



Hochschule für Angewandte Wissenschaften Hamburg  
*Hamburg University of Applied Sciences*

## **Master Thesis**

Mohan Pannirselvam

# **Intuitive Mixed Reality Robot Program Generation for Pick and Place Tasks**

*Fakultät Technik und Informatik  
Department Fahrzeugtechnik und Flugzeugbau*

*Faculty of Engineering and Computer Science  
Department of Automotive and  
Aeronautical Engineering*



**Mohan Pannirselvam**

**Title:**

**Intuitive Mixed Reality Robot Program  
Generation for Pick and Place Tasks**

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Flugzeugbau (M.Sc)  
am Department Fahrzeugtechnik und Flugzeugbau  
der Fakultät Technik und Informatik  
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:  
Fraunhofer-Institut für Fertigungstechnik und  
Angewandte Materialforschung IFAM  
Ottenbecker Damm 12  
21684 Stade

Erstprüfer/in : Prof. Dr. Kai von Luck  
Zweitprüfer/in : Dipl.Ing. Björn Reichel

Abgabedatum: 09-04-2021

# Zusammenfassung

**Name des Studierenden: Mohan Pannirselvam**

**Titel der Masterthesis**

Intuitive Mixed-Reality-Roboterprogrammierung für Pick-and-Place-Aufgaben

**Stichworte**

Mixed Reality, Roboterprogrammierung, Intuitive Benutzeroberfläche, Pick and Place

**Kurzzusammenfassung**

Die Roboterprogrammierung geht ständig in der Richtung von einer intuitiveren Methode. In den letzten Jahren ist die Effektivität, Genauigkeit und Zuverlässigkeit von Mixed Reality immer ausgereifter geworden. Die Technologie ist bereits in verschiedenen Bereichen wie Bau, Medizin und mehr implementiert. In dieser Arbeit wird das Potenzial der Mixed-Reality-Technologie, eine wichtige Rolle bei den Methoden der Roboterprogrammierung zu spielen, untersucht. Ein spezifischer Anwendungsfall der Roboterprogrammierung, die Pick-and-Place-Aufgabe, wird als Beispiel gewählt, um das Potenzial der Integration der Mixed-Reality-Technologie zu untersuchen. Es wird eine Recherche über den Stand der Technik der Roboterprogrammierung und der Mixed-Reality-Technologie durchgeführt. Außerdem werden ähnliche Projekte, die das gleiche Ziel verfolgen, zusammengefasst und mit nützlichen Erkenntnissen vorgestellt. Im Verlauf der Arbeit wird eine Mixed-Reality-basierte Architektur für die Roboterprogrammierung entworfen und entwickelt. Der Entwurfsteil nutzt Erkenntnisse und Anleitungen aus bestehenden ähnlichen Projekten, um eine innovative Idee zu generieren und ähnliche Fehler zu vermeiden. Grundlegende Prinzipien für den Entwurf einer Mixed-Reality-basierten Roboterprogrammieranwendung, wie die Verwendung eines modularen Designansatzes, eine Server-Client-Architektur und die Automatisierung der Roboterprogrammiererstellung werden in dieser Arbeit vorgestellt. Das entwickelte System wird durch einen systematischen Versuchsaufbau mit einem Laser Tracker analysiert und validiert. Aus der Versuchsanalyse wird eine Endgenauigkeit von  $\pm 5 \text{ cm}$  mit der entwickelten MR-Anwendung registriert. Die Identifizierung der Quelle von Ineffizienz und Ungenauigkeit innerhalb des Systems wird durch systematische Argumentation durchgeführt und die möglichen Schritte zur Verbesserung der Systemleistung werden ebenfalls angegeben. Folglich wird die Art der Pick-and-Place-Aufgabe, die durch dieses endgültige System durchgeführt werden kann, durch bestimmte Anforderungen an Art von Roboteraufgaben eingegrenzt. Abschließend wird ein konkretes Urteil über das Potenzial der MR-Technologie-Integration für Methoden der Roboterprogrammierung gefällt. Es werden mehrere Verbesserungsschritte und mögliche zukünftige Entwicklungen mit dem bestehenden System präsentiert.

