regards to published worked, the inside-out tracking of the human was presented in [110]. In [32] Petković presented the HIE algorithm based on hidden Markov models. We

provided the sensor data from the HoloLens and conducted the tests in a real automated warehouse and in one simulated in VR. The HIE algorithm was slightly modified and reused in [137], where it was applied to detect the intentions in a collaborative setting between a human and an industrial robot.

This chapter concluded the main components of the end-to-end HRC system. Throughout the previous chapters we have seen how to map the environment, reference objects inside the environment, setup the workspace of a new robot, program it, and finally interact with it, all through the use of a completely mobile HMD device. The next chapter will present the envisioned system workflow, communication between devices and the software components of this system.

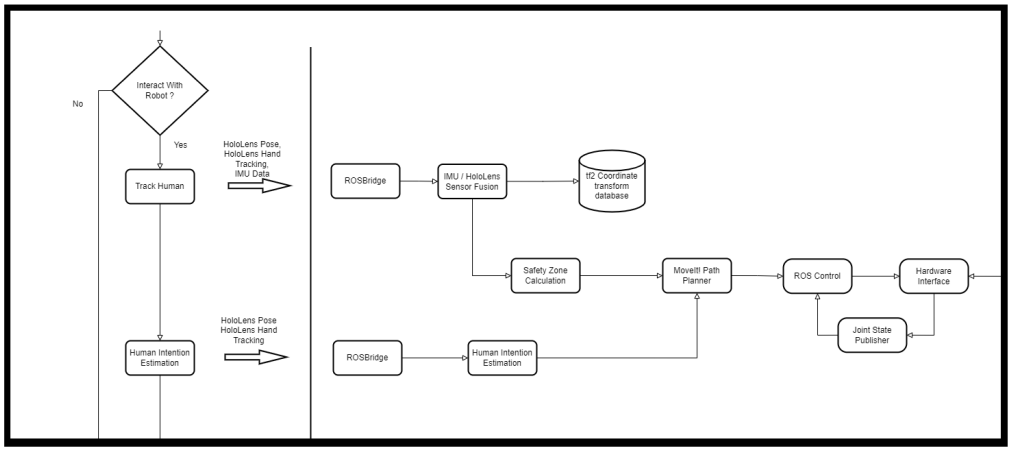


Figure 5.15: The interaction and tracking components of the system pipeline. To the left of the line are the components running on the HoloLens HMD, while right of the line are the components running on a PC with the ROS.

6 System Overview

In the previous three chapters we have introduced the various components necessary for implementing a completely wearable HRI system. First, in Chapter 3, we introduced different ways to use the HoloLens to construct a map of the environment. Based on this map we proposed methods to locate the industrial robot in the map and calculate the coordinate transform between the robot and the HoloLens coordinate systems. In Chapter 4, we first presented how to convert the map of the environment into a representation of obstacles and how to add no-go zones. This is important for planning collision-free trajectories. Next we presented three different methods for programming industrial robots using the HoloLens - via Holographic waypoints, via generalized hand guidance, and via inside-out tracking of smart tools. Finally, in Chapter 5, a method for inside-out tracking of human coworkers and establishing a safety zone around them has been proposed. A human intention estimation algorithm has also been proposed to supplement safety as well as making the interaction between robots and humans more intuitive.

This chapter will act as a practical synthesis and show how the different components work together to form an integrated HRI system. The system is distributed over two devices, the Microsoft HoloLens and a desktop computer. When interacting with multiple robots, it is assumed each robot has its own connected computer, with the appropriate urdf, 3D models, hardware interfaces etc. The software components themselves on the computer side, however, would remain the same.

An important consideration is the limited processing power of the HoloLens. The program running on the HoloLens should always run at a minimum of 60 FPS, otherwise we risk the user getting dizzy or nauseous [140]. Therefore we offload as much processing as possible to the desktop computer, keeping the visualization and interaction components on the device as well as some data preprocessing.

6.1 Desktop Computer

For implementing the system on the computer side we used the Melodic distribution of ROS. In ROS, different programs and scripts are abstracted as nodes. Nodes communicate with each other via topics (messages that are broadcasts and can be subscribed to by different nodes), services (functions that nodes may offer and other nodes may request) and actions, which were not used in this thesis.

The computer communicates with the robots using KUKA's Robot Sensor Interface (RSI) interface. In [141] the communication over RSI and the KVP server were compared. It was found that there is a 120 ms lag in the RSI compared with the 50ms for the proposed approach. using the KVP. Still, in most cases, the RSI interface should suffice. The hardware interface and joint state publisher nodes take care of the communication with the robot and convert the RSI messages to topics and vice versa. The ROSBridge node

takes care of converting the incoming WebSocket JavaScript Object Notation (JSON) messages from the HoloLens to topics and services.

The PCL library was used for working with point clouds. A ROS wrapper was used to implement the programs as ROS nodes. For working with 2D images OpenCV was used in the same way.

ROS nodes can be written either in C++ or Python, while both OpenCV and PCL are written in C++. Therefore the majority of nodes were written using C++, with a few taken from student projects (like the hand tracking using the IMUs) are written in Python.

For MoveIt a small setup step is needed to generate the robot configuration files. This is done through a setup GUI provided by the MoveIt package itself.

The urdf and the robot meshes are assumed to be available. There are two sets of meshes, the collision and visual ones. The collision meshes can be simplified for faster collision checking, but they can also be the same as the visual meshes.

The tf2 node is used as a storage of coordinate system transforms in the system. The OctoMap can be saved and read from a .bt file.

6.2 HoloLens

The HoloLens Programs are made using the Unity SDK and written in C#. The MRTK toolkit, developed by Microsoft to facilitate developing programs for the HoloLens and other MR headsets, was also extensively used due to its collection of ready-to-use holograms (game objects) and scripts for interacting with the HoloLens itself. The ROS# package developed by Siemens provides useful tools for importing urdfs and communication with ROS over ROSBridge using JSON messages.

For reading and preprocessing the raw sensor data, a co-program was written in C++ with the HoloLensForCV libraries. HoloLensForCV is based on open CV and designed to easily interact with the sensor onboard the HoloLens. In the future, this co-program should be directly integrated into the Unity project itself, so that the user needs to start a single program.

As previously mentioned, the communication is based on ROS#, but there are other alternatives. In a recent paper Allspaw *et al.* [142] benchmarked different Unity to ROS communication implementations. It was found that ROS# has the worst performance. Therefore in the future the communication interface should be reconsidered to be based either on ROS.NET developed by the University of Michigan, or the ROS-TCP-Connector made by Unity itself.

6.3 Pipeline

The program pipeline can be seen in Fig. 6.1. First, using the HoloLens UI the IP address of the computer connected to the desired robot can be entered and a connection established (Fig. 6.2). Once the connection is established the urdf and the visual and collision meshes are communicated to the HoloLens to be visualized. The user then walks around the

robot workspace to map it. Here the spatial mesh that the HoloLens builds by default is visualized, while depth sensor data are streamed to the computer and used to build a point cloud of the environment. The mesh was used for visualization as it requires less processing due to it being readily available from the HoloLens' memory itself and acts as a useful proxy for the area mapped.