# Summary

**Name of Student: Mohan Pannirselvam**

**Title of the paper**

Intuitive Mixed Reality Robot Programming for Pick and Place Tasks

**Keywords**

Mixed Reality, Robot Programming, natural user Interface, Pick and Place Tasks

**Abstract**

The robot programming is constantly involving in the direction of more intuitive ways and methods, to enable the operator to implement a robot task faster and easier. In recent years, the mixed reality maturing in terms of effectiveness, accuracy and reliability. The technology is already implemented in various fields like construction, medical and more. In this thesis, the potential of mixed reality technology to play a vital role in the robot programming methods is addressed. A specific robot programming use case, the pick and place task, is chosen as example to investigate the potential of the mixed reality technology integration. A research is conducted on the state of the art of robot programming and mixed reality technology. Furthermore, similar projects which address the same goal are summarized and presented with useful insights. Throughout the thesis, a mixed reality-based robot programming architecture is designed and developed. The design part uses insights and guidance from existing similar projects in order to generate an innovative idea and to avoid similar mistakes. Fundamental principles for designing a mixed reality-based robot programming application, like the use of modular design approach, a server-client architecture, and automation of robot program generation are introduced in this thesis. The developed system is analyzed and validated through a systematical experiment set-up using a laser tracker. From the experiment analysis a final accuracy of  $\pm 5\text{ cm}$  is registered with the developed MR application. The identification of the source of inefficiency and inaccuracy within the system is done through systematic reasoning and the possible steps for improvement of the system performance are stated as well. Consequently, the type of pick and place task which can be performed through this final system is narrowed down through certain requirements on the object which needs to be picked. Finally, a concrete verdict on the potential of MR technology integration for robot programming methods, is made. Several improvement steps and possible future developments with the existing system are outlined.

# Table of Contents

<b>Symbols and Abbreviations.....</b>	<b>i</b>
<b>List of Tables.....</b>	<b>ii</b>
<b>List of Figures.....</b>	<b>iii</b>
<b>1 Introduction .....</b>	<b>1</b>
1.1 Problem Statement .....	2
1.2 Thesis Goal.....	2
1.3 Thesis Structure.....	2
<b>2 Analysis .....</b>	<b>4</b>
2.1 Thesis Focus Field.....	4
2.2 Existing Robot Programming Methods.....	4
2.2.1 Syntax or Icon based Programming .....	6
2.2.2 Lead Through .....	7
2.2.3 Walk Through .....	8
2.2.4 Simulation/Offline Programming .....	9
2.2.5 Learning System based Robot Programming .....	10
2.3 Pick and Place Robot Task .....	11
2.4 State of the Art Mixed Reality Technology .....	12
2.4.1 Definition.....	12
2.4.2 Mixed Reality Working Principle .....	13
2.4.3 Pose Estimation in Mixed Reality .....	15
2.4.4 Mixed Reality Output Devices.....	16
2.4.5 MR Development Approaches.....	18
2.5 Related Works on Mixed Reality for Robot Programming .....	18
2.6 Laboratory Environment.....	21
2.6.1 Universal Robot 10 .....	22
2.6.2 Workstation PC.....	23
2.6.3 Unity IDE .....	23
2.7 Summary.....	23
<b>3 Concept and Design .....</b>	<b>25</b>
3.1 Requirement Definition and Analysis .....	25
3.1.1 Primary Functional Requirements .....	25
3.1.2 Secondary Functional Requirements.....	26
3.1.3 Non-Functional Requirements .....	30
3.2 General Conception .....	31
3.2.1 Programmable Digital Twin.....	31
3.2.2 Coordinate System Descriptions .....	33
3.2.3 Integration of Robot Operating System.....	34
3.2.4 Pick Object and Gripper Type Selection.....	36
3.2.5 Pointer Type Selection.....	37

3.3	Use Case Study .....	38
3.4	Summary.....	40
<b>4</b>	<b>System Implementation.....</b>	<b>42</b>
4.1	System Architecture .....	42
4.2	System Sequence.....	43
4.2.1	ROS Ecosystem .....	46
4.2.2	UR10 Custom Configuration.....	47
4.2.3	Unity IDE Project Setting .....	47
4.3	Modular Design Implementation .....	48
4.3.1	Communication Module.....	49
4.3.2	Interaction Module .....	50
4.3.3	Registration Module.....	53
4.3.4	Path Specification Module .....	56
4.3.5	Editing and Preview Module .....	58
4.3.6	Execution Module .....	59
4.3.7	Calibration Module.....	61
4.4	Summary.....	61
<b>5</b>	<b>Experiments and Evaluation .....</b>	<b>63</b>
5.1	Methodology and Set-Up .....	65
5.2	Results .....	70
5.2.1	Quantitative Performance .....	71
5.2.2	Qualitative Performance .....	78
5.3	Summary.....	80
<b>6</b>	<b>Conclusion .....</b>	<b>83</b>
	<b>References .....</b>	<b>85</b>

# Symbols and Abbreviations

MR	Mixed Reality
AR	Augmented Reality
VR	Virtual Reality
SDK	Software development kit
SLAM	Simultaneous localization and mapping
ML	Machine Learning
UI	User Interface
UX	User Experience
EE	End Effector
OS	Operating System
CPU	Central Processing Unit
GPU	Graphics Processing Unit
IMU	Inertial Measurement Unit
PbD	Programming by Demonstration
TCP	Tool Centre Point
IDE	Interactive Development Environment
URDF	Unified Robot Description Format
UR	Universal Robot
SRM	Spherical Retroreflective Mirror
KPI	Key Performance Indicator
OPC	Open Platform Communications



# List of Tables

Table 1 Advantages and disadvantages of syntax or icon-based programming.....	6
Table 2 Advantages and disadvantages of lead through methods .....	7
Table 3 Advantages and disadvantages of walk through method.....	8
Table 4 Advantages and disadvantages of simulation.....	10
Table 5 Advantages and disadvantages of learning system-based method .....	11
Table 6 Accuracy performance of various MR-ecosystem.....	20
Table 7 UR10 data sheet.....	22
Table 8 Workstation PC system specification.....	23
Table 9 Primary functional requirements .....	26
Table 10 Possible advantages of MR based robot programming method .....	27
Table 11 Secondary functional requirements .....	30
Table 12 Non-functional requirements.....	31
Table 13 Selection between real pointer and virtual pointer .....	37
Table 14 Selection of AR SDK.....	54

# List of Figures

Figure 2-1 Thesis focus filed.....	4
Figure 2-2 Classification of existing robot programming method .....	5
Figure 2-3 Fundamental syntax block of robot program code.....	6
Figure 2-4 The working principle of mixed reality technology .....	14
Figure 2-5 Recognition of feature points from RGB data [5].....	15
Figure 2-6"Magic lens" type MR effect done with handheld device [4].....	17
Figure 2-7 HMD Device example: HoloLens 2.....	18
Figure 2-8 Marker-based AR stylus to demonstrate the robot path line. [4] [6].....	19
Figure 2-9 UR10 joint designations .....	22
Figure 3-1 Workflow of the design process.....	25
Figure 3-2 Intuitiveness level associated with robot programming methods.....	29
Figure 3-3 Programmable digital twin concept.....	33
Figure 3-4 Basic ROS system architecture.....	35
Figure 3-5 Suction gripper structure and the pick object.....	37
Figure 3-6 Use case study with the combination of the general concepts .....	39
Figure 3-7 Final concept of the system.....	41
Figure 4-1 Final system architecture.....	43
Figure 4-2 System sequence diagram.....	45
Figure 4-3 Modular components of the system.....	48
Figure 4-4 Virtual draggable end-effectors with custom gizmo .....	51
Figure 4-5 Final UI of the stand-alone application .....	52
Figure 4-6 MR UI of the HoloLens version of the client app .....	53
Figure 4-7 Digital twin registered with different type of MR-Targets .....	55