Once the user is satisfied with the mapped region, they position the seed hologram near the robot base. With the voice command *"Send Mesh"* the mapping step is finalized and the robot registered. Its points are removed from the point cloud and and OctoMap representation of the environment created. Using the voice command *"Get Map"* the OctoMap can be visualized through the HoloLens. The individual voxels can then be moved and removed. Using the voice command *"Add Obstacle"* virtual obstacles representing no-go zones for the robot manipulator may be added. Finally, with the voice command *"Return Map"* the edited OctoMap including the virtual obstacles are sent back to the computer for post processing and the creation of the final OctoMap.

The robot can now be programmed using the three different methods described in Chapter 4. If one wishes to use the waypoint method (see Sec. 4.2.2.1), they can add new waypoints via the *"Add waypoint"* voice command. When all the waypoints are placed the user invokes the planning through the *"Plan Path"* voice command. The poses of the waypoints are sent to ROS where MoveIt is used to plan the trajectory. Once the planning is finished the trajectory is sent back to the HoloLens where the Holographic robot executes the path. Then, if the user is satisfied with the trajectory, they can execute it with the real robot using the voice command *"Execute path"*.

When using the second method, hand guidance (see Sec. 4.2.2.2), the user just approaches the robot and uses the drag gesture to push and pull different robot links. Using a floating menu the user may record the motions of the robot by pressing the *"Record*" button. The recording can also be paused or deleted. When the robot teaching is finished the user can use the *"Replay"* button to execute the taught trajectory.

The third and final method is tracking of external tools (see Sec. 4.2.2.3). For this method the user needs a tool outfitted with IR markers, as well as a text file describing the positions of the various markers in the marker coordinate frame. The user may use the tool to draw the path of the robot. This path could be sampled and a planning request may be sent to MoveIt analogous to the waypoint method.

Once the programming is completed the human is tracked inside the robot working area while the robot executes its task. If the task is collaborative, HIE may be used to better plan for robot trajectories, as described in Sec. 5.2.3.

6.4 Discussion and Future Work

The presented system pipeline is conceptual, as not all the system components were integrated into the proposed pipeline. That said, several overlapping pipelines were implemented. For example, the user could map the environment, reference the robot, create the Octomap, and use MoveIt for path planning with only only two programs, one on the HoloLens and one in ROS. The MoveIt path planning however was done through the RViz package with the MoveIt plugin.

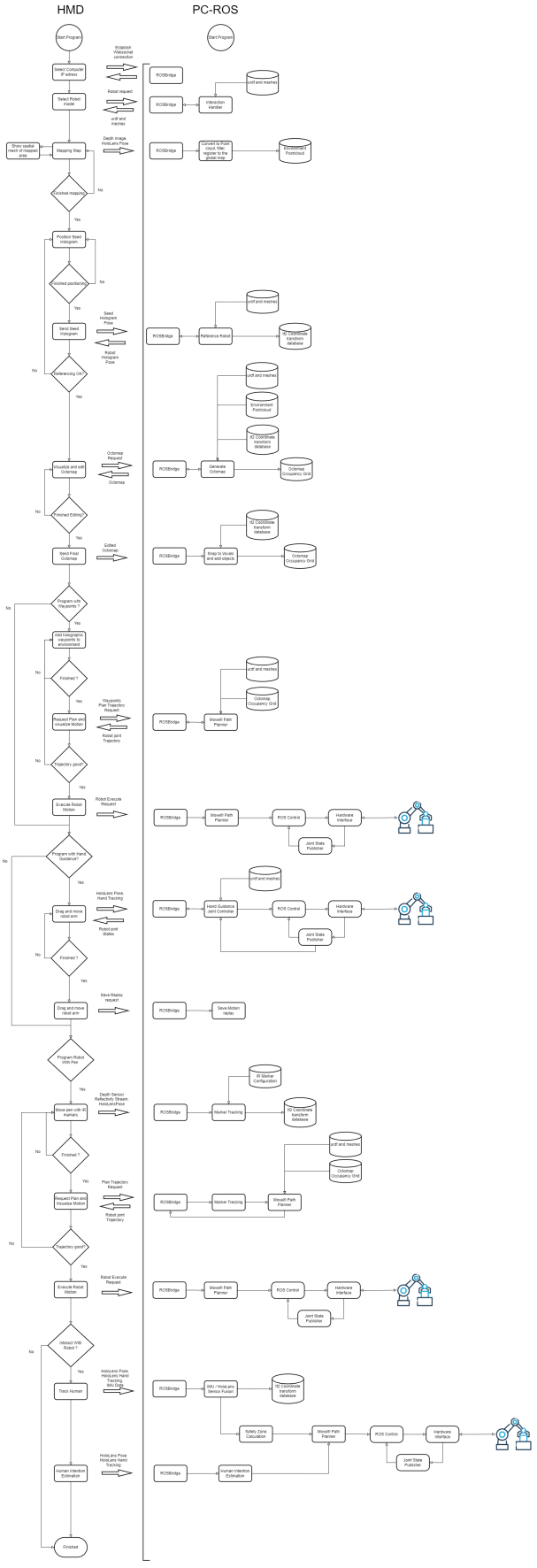


Figure 6.1: The proposed system workflow and communication

Figure 6.2: The UI menu to select the IP of the desired computer

The robot programming through waypoints featured a mapping and referencing step, as well as direct communication with MoveIt to plan, visualize and execute the trajectory on the real robot, however the OctoMap representation was not used in this this case. Similarly, the hand guidance featured mapping and referencing as well as direct control of the industrial robot through communication with ROS Control. Again the Octomap wasn't integrated.

The tool tracking method was not combined with the rest of the framework including the mapping and referencing. It also was not used to directly program a robot. It merely served as a demonstration of how a wide variety of robot programming methods can be integrated into a system based on HMDs. The integration would be easy enough, as proceed as described previously. The trajectory of the tool would be sampled and a planning request sent to MoveIt.

The tracking of the human was not tested with the safety cylinders nor with a moving robot slowing or stopping based on the distance of the user, Although it was connected to the referencing step and the position of the head and hands of the user were saved and updated in the tf2 node.

Finally, the HIE was not integrated into the previous pipeline. It was tested with different scenarios as described in Sec. 5.2.3 including with a moving robot, and integrated into ROS, but not the rest of the pipeline.

Large scale user tests would also be beneficial, though numerous papers have already proved the benefits of using AR in HRI, especially in the field of programming [105]. Still, some components such as the OctoMap workspace setup, are novel to our approach, and further user feedback would be extremely useful.

To make this research commercially and practically feasible, the first step would be to integrate all the parts into a single framework, with the user launching one program on the HoloLens and one on the computer connected to the robot. The UI should also be greatly improved and made streamlined. The use of the HoloLens 2 would ease the interaction where gestures are concerned, as it possesses full hand skeleton tracking. The skeleton tracking and the eye tracking would also be very beneficial in expanding the HIE capabilities. Further improvements would be integrating faster and more efficient communication between the HMD and the computer, as well as the communication between the computer and the robot.

Another open question is how to improve the referencing and tracking precision, and test the cases where the localisation precision of the HoloLens presents a bottleneck in the HRI scenario. Identifying failure cases and mitigation strategy would help elucidate the minimum needed requirements of any future HMD if it is to be used for HRI.

The HIE in particular could actually be used in all of the components, as it could also improve the HMI to the HMD and the presented system is general.