Figure 4-8 Valve structure used as model target .....	56
Figure 4-9 Waypoints represented by 3D arrow model.....	58
Figure 4-10 Movable red-ball in the autonomous mode .....	60
Figure 5-1 Measurement zone and quadrant separation .....	67
Figure 5-2 Custom end effector .....	67
Figure 5-3 Pointer structure and virtual end effector pointer.....	70
Figure 5-4 Results of the experiment 1 and experiment 2 on quadrant 1 .....	71
Figure 5-5 Results of the experiment 1 and experiment 2 on quadrant 2 .....	71
Figure 5-6 Final offset magnitude of measurements .....	73
Figure 5-7 Offset variance form the respective mean value of each axis .....	75
Figure 5-8 Estimation of pick object parameter .....	76
Figure 5-9 Execution of a pick and place task with the MR application .....	79

# 1 Introduction

Robot programming is a fundamental step that needs to be undertaken to perform a robot operation. However, the robot programming method involves a complex and tedious procedure that the operator needs to go through in order to implement robot commands which are necessary for the completion of the desired robot operation. This creates a situation where only a group of people with specific knowledge and experience in robot programming can perform a robot operation task. In the past years, the robot programming field undergoes active research and innovation to produce a more intuitive and easier way of robot programming. For example, teach pendant method, kinesthetic teaching and machine learning-based robot programming. Such methods enable the operator to perform robot operations faster and easier. Yet, those methods still exhibit demands of robot programming experience, acquisition of various industrial terms and programming skills in various degrees on the operator side.

This situation creates a limitation in the current usage of robots. Nowadays, robot systems are slowly being installed in non-industrial scenarios like small and medium-sized enterprises (SME), entertainment, hospitality, medicine and the military. In most cases, the person who is utilizing the robot directly in such industries does not have a high level of robot programming knowledge or experience in operating a robot. Therefore, a separate personal needs to be available in order to program the robot when needed. This creates a situation where the procedure of reprogramming of the robot is not straightforward and demands a lot of time and resources from such industries. This slows down the integration of robots in such industries and the full potential of the robot can't be utilized due to the complexity of its programming steps.

An alternative method needs to be available where a common user can command a robot to perform a robot operation successfully and independently. Such method should reduce the complexity of the robot programming on the user side so that the user can focus fully on the complexity of the task instead. One of the possible solution for a such method can be found through the uprising mixed reality technology. This technology is a form of interactive display technology which provides a new form of information transfer to the user. Mixed reality technology has contributed profound impact in fields like worker training, education, entertainment, construction, medicine and many more. In this thesis, the possibility of mixed reality to solve the previously mentioned challenge is addressed through scientific research and by designing and deploying such mixed reality-based robot programming system as a proof of concept. Furthermore, the performances and reliability of the deployed mixed reality system are validated and analyzed through experiments and through systematic reasoning. Finally, a concrete verdict on the potential of MR technology

integration for robot programming methods is made. Several improvement steps and possible future developments with the existing system are outlined.

## **1.1 Problem Statement**

The existing robot programming methods demand knowledge of robot programming and experience. The user needs to be competent in field of programming and industrial terms in order to execute a successful robot task. A method should be designed which enables the user to concentrate on the complexity of the robot task instead of the complexity of the robot programming. Mixed reality technology promises a potential solution for such robot programming methods. Recently the mixed reality technology has undergone profound improvement in perspectives of hardware and software. Which in return the technology offers an immersive, stable and interactive user interface that can be utilized to revolutionize the format of robot programming.

## **1.2 Thesis Goal**

The motivation of this thesis is to justify and verify the integration of mixed reality technology as part of robot programming method. In order to do that, a robot program generation system which functions along with a mixed reality technology needs to be designed and developed as a proof of concept. Furthermore, the developed system should carry out a typical robot task like the pick and place task successfully. This is important to determine the limitation and challenges which surrounds the concept when it is used to perform an actual robot task. Furthermore, the developed system should enable a user to execute the pick and place task with an intuitive and flexible manner than the existing robot programming methods. Other than that, the developed system should be tested out for its performance and reliability through experiments. Finally, based on the results and performance output of the system, a concrete conclusion should be derived to justify the possibility of a mixed reality-based robot program generation concept overall.

## **1.3 Thesis Structure**

The rest of the thesis is structured into five main chapters. Chapter 2 presents related background information on the state-of-the-art robot programming methods and mixed reality technology. Furthermore, similar projects which are done in the field of mixed reality based robot programming are summarized and analyzed. From there important insights for the concept design are derived. In the following chapter 3, the concept is converted into a functional system architecture that can fulfil this thesis goals. The main design of the system and its components are presented in this chapter as well. Meanwhile, in chapter 4, the procedure and steps that had been taken to implement this design in a real lab environment are outlined. In chapter 5,

the performance of the developed system is validated and analyzed through experiments. In this chapter, the results of the experiment and the analyzes which made based on systematic reasoning are presented. Finally, in chapter 6 the final verdict on the possibility of mixed reality technology to be used for robot programming will be presented. Furthermore, an outlook for the thesis topic will be outlined in this chapter as well.

## 2 Analysis

In this chapter, the state-of-the-art robot programming methods and mixed reality technologies are presented. The important technical terms and working principles of the technologies which are related to this focus field are identified and analyzed. Previous projects and researches regarding this topic are presented and important insights and conclusions are derived. The testing environment and the hardware resources which act as the boundary condition of the thesis are stated and analyzed.

### 2.1 Thesis Focus Field

The focus of this thesis is the combination of robot programming methods and mixed reality technology. This overlapping field holds the potential for a new form of robot programming approaches that can yield a more intuitive and simpler form to control a robot. The advantages and disadvantages of the currently available robot programming need to be identified. The current state of mixed reality technology needs to be studied and the key parts which are relevant for possible integration into robot programming tasks need to be identified.

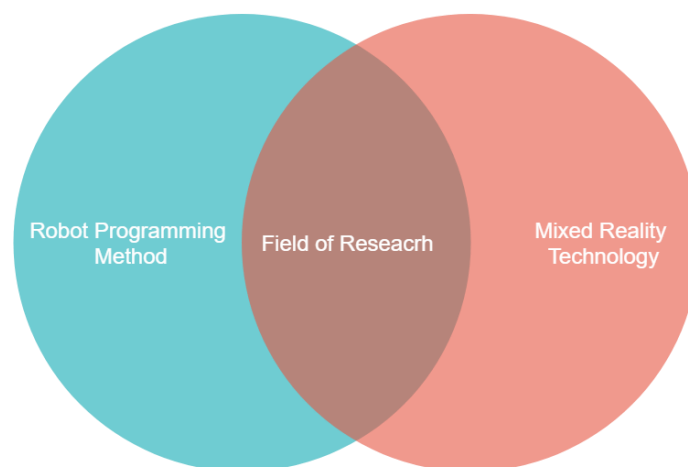


Figure 2-1 Thesis focus field

### 2.2 Existing Robot Programming Methods

A robot can carry out a task repetitively with high precision, high speed, and high endurance. The essence of a robotic task is to plan collision-free motions of a complex mechanical body from a start position to a goal position. This plan should be done in an environment with a collection of static or moving obstacles. The concept itself is relatively easy however it comes with a hard-computational task [1]. Robot movement can be programmed in the 3D space by using robot

software. The robot software on the other hand is used to manipulate objects and tools in the real world through command lines.