A more long term goal would be perhaps to focus less on an established interaction pipeline and more on using the components as a set of tools that the user can use to program their own HRI. This would mean that the user is able to use the HMD and AR to author their own HRI programs and pipelines. This was proposed as future research opportunity by Suzuki *et al.* in their extensive survey on AR-based HRI [11]. According to the survey there were only two research papers about the subject. The first one by Suzuki*et al.* uses AR and a Swarm UI - a swarm of robots as a user interface, to program other interactions for the Swarm UI [143]. The other, more closely related, is the GhostAR by Cao *et al.* [144]. The framework utilizes an AR "ghosts" of the user to synchronize the interactions between robots and humans. A collaboration model was developed which takes the real-time captured motion as inputs, maps it to the previously authored human actions, and outputs the corresponding robot actions to achieve adaptive collaboration.

6.5 Conclusion

This chapter showed the specific implementation of the components presented in the previous chapters and puts them in the context of program pipeline. Although the entire pipeline itself was out of the scope of this work, we have demonstrated how different components have been integrated into their own overlapping sub-pipelines which further demonstrates that the implementation of the proposed pipeline is feasible. We also presented future system improvements to make the system more practical.

In the next chapter we will present other interesting applications of HMDs that are not directly connected to the envisioned HRC pipeline, but may yet be useful in the broader scope of HRC. These are namely moving object tracking, HMDs as a data source for LfD, and HRI design.

7 Summary and Outlook

We have reached the final chapter of this work. Through the previous chapters we presented algorithms and methods to implement a full HRI system for HMDs.

In 2008, Bill Gates predicted that by 2025 there will be a robot in every home. Though that might be the case for robots like the Roomba, freely programmable robots are still far away. In industry, collaborative robots are becoming increasingly popular, with easier lead-through programming and safety for human coworkers. In 2023, Intrinsic, an alphabet company, launched their beta version of the Flowstate. Flowstate is a workflow-based programming environment that uses programming blocks. It also features an integrated simulator. Such a concept however is not knew, and it remains to be seen if such a system truly enables lay-users to program real industrial robots. Still it shows a drive to bring robots to small enterprises and allow users to program and interact with robots without any training in robotics.

This thesis proposes a paradigm shift, where robot programming takes place in-situ, and is mediated by HMDs, wearables and AR. This makes robot setup and programming more transparent and intuitive than alternative methods. Smart devices that support AR interaction, and which possess the sensors needed for the algorithms proposed here, are already widely available. Although HMDs are still not widely used, they are becoming ever more common. Judging from the current trends it is not far-fetched to assume that HMDs will become a very common device in the electronics' consumer market.

Outsourcing sensors and interaction modalities to smart devices makes robot cells cheaper and less complex. It also allows a single user to interact with several robots, without the need for every single robot cell possessing the sensors needed for intuitive programming and HRI. This makes production lines that utilize robots cheaper.

To achieve functional safety, the system can be extended with a wearable safety vest based on UWB ranging.

This final chapter will present a short summary of this work. We will discuss the results and the issues faced as well as possible improvements. We will then attempt to draw conclusions and suggestions on how to generally implement HRI systems using HMDs. Finally we will present future directions of research.

7.1 Summary

In the first chapter, **Chapter 1**, we outlined the main motivation of this work, namely to make robots more intuitive to program and interact with, without increasing the complexity and cost of industrial robot cells. We also outlined the contributions. Shortly, the main ones being developing all the components for an end-to-end system. Although

programming industrial robots via HMDs has been a very popular research topic, to the best of the author's knowledge, no holistic system such as the one here has ever been proposed. Parallel to this, some components and methods are completely novel. Particularly the mapping and setup of the robot's working environment, the general hand-guidance framework and the HMD-based HIE.

Chapter 2 introduced the concept of AR, mixing digital content into the real world which is interactable in real-time and registered to real world objects, and how AR is applied to the field of robotics. We then went through the possible hardware implementations of the AR, depending on the system mobility and display technology. We motivated the selection of an HMD as the device of choice for implementing the vision of this work. We then describe in depth the Microsoft HoloLens, our HMD of choice, it's hardware and capabilities. We shortly introduce other wearables - digital devices possessing sensors and connectivity that are worn on the human user, and how they are applied to HRI. Finally we present SDKs, libraries and programs that are universally useful for developing AR-based robotic applications.

In **Chapter 3** we described the first basic components of the system, the mapping and the referencing. We map the environment by using the raw depth sensor data of the HoloLens. We convert the depth data frames to a point cloud and register it to the rest of the map. We discard points closer than 1m to eliminate interference from the user's hands, and point further than 3.3m as they become too noisy. ICP registration between the point cloud frames did not contribute to the overall precision. We filter the point cloud with a voxel grid filter to maintain a uniform point density. We than apply an outlier removal filter, removing any points with less than 9 neighbours sin a 5cm radius. A post-processing step consisting of MLS smoothing and a RANSAC plane detection, where all points in the vicinity of the plane are mapped to the plane itself was implemented and showed to produce better results. Comparing the point cloud of the IPR robot hall generated with this method and one taken with a FARO laser scanner with a 1mm precision showed that 75% of the points lie within a 3.6 cm distance from the laser scanner baseline. Three referencing methods were proposed. We chose the method where the user places a seed hologram as a first guess and then an ICP registration step improves the guess. Even though the RMS distance between the referenced robot point cloud and the environment point cloud is around 5mm for solid user guesses, other tests suggest that the error between the robot hologram and the real world robot are slightly above 2cm. This is due in large part to the HoloLens localization error which, through several different tests, was found to be around 2cm.

Chapter 4 presented a method to use the mapping and referencing to represent the workspace of an industrial robot. We used an OctoMap environment representation. Firstly a k-d tree of the point cloud is constructed, speeding up nearest-neighbour searches. The robot points are removed based on the closeness of environment points and the points of the referenced robot model. All the points outside the reach of the robot are also removed to speed up calculation. The remaining point cloud is filtered with a voxel grid filter with the voxels being the same size as the voxels of the OctoMap. Finally a binary OctoMap is created from the point cloud. The OctoMap is overlaid over the real-world workspace on the HoloLens. Here the user can edit the OctoMap and add no-go zones to the robot. Once the user finished editing the OctoMap, the new OctoMap is sent back to the ROS computer, where the edited components are snapped back to the voxel grid. This OctoMap can now be used for collision-free planning using MoveIt, Numerous tests have

shown almost no collisions, with a few edge-cases solved by padding the OctoMap with voxels of half the size as the original ones. Three different methods to program robots via HMDs have also been presented. The first consists of placing holographic waypoints, whose position in the robot coordinate system, available after referencing, is sent to the MoveIt to generate a trajectory. This method is similar to others proposed in different works, with the main difference being that a proper path planner is used which can calculate collision- free trajectories. The general hand guidance uses the developed referencing algorithm and the in-built hand tracking capabilities of the HoloLens HMD to emulate the hand guidance capabilities of CoBots on any industrial robot. No additional sensors are required. It was shown that the algorithm is able to successfully emulate hand guidance, with the user pushing and pulling any link of the robot. The third method used the IR reflectivity stream of the HoloLens for inside-out tracking of smart tools equipped with IR markers. Such tools are quite popular for intuitive robot programming yet require an external marker tracking system to work. We tested the accuracy of the tool tracking using an ART-3 marker tracking system. Our tracking method achieved a static position error of 1.9 mm and angular value of 0.37°. In the dynamic case, where the HoloLens itself moves significantly the positional and angular errors were 22.1mm and 3.87° respectively. The large dynamic error may once again be attributed to the HoloLens localization error.