The use of controller-specific languages is the original method for programming robots since each robot controller can use different machine languages to create executable robot programs. Symbol-based programming methods have been developed based on the control languages, where a symbol usually consists of one or more common robot functions to represent a robot program in the form of a flowchart.

Lead-through and walk-through programming methods represent the two forms of direct interaction that require users to be present in the industrial robot work environment. Lead-through programming uses a handheld programming device called teach pendant, to control the robot. These two methods typically use a text editor or graphics-based interface to facilitate the storage of robot configurations to be learned and the creation of robot programs.

Offline programming is an alternative method where the user does not need to be present. The programming process is done in a simulated environment where the kinematics and the structure of the robot are modelled out in a simulation environment. The operator can generate desired robot programs through interaction within this simulation environment.

The implementation of the above-mentioned methods is carried out either at the robot-controller level, in a simulation environment, or in the actual robot environment itself. This is clearly shown in figure 2.2. In this section, the mentioned robot programming methods are further analyzed, and their respective advantages and disadvantages are identified.

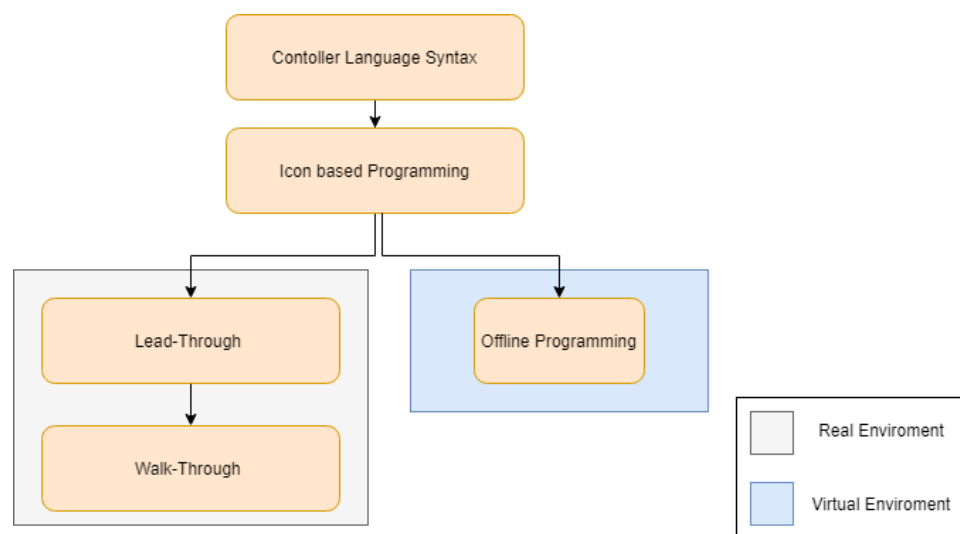


Figure 2-2 Classification of existing robot programming method



## 2.2.1 Syntax or Icon based Programming

Fundamentally, robot software is a set of coded commands or instruction which will be uploaded into the robot controller. This set of commands will be interpreted by the robot controller and execute the appropriate movement of the robot joints to achieve the instructed movement. Each robot manufacture has its robot software. However, the core building blocks of robot software are the data objects and lists of instructions which are known as program flow as shown in figure 2-3 [1].