Inside-out tracking and the intention estimation of the human coworker is introduced in **Chapter 5**. The tracking was done using the HoloLens HMD and two IMUs on the wrist. The main concept is that the IMUs allow tracking of the arms of the user while not in the field of view of the HoloLens, making it possible to construct a virtual safety barrier around the user consisting of five cylinders - one around the head and the body anchored to the localisation of the HoloLens, and two for each upper arm and forearm calculated using the data of the IMUs. This would allow for safe HRI while presenting more opportunity for interaction. The main issue with IMUs is that they accumulate drift, meaning that the localization error continues to increase over time. This can be solved using the HoloLens' in-built hand detection and localisation, which does not accumulate drift. As the hand moves into the field of view of the HoloLens' sensors, the drift can be bound by fusing the IMU and HoloLens hand tracking data. Two different filtering and sensor fusion strategies have been tested - a Kalman filter approach and a combination of low and high-pass filters. The latter proved superior in tests. Tests with the HoloLens showed that this method indeed bounds the drift of the IMUs. A method to calculate the arm parameters and joint states was also presented, although no safety tests with the robot were made. The intention estimation was based on the framework developed by Petković. The intention estimator is based on a HMM, with states corresponding to each goal, one for the unknown state (when the system cannot decide about the exact goal) and one for irrational worker (when the system detects the user is not following any of the goals). In the original collaboration paper, the HoloLens localisation was used in an autonomous warehouse and it was shown that the system is able to deal with moving obstacles, AGVs in this case, and with adding or removing goals online. It was also shown that the estimator follows human intuition about which goal the user is moving to, both in a small scale test in a real test warehouse with the HoloLens, and in larger scale tests in VR. The work was extended to not only use the pose of the HoloLens, but also the tracking of the hands to determine which object the user wants to grab or interact with in a collaborative task with an industrial robot. The original system was used with minor modification to include hand movement and some minor changes to the parameters. It was shown that the original framework is capable of handling this scenario as well.

In **Chapter 6** all the components were brought together into a holistic system pipeline. We described the implementation of the systems on the two main components, the HoloLens and a desktop computer running ROS. The pipeline was described and it was shown how such a pipeline could support the user all the way from setting up the robot working environment to programming it to interacting with it in a safe way. Although the entire pipeline was not implemented, several overlaying pipelines were. That shows that constructing such a pipeline is feasible. Possible improvements to the pipeline were also discussed.

7.2 Discussion

The system presented in this work represents the first endeavour to develop an end-to-end interaction system to make using robots easier and more intuitive. Though we have developed this system for the Microsoft HoloLens, thought has been put into making the algorithms as device agnostic as possible. In some cases, such as for mapping and referencing, we proposed several methods depending on what could be the possible sensors or accessible data. In the end, however, the applicability on other devices relies on the device developer themselves and whether they allow access to sensor or relevant data or not.

The second important point to address is the lack of user studies. There are several reasons for that. Throughout this paper we have referenced other authors and their research containing user studies that prove AR-based HRI improves the user experience and in almost all cases lowers the mental load. The main issue with most user studies in robotics is usually their small sample size, though with a large number of paper the overall sample size becomes adequate for the conclusions presented. As this is a large pipeline with many components, any user study of the entire system would be long and necessitate a large number of participants to draw proper conclusions. Second, it is unclear how to obtain a baseline. Should the users try to setup the robot cell and program the robot arm for a task using classical industrial methods? These usually take a long time and require such expert knowledge that the average person or even robotics student does not posses. Perhaps an appropriate measure would be to take only lay users that have never worked with a robot and have them setup the robot cell and program a simple task. Here again we run into the issue of sample size and lack of a baseline or alternative approach. Another issue we face is that the system would require quite a bit of work to develop a proper, easy to understand UI. A bad UI would skew the results no matter how good and efficient the actual system is.

Thus possibly the most important point to further develop the system is to construct a user-friendly UI over the entire pipeline. This would then allow large scale user tests both with professionals and lay users. It is our opinion that such tests would greatly benefit not only the entire system, but also specific components and allow the research community to obtain practical best practices for implementing their AR-based HRI algorithms.

Perhaps the largest and most influential bottleneck of the system was the HoloLens' localization error of around 2cm. The localization error is dependent on several factors, including the amount of features in the environment, lighting, the motions performed etc. It was however found that both in an autonomous warehouse setting and in the robot hall,

the error should be expected to be slightly higher than 2cm. As it was seen this error influences the referencing precision, meaning the error propagates through the other steps, as well as the quality of the environmental map and the precision of the tool tracking, which don't rely directly on referencing but do on the localization. In the mapping tests it was found that 75% of map points within 3.6 cm distance from ground truth taken with laser scanner. Again most of that error could reasonably be attributed to the error of localisation.

It is very hard to get a reliable assessment for the localisation error for most HMDs. The newer generation of HoloLens, according to the research by Soares *et al.* [145] has a similar localization error to HoloLens 1, though they compared the position of the hand, which probably introduced an additional error source. In [146] a HoloLens 2 was mounted on a KUKA arm in a monotonous environment to provide the baseline. Several metrics were calculated. The maximum error between the end and start points of a circular motion were found to be 49.3 mm. The maximum error between the predefined path and the closest point on the HoloLens 2 was measured to be 7.9mm. However when the traveled path was compared, the maximum error rose to 1220.1 mm. The featureless environment was selected so that the localization data relies mostly on the IMU which shows large drifts. In [147] the hologram stability was tested and the maximum error to be 6.2 mm while citing other work that went up to 9.6 mm. Naturally this is not a good measure of the localisation accuracy itself. It is safe to assume that the localization accuracy of the newer generation is comparable to the HoloLens 1. Other HMDs or custom localization algorithms may be needed to push this error to the millimeter range, which would be adequate for the proposed applications.

If the robot has precision sensors for completing its task, these may be relied upon to improve the final accuracy of the program itself like in the method proposed by Hartman *et al.* [98].

Switching to the HoloLens 2 however would have benefits such as hand skeleton tracking and eye tracking. Hand skeleton tracking in particular would make the hand guidance and the intention estimation algorithm much more user friendly, as it wouldn't rely on specific gestures to track. Eye tracking combined with the hand skeleton tracking could also improve and extent the intention estimation not only to which is the user's goal but also which action the user is intending to preform. Newer devices offer hand and eye tracking as well as better quality and form factor.

On the computer side, integrating ROS 2 would provide real-time control capabilities. Although lack of real-time control didn't have any effect on the research presented here, it is foreseeable that it would be a beneficial feature to have in a practical system. Connection options between the computer and the robot should also be considered and improved to reduce lag. Finally, as already mentioned, using another framework for communication between the HMD and the computer could further reduce lag, as shown in the paper by Allspaw *et al.* [142].

Also to note for the practical usability of this system is the safety aspect. Although this system can be used out-of-the-box with collaborative robots, as they are already safety certified to work alongside humans, it would be impossible to do so with standard industrial arms. The main issue here being that the system proposed here, relying on consumer hardware, would not be able to comply with all the norms for a safety certification. For example we cannot guarantee real-time operation of the entire system. It would take great

effort to do so and probably additional custom hardware would be required. The more practical solution would be integrating a safety vest, such as the one developed by Končar for the SafeLog project, to provide functional safety. This vest relies on UWB ranging and safety computers and it was already demonstrated to be safety certifiable. The system would require slight modification to the robot cell in the form of several UWB transceivers mounted on the robot itself as well as a safety computer connected to the emergency stop of the robot. An additional benefit would be that the UWB ranging data could be used to improve the localisation accuracy of the HMD.

7.3 Outlook

When digital computers were first introduced, they were cumbersome and specialized equipment. Only with the advent of better HMIs did they become an ubiquitous technology. These HMIs included better input devices such as the computer mouse and the floppy disk, but also better UI such as graphical interfaces.

For robots to achieve the same, they require better HMIs. As robots are spatial systems, they need HMIs that can comfortably handle 3D spaces. HMDs offer plenty of sensors and input methods as well as providing spatial visualization through AR. HMDs are fully mobile and can be complemented with other wearable systems to extend their capabilities. Mobility is extremely beneficial in future applications where robots outnumber the human workers, as it allows single users to interact with many robots. Throughout this work we have developed novel algorithms to prove than HMDs can be used for everything from robot set up to human interaction with robots. In a recent work, Lauritsen *et al.* [148] showed that HMDs and AR can also be useful in the installation and deployment of industrial robots. Thus, HMDs are an appropriate choice of an HMI that can handle every step of the robot use.

The next step of this research would be to implement an integrated holistic system, with interfaces to different robots and robot manufacturers. Further goals would be to make the user interaction more fluid and preform user tests and implement a test-feedback-improvement cycle. As previously mentioned, increasing localisation accuracy, improving communication between devices and allowing real-time control of the robot through the desktop computer are all open challenges.

Once a practical working system is achieved, such a system should then be modified to become an AR-based HRI programming environment to further the development of different apps and use cases. The components already developed would then constitute the basis of the HRI functions of the system. Combining these functions and adding new ones would drastically increase the ease and range of use cases for industrial robots.

We hope this works provides a good overview of what can be done with HMDs in the field of HRI and that some may get inspired to pursue this line of research and further develop this promising research area. Thank you for reading.

Bibliography

[1] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *IEICE TRANSACTIONS on Information and Systems*, vol. 77, no. 12, pp. 1321–1329, 1994.

[2] Microsoft. (2018) Hololens (1st gen) hardware. [Online]. Available: https://www.microsoft.com/en-us/research/blog/microsoft-hololens-facilitates-computer-vision-research-by-providing-access-to-raw-image-sensor-streams-with-research-mode/

[3] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald *et al.*, "The kuka-dlr lightweight robot arm-a new reference platform for robotics research and manufacturing," in *Robotics (ISR), 2010 41st international symposium on and 2010 6th German conference on robotics (ROBOTIK)*. VDE, 2010, pp. 1–8.

[4] R. D. Schraft and C. Meyer, "The need for an intuitive teaching method for small and medium enterprises," *VDI BERICHTE*, vol. 1956, p. 95, 2006.

[5] M. Hägele, K. Nilsson, and J. N. Pires, *Industrial Robotics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 963–986. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5_43

[6] R. Bischoff and A. Kazi, "Perspectives on augmented reality based human-robot interaction with industrial robots," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4. IEEE, 2004, pp. 3226–3231.

[7] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE computer graphics and applications*, vol. 21, no. 6, pp. 34–47, 2001. [Online]. Available: https://ieeexplore.ieee.org/document/963459

[8] D. Drascic, J. J. Grodski, P. Milgram, K. Ruffo, P. Wong, and S. Zhai, "Argos: A display system for augmenting reality," in *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, ser. CHI '93. New York, NY, USA: ACM, 1993, pp. 521–. [Online]. Available: http://doi.acm.org/10.1145/169059.169506

[9] T. Williams, D. Szafir, and T. Chakraborti, "The reality-virtuality interaction cube: A framework for conceptualizing mixed-reality interaction design elements for hri," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 520–521.

[10] M. Walker, T. Phung, T. Chakraborti, T. Williams, and D. Szafir, "Virtual, augmented, and mixed reality for human-robot interaction: A survey and virtual

design element taxonomy," *arXiv preprint arXiv:2202.11249*, 2022. [Online]. Available: https://arxiv.org/abs/2202.11249

[11] R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt, "Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3491102.3517719

[12] T. Williams, D. Szafir, T. Chakraborti, and E. Phillips, "Virtual, augmented, and mixed reality for human-robot interaction (vam-hri)," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2019, pp. 671–672.

[13] G. Cortes, E. Marchand, G. Brincin, and A. Lécuyer, "Mosart: Mobile spatial augmented reality for 3d interaction with tangible objects," *Frontiers in Robotics and AI*, vol. 5, p. 93, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2018.00093

[14] J. P. Rolland, F. Biocca, F. Hamza-Lup, Y. Ha, and R. Martins, "Development of head-mounted projection displays for distributed, collaborative, augmented reality applications," *Presence: Teleoperators and Virtual Environments*, vol. 14, no. 5, pp. 528–549, 2005. [Online]. Available: https://doi.org/10.1162/105474605774918741

[15] X. Cao, C. Forlines, and R. Balakrishnan, "Multi-user interaction using handheld projectors," in *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM, 2007, pp. 43–52.

[16] K. D. Willis, I. Poupyrev, S. E. Hudson, and M. Mahler, "Sidebyside: ad-hoc multi-user interaction with handheld projectors," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 431–440.

[17] P. Milgram, S. Zhai, D. Drascic, and J. Grodski, "Applications of augmented reality for human-robot communication," in *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 3, Jul 1993, pp. 1467–1472 vol.3.

[18] A. Rastogi, P. Milgram, D. Drascic, and J. J. Grodski, "Telerobotic control with stereoscopic augmented reality," in *Stereoscopic Displays and Virtual Reality Systems III*, M. T. Bolas, S. S. Fisher, M. T. Bolas, S. S. Fisher, and J. O. Merritt, Eds., vol. 2653, International Society for Optics and Photonics. SPIE, 1996, pp. 115 – 122. [Online]. Available: https://doi.org/10.1117/12.237424

[19] J. C. Maida and C. Bowen, "Utilization of the space vision system as an augmented reality system for mission operations," 2003.

[20] N. R. Corby Jr and C. A. Nafis, "Augmented reality telemanipulation system for nuclear reactor inspection," in *Telemanipulator and Telepresence Technologies*, vol. 2351. International Society for Optics and Photonics, 1995, pp. 360–365.

[21] S. Kohn, A. Blank, D. Puljiz, L. Zenkel, O. Bieber, B. Hein, and J. Franke, "Towards a real-time environment reconstruction for vr-based teleoperation through model segmentation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1–9.

[22] I. Gonsher, Y. Han, K. Desingh, and A. Gokaslan, "Prototyping mixed reality large screen mobile telepresence robots," in *Proceedings of the 5th International Workshop on Virtual, Augmented, and Mixed Reality for HRI (VAM-HRI)*, 2022.

[23] H. Fang, S. Ong, and A. Nee, "Interactive robot trajectory planning and simulation using augmented reality," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 227 – 237, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584511001116

[24] G. Reinhart, U. Munzert, and W. Vogl, "A programming system for robot-based remote-laser-welding with conventional optics," *CIRP Annals*, vol. 57, no. 1, pp. 37 – 40, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0007850608000917

[25] M. F. Zaeh and W. Vogl, "Interactive laser-projection for programming industrial robots," in *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*, Oct 2006, pp. 125–128.

[26] D. Sprute, K. Tönnies, and M. König, "Virtual borders: Accurate definition of a mobile robot's workspace using augmented reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 8574–8581.

[27] D. Sprute, P. Viertel, K. Tönnies, and M. König, "Learning virtual borders through semantic scene understanding and augmented reality," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4607–4614.

[28] K. Hosoi, V. N. Dao, A. Mori, and M. Sugimoto, "Visicon: A robot control interface for visualizing manipulation using a handheld projector," in *Proceedings of the International Conference on Advances in Computer Entertainment Technology*, ser. ACE '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 99–106. [Online]. Available: https://doi.org/10.1145/1255047.1255068

[29] T. Hiraki, S. Fukushima, Y. Kawahara, and T. Naemura, "Navigatorch: Projection-based robot control interface using high-speed handheld projector," in *SIGGRAPH Asia 2019 Emerging Technologies*, ser. SA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 31–33. [Online]. Available: https://doi.org/10.1145/3355049.3360538

[30] D. Schmidt, D. Molyneaux, and X. Cao, *PICOntrol: Using a Handheld Projector for Direct Control of Physical Devices through Visible Light*. New York, NY, USA: Association for Computing Machinery, 2012, p. 379–388. [Online]. Available: https://doi.org/10.1145/2380116.2380166

[31] P. Hübner, K. Clintworth, Q. Liu, M. Weinmann, and S. Wursthorn, "Evaluation of hololens tracking and depth sensing for indoor mapping applications," *Sensors*, vol. 20, pp. 1021:1–23, 02 2020.

[32] T. Petković, D. Puljiz, I. Marković, and B. Hein, "Human intention estimation based on hidden markov model motion validation for safe flexible robotized warehouses," *Robotics and Computer-Integrated Manufacturing*, vol. 57, pp. 182–196, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584518302965

[33] D. Puljiz, F. Krebs, F. Bosing, and B. Hein, "What the hololens maps is your workspace: Fast mapping and set-up of robot cells via head mounted displays and augmented reality," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 11 445–11 451.

[34] D. Puljiz, A. G. Vasilache, M. Mende, and B. Hein, "Inside-out infrared marker tracking via head mounted displays for smart robot programming," in *5th International Workshop on Virtual, Augmented, and Mixed Reality for HRI*, 2022. [Online]. Available: https://openreview.net/forum?id=raMebPsrtyc

[35] C. Kemp, "Wearables and robots: A shared view," *IEEE Pervasive Computing*, vol. 5, no. 3, pp. 16–20, 2006.

[36] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[37] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 3–15, Jan 2003.

[38] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[39] K. Konolige, "Small vision systems: Hardware and implementation," in *Robotics research*. Springer, 1998, pp. 203–212.

[40] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[41] M. Zolotas, J. Elsdon, and Y. Demiris, "Head-mounted augmented reality for explainable robotic wheelchair assistance," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1823–1829.

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

[43] H. Fang, S. Ong, and A. Nee, "Interactive robot trajectory planning and simulation using augmented reality," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 227 – 237, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584511001116

[44] C. Mateo, A. Brunete, E. Gambao, and M. Hernando, "Hammer: An android based application for end-user industrial robot programming," in *Mechatronic and Embedded Systems and Applications (MESA), 2014 IEEE/ASME 10th International Conference on*. IEEE, 2014, pp. 1–6.

[45] S. Stadler, K. Kain, M. Giuliani, N. Mirnig, G. Stollnberger, and M. Tscheligi, "Augmented reality for industrial robot programmers: Workload analysis for task-based, augmented reality-supported robot control," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug 2016, pp. 179–184.

[46] D. Krupke, F. Steinicke, P. Lubos, Y. Jonetzko, M. Görner, and J. Zhang, "Comparison of multimodal heading and pointing gestures for co-located mixed reality human-robot interaction," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1–9.

[47] S. Hashimoto, A. Ishida, M. Inami, and T. Igarashi, "Touchme: An augmented reality interface for remote robot control." *JRM*, vol. 25, no. 3, pp. 529–537, 2013.

[48] H. Liu, Y. Zhang, W. Si, X. Xie, Y. Zhu, and S.-C. Zhu, "Interactive robot knowledge patching using augmented reality," in *2018 IEEE 21st International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1947 –1954.

[49] F. Chu, R. Xu, Z. Zhang, P. A. Vela, and M. Ghovanloo, "Hands-free assistive manipulator using augmented reality and tongue drive system," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 5463–5468.

[50] J. Elsdon and Y. Demiris, "Augmented reality for feedback in a shared control spraying task," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1939–1946.

[51] G. Cimen, Y. Yuan, R. W. Sumner, S. Coros, and M. Guay, "Interacting with intelligent characters in ar," *International SERIES on Information Systems and Management in Creative eMedia (CreMedia)*, no. 2017/2, pp. 24–29, 2018.

[52] M. Ostanin, R. Yagfarov, and A. Klimchik, "Interactive robots control using mixed reality ∗∗the work presented in this paper was supported by the grant of russian science foundation 17-19-01740." *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 695–700, 2019, 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896319312613

[53] M. Ostanin, S. Mikhel, A. Evlampiev, V. Skvortsova, and A. Klimchik, "Human-robot interaction for robotic manipulator programming in mixed reality," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2805–2811.

[54] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[55] J. Miseikis, P. Knobelreiter, I. Brijacak, S. Yahyanejad, K. Glette, O. J. Elle, and J. Torresen, "Robot localisation and 3d position estimation using a free-moving camera and cascaded convolutional neural networks," in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2018, pp. 181–187.

[56] N. Mellado, D. Aiger, and N. J. Mitra, "Super 4pcs fast global pointcloud registration via smart indexing," *Computer Graphics Forum*, vol. 33, no. 5, pp. 205–215, 2014. [Online]. Available: http://dx.doi.org/10.1111/cgf.12446

[57] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze, "Our-cvfh–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation," in *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*. Springer, 2012, pp. 113–122.

[58] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010.

[59] S. Salti, F. Tombari, and L. Di Stefano, "Shot: Unique signatures of histograms for surface and texture description," *Computer Vision and Image Understanding*, vol. 125, pp. 251–264, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314214000988

[60] G. Popović, A. Hadviger, I. Marković, and I. Petrović, "Computationally efficient dense moving object detection based on reduced space disparity estimation," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 360–365, 2018.

[61] R. Vassallo, A. Rankin, E. C. Chen, and T. M. Peters, "Hologram stability evaluation for Microsoft HoloLens," in *Medical Imaging 2017: Image Perception, Observer Performance, and Technology Assessment*, vol. 10136. International Society for Optics and Photonics, 2017, p. 1013614. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10136/1013614/Hologram-stability-evaluation-for-Microsoft-HoloLens/10.1117/12.2255831.short

[62] I. Sipiran and B. Bustos, "Harris 3d: A robust extension of the harris operator for interest point detection on 3d meshes," *The Visual Computer*, vol. 27, pp. 963–976, 11 2011.

[63] D. Puljiz, K. S. Riesterer, B. Hein, and T. Kröger, "Referencing between a head-mounted device and robotic manipulators," in *Proc. of the 2nd International Workshop on Virtual, Augmented, and Mixed Reality for Human-Robot Interactions (VAM-HRI)*, 2019.

[64] Z. Pan, J. Polden, N. Larkin, S. V. Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," in *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, June 2010, pp. 1–8.

[65] P. Neto and N. Mendes, "Direct off-line robot programming via a common cad package," *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 896 – 910, 2013.

[66] M. Vincze, A. Pichler, and G. Biegelbauer, "Detection of classes of features for automated robot programming," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 1, Sep. 2003, pp. 151–156 vol.1.

[67] S. K. Ong, J. W. S. Chong, and A. Y. Nee, "A novel ar-based robot programming and path planning methodology," *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 3, pp. 240–249, 2010.

[68] D. Lee and Y. S. Park, "Implementation of augmented teleoperation system based on Robot Operating System (ROS)," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5497–5502. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8594482

[69] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, p. 509–517, sep 1975. [Online]. Available: https://doi.org/10.1145/361002.361007

[70] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[71] K. Khoshelham, H. Tran, and D. Acharya, "Indoor Mapping Eyewear: Geometric Evaluation of Spatial Mapping Capability of Hololens," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4213, pp. 805–810, Jun. 2019.

[72] G. Ajaykumar, M. Steele, and C.-M. Huang, "A survey on end-user robot programming," *ACM Comput. Surv.*, vol. 54, no. 8, oct 2021. [Online]. Available: https://doi.org/10.1145/3466819

[73] V. Villani, F. Pini, F. Leali, C. Secchi, and C. Fantuzzi, "Survey on human-robot interaction for robot programming in industrial applications," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 66–71, 2018, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318313600

[74] Z. Zhou, R. Xiong, Y. Wang, and J. Zhang, "Advanced robot programming: A review," *Current Robotics Reports*, vol. 1, pp. 251–258, 2020.

[75] M. Hanses, R. Behrens, and N. Elkmann, "Hand-guiding robots along predefined geometric paths under hard joint constraints," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2016, pp. 1–5.

[76] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[77] S. Moe and I. Schjølberg, "Real-time hand guiding of industrial manipulator in 5 dof using microsoft kinect and accelerometer," in *2013 IEEE RO-MAN*, Aug 2013, pp. 644–649.

[78] S. D. Lee, K. H. Ahn, and J. B. Song, "Torque control based sensorless hand guiding for direct robot teaching," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 745–750.

[79] M. Ragaglia, A. M. Zanchettin, L. Bascetta, and P. Rocco, "Accurate sensorless lead-through programming for lightweight robots in structured environments," *Robotics and Computer-Integrated Manufacturing*, vol. 39, pp. 9 – 21, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S073658451500126X

[80] A. Stolt, F. B. Carlson, M. M. G. Ardakani, I. Lundberg, A. Robertsson, and R. Johansson, "Sensorless friction-compensated passive lead-through programming for industrial robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 3530–3537.

[81] J. M. Rodríguez Corral, I. Ruíz-Rube, A. Civit Balcells, J. M. Mota-Macías, A. Morgado-Estévez, and J. M. Dodero, "A study on the suitability of visual languages for non-expert robot programmers," *IEEE Access*, vol. 7, pp. 17 535–17 550, 2019.

[82] M. Wächter, S. Ottenhaus, M. Kröhnert, N. Vahrenkamp, and T. Asfour, "The armarx statechart concept: Graphical programing of robot behavior," *Frontiers in Robotics and AI*, vol. 3, 2016. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2016.00033

[83] Y. Hua, S. Zander, M. Bordignon, and B. Hein, "From automationml to ros: A model-driven approach for software engineering of industrial robotics using ontological reasoning," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016, pp. 1–8.

[84] T. B. Ionescu, "Leveraging graphical user interface automation for generic robot programming," *Robotics*, vol. 10, no. 1, p. 3, 2020.

[85] M. Edmonds, F. Gao, X. Xie, H. Liu, S. Qi, Y. Zhu, B. Rothrock, and S.-C. Zhu, "Feeling the force: Integrating force and pose for fluent discovery through imitation learning to open medicine bottles," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3530–3537.

[86] Y. Shavit, N. Figueroa, S. S. M. Salehian, and A. Billard, "Learning augmented joint-space task-oriented dynamical systems: A linear parameter varying and synergetic control approach," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2718–2725, 2018.

[87] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020. [Online]. Available: https://doi.org/10.1146/annurev-control-100819-063206

[88] B. Singh, R. Kumar, and V. P. Singh, "Reinforcement learning in robotic applications: a comprehensive survey," *Artificial Intelligence Review*, pp. 1–46, 2022.

[89] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International journal of robotics research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[90] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 737–744.

[91] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning – ICANN 2018*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds. Cham: Springer International Publishing, 2018, pp. 270–279.

[92] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, "Palm-e: An embodied multimodal language model," in *arXiv preprint arXiv:2303.03378*, 2023.

[93] B. Hein and H. Wörn, "Intuitive and model-based on-line programming of industrial robots: New input devices," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3064–3069.

[94] A. Gaschler, M. Springer, M. Rickert, and A. Knoll, "Intuitive robot tasks with augmented reality and virtual obstacles," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6026–6031.

[95] S. Ong, A. Yew, N. Thanigaivel, and A. Nee, "Augmented reality-assisted robot programming system for industrial applications," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101820, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584519300250

[96] M. B. Luebbers, C. Brooks, C. L. Mueller, D. Szafir, and B. Hayes, "Arc-lfd: Using augmented reality for interactive long-term robot skill maintenance via constrained learning from demonstration," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 3794–3800.

[97] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, "Ganerated hands for real-time 3d hand tracking from monocular rgb," in *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2018. [Online]. Available: https://handtracker.mpi-inf.mpg.de/projects/GANeratedHands/

[98] D. Hartmann, M. Mende, D. Štogl, B. Hein, and T. Kröger, "Robot-based machining of unmodeled objects via feature detection in dense point clouds," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7777–7783.

[99] B. Horn, H. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *Journal of the Optical Society of America A*, vol. 5, pp. 1127–1135, 07 1988.

[100] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[101] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, pp. 376–380, 1991.

[102] K. Kanatani, "Analysis of 3-d rotation fitting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 543–549, 1994.

[103] J. Challis, "A procedure for determining rigid body transformation parameters," *Journal of Biomechanics*, vol. 28, no. 6, pp. 733–737, Jun. 1995.

[104] K. Shoemake, "Animating rotation with quaternion curves," in *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 1985, pp. 245–254.

[105] W. P. Chan, G. Hanks, M. Sakr, T. Zuo, H. Machiel Van der Loos, and E. Croft, "An augmented reality human-robot physical collaboration interface design for shared, large-scale, labour-intensive manufacturing tasks," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 11 308–11 313.

[106] T. Kröger, "Opening the door to new sensor-based robot applications—the reflexxes motion libraries," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1–4.

[107] F. Bérard, G. Wang, and J. R. Cooperstock, "On the limits of the human motor control precision: The search for a device's human resolution," in *Human-Computer Interaction – INTERACT 2011*, P. Campos, N. Graham, J. Jorge, N. Nunes, P. Palanque, and M. Winckler, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 107–122.

[108] D. Puljiz, E. Stöhr, K. S. Riesterer, B. Hein, and T. Kröger, "Sensorless hand guidance using microsoft hololens," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2019, pp. 632–633.

[109] ——, "General hand guidance framework using microsoft hololens," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5185–5190.

[110] D. Puljiz and B. Hein, "Concepts for end-to-end augmented reality based human-robot interaction systems," *Factory of the Future Workshop, IROS 2019*, 2019.

[111] J. T. C. Tan, F. Duan, Y. Zhang, K. Watanabe, R. Kato, and T. Arai, "Human-robot collaboration in cellular manufacturing: Design and development," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 29–34.

[112] C. Vogel and J. Saenz, "Optical workspace monitoring system for safeguarding tools on the mobile manipulator valeri," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*, June 2016, pp. 1–6.

[113] C. Vogel, C. Walter, and N. Elkmann, "A projection-based sensor system for safe physical human-robot collaboration," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 5359–5364.

[114] C. Vogel, M. Fritzsche, and N. Elkmann, "Safe human-robot cooperation with high-payload robots in industrial applications," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2016, pp. 529–530.

[115] J. A. Corrales, F. A. Candelas, and F. Torres, "Hybrid tracking of human operators using imu/uwb data fusion by a kalman filter," in *2008 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2008, pp. 193–200.

[116] J. Fan, W. Xu, Y. Wu, and Y. Gong, "Human tracking using convolutional neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1610–1623, Oct 2010.

[117] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 1302–1310.

[118] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: Real-time 3d human pose estimation with a single rgb camera," *ACM Transactions on Graphics*, vol. 36, no. 4, July 2017. [Online]. Available: http://gvv.mpi-inf.mpg.de/projects/VNect/

[119] K. C. Hoang, W. P. Chan, S. Lay, A. Cosgun, and E. Croft, "Virtual barriers in augmented reality for safe and effective human-robot cooperation in manufacturing," in *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2022, pp. 1174–1180.

[120] P. Jarzem and R. Gledhill, "Predicting height from arm measurements," *Journal of Pediatric Orthopaedics*, vol. 13, no. 6, pp. 761–765, 1993.

[121] S. Reeves, "The relationship between arm-span measurement and height with special reference to gender and ethnicity." *european Journal of clinical nutrition*, vol. 50, no. 6, pp. 398–400, 1996.

[122] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, June 2011, pp. 1–7.

[123] L. M. Pereira *et al.*, "State-of-the-art of intention recognition and its use in decision making," *AI Communications*, vol. 26, no. 2, pp. 237–246, 2013.

[124] H.-I. Lin and W.-K. Chen, "Human intention recognition using markov decision processes," in *2014 CACS International Automatic Control Conference (CACS 2014)*, 2014, pp. 340–343.

[125] C. Schlenoff, Z. Kootbally, A. Pietromartire, M. Franaszek, and S. Foufou, "Intention recognition in manufacturing applications," *Robotics and Computer-Integrated Manufacturing*, vol. 33, pp. 29–41, 2015, special Issue on Knowledge Driven Robotics and Manufacturing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S073658451400057X

[126] L. Bascetta, G. Ferretti, P. Rocco, H. Ardö, H. Bruyninckx, E. Demeester, and E. Di Lello, "Towards safe human-robot interaction in robotic cells: An approach based on visual tracking and intention estimation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 2971–2978.

[127] Q. Wang, W. Jiao, R. Yu, M. T. Johnson, and Y. Zhang, "Virtual reality robot-assisted welding based on human intention recognition," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 799–808, 2020.

[128] T. Chakraborti, S. Sreedharan, A. Kulkarni, and S. Kambhampati, "Projection-aware task planning and execution for human-in-the-loop operation of robots in a mixed-reality workspace," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4476–4482.

[129] R. Singh, T. Miller, J. Newn, L. Sonenberg, E. Velloso, and F. Vetere, "Combining planning with gaze for online human intention recognition," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018, p. 488–496.

[130] A. D. Dragan, S. Bauman, J. Forlizzi, and S. S. Srinivasa, "Effects of robot motion on human-robot collaboration," in *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2015, pp. 51–58.

[131] T. Williams, M. Bussing, S. Cabrol, E. Boyle, and N. Tran, "Mixed reality deictic gesture for multi-modal robot communication," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2019, pp. 191–201.

[132] M. Walker, H. Hedayati, J. Lee, and D. Szafir, "Communicating robot motion intent with augmented reality," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '18. New York,

NY, USA: Association for Computing Machinery, 2018, p. 316–324. [Online]. Available: https://doi.org/10.1145/3171221.3171253

[133] H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized voronoi graph," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 96–125, 2000.

[134] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, pp. 3310–3317 vol.4.

[135] G. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[136] C. Zhu, Q. Cheng, and W. Sheng, "Human intention recognition in smart assisted living systems using a hierarchical hidden markov model," in *2008 IEEE International Conference on Automation Science and Engineering*, 2008, pp. 253–258.

[137] D. Puljiz, B. Zhou, K. Ma, and B. Hein, "HAIR: Head-mounted AR intention recognition," in *4th International Workshop on Virtual, Augmented, and Mixed Reality for HRI*, 2021. [Online]. Available: https://openreview.net/forum?id=bR9Bc2NApG8

[138] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation," *Foundations and Trends® in Signal Processing*, vol. 11, no. 1-2, pp. 1–153, 2017. [Online]. Available: http://dx.doi.org/10.1561/2000000094

[139] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[140] B. D. Adelstein, T. G. Lee, and S. R. Ellis, "Head tracking latency in virtual environments: psychophysics and a model," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 47, no. 20. SAGE Publications Sage CA: Los Angeles, CA, 2003, pp. 2083–2087.

[141] M. Arbo, I. Eriksen, F. Sanfilippo, and J. Gravdahl, "Comparison of kvp and rsi for controlling kuka robots over ros∗∗the work reported in this paper was based on activities within centre for research based innovation sfi manufacturing in norway, and is partially funded by the research council of norway under contract number 237900." *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9841–9846, 2020, 21st IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320334509

[142] J. Allspaw, G. LeMasurier, and H. Yanco, "Comparing performance between different implementations of ros for unity," *6th International Workshop on Virtual, Augmented, and Mixed Reality for HRI*, 2023. [Online]. Available: https://openreview.net/forum?id=WH3yhsbBjj

[143] R. Suzuki, J. Kato, M. D. Gross, and T. Yeh, "Reactile: Programming swarm user interfaces through direct physical manipulation," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–13. [Online]. Available: https://doi.org/10.1145/3173574.3173773

[144] Y. Cao, T. Wang, X. Qian, P. S. Rao, M. Wadhawan, K. Huo, and K. Ramani, "Ghostar: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality," in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 521–534. [Online]. Available: https://doi.org/10.1145/3332165.3347902

[145] I. Soares, R. B. Sousa, M. Petry, and A. P. Moreira, "Accuracy and repeatability tests on hololens 2 and htc vive," *Multimodal Technologies and Interaction*, vol. 5, no. 8, 2021. [Online]. Available: https://www.mdpi.com/2414-4088/5/8/47

[146] I. Matyash, R. Kutzner, T. Neumuth, and M. Rockstroh, "Accuracy measurement of hololens2 imus in medical environments," *Current Directions in Biomedical Engineering*, vol. 7, no. 2, pp. 633–636, 2021.

[147] M. Unger, S. Heinrich, M. Rick, D. Halama, and C. Chalopin, "Hologram accuracy evaluation of hololens 2 for thermal imaging in medical applications," *Current Directions in Biomedical Engineering*, vol. 8, no. 2, pp. 193–196, 2022.

[148] O. B. Lauritsen, J. Andersen, K. Zielinski, and M. B. Kjærgaard, "Ar-based ui for improving io setup in robot deployment process," *6th International Workshop on Virtual, Augmented, and Mixed Reality for HRI*, 2023. [Online]. Available: https://openreview.net/forum?id=18dfsIIr41