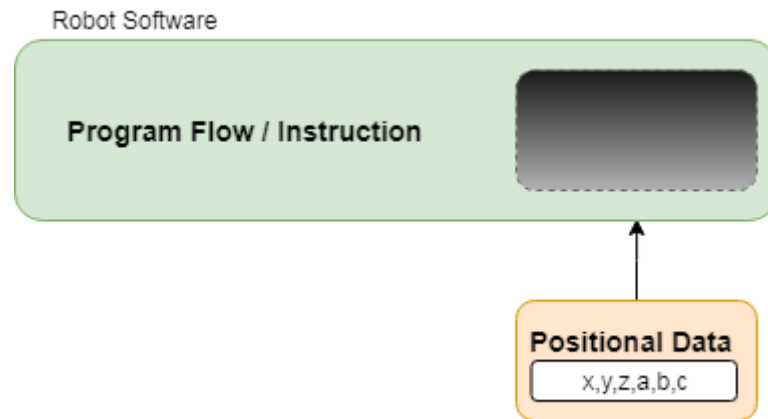


Figure 2-3 Fundamental syntax block of robot program code

Different kinds of robot movement can be achieved by changing the positional data without changing the program flow and vice versa. Robot software is highly proprietary. There more than 30 different industrial robot manufactures which means there are also more than 30 different types of robot programming languages. [2]

Advantages	<ul style="list-style-type: none"> <li>• Low-level programming flow.</li> <li>• More stable and modification freedom</li> </ul>
Disadvantages	<ul style="list-style-type: none"> <li>• Need knowledge and experience in specific controller language of the robot manufacturer</li> <li>• Need to be programmed with less or no graphical assistance</li> <li>• Prone to human-careless mistake</li> </ul>

Table 1 Advantages and disadvantages of syntax or icon-based programming

## 2.2.2 Lead Through

Lead-through or commonly known as teaching pendant is the most popular and adopted method in robot programming. According to the British Automation and Robot Association, over 90% of the robots are programmed using this approach. Due to this fact it is the most familiar form of robot programming for most robot operators. This method normally categorized as online-programming method.

This method relies on the teach pendant hardware which acts as a programming unit. The early generation of the teach pendant was simpler. They were equipped with a magnetic tape system to store the robot programs. Over the year the complexity of the teach pendant has evolved dramatically. It almost becomes a machine itself. Therefore, the robot operator must study the user manual of the teach pendant to start programming the robot with it.

One of the features of the teach pendant is to manually send the robot to the desired goal position and record the current joint states. Afterward, these recorded states can be replayed by the robot and produce the robot's motion again. Furthermore, the different motion speeds of the robot can be chosen with a teach pendant. Generally, the operator activates the "Teach-Mode" in the teach pendant to set the robot in a more secure mode with a reduced speed limit. This is important for precise robot positioning and reliable inspection of the taught path.

Advantages	<ul style="list-style-type: none"><li>• Allow precise and flexible positioning. (The robot can be programmed using numerical coordinate systems, in either world coordinates, robot base coordinates, tool coordinates, or another coordinates system)</li><li>• Easy and quick programming of movement tasks like straight-line robot motion</li></ul>
Disadvantages	<ul style="list-style-type: none"><li>• Test runs after programming and reprogramming in "teach-mode" lead to downtimes of the system</li><li>• Requires expert training and examination to learn and master this programming method</li><li>• Difficult for people who are skilled craftsmen but poor in understanding programming methodology</li></ul>

Table 2 Advantages and disadvantages of lead through methods

### 2.2.3 Walk Through

This method has some similarity to the teach pendant or lead through the approach. However, it is a more intuitive way of programming as the robot operator demonstrates the movement of the robot. These methods involve moving the robot around, either by manipulation of the force sensor or with a custom joystick attached to the robot wrist just above the end effector. After moving the end effector to the desired location, the operator can store the position. This methodology is popularly accepted in many collaborative robots' operations. Since the operator can immediately use the robot for its tasks.

Advantages	<ul style="list-style-type: none"> <li>• Quicker than the traditional teach pendants method. It eliminates the need for multiple button pressing, allowing the operator to simply move the robot to the desired position</li> <li>• It is a more human sense-orientated way of programming. The operator control and adjust the position of the robot by his direct intuition for the task. Which enables a faster learning curve than the other programming methods</li> <li>• No programming knowledge is needed</li> <li>• A very good alternative for detailed tasks that require many layers of code to achieve the same effect, such as welding or painting intricate forms</li> </ul>
Disadvantages	<ul style="list-style-type: none"> <li>• Produce the same downtime in the production line like the teach pendant</li> <li>• Moving the robot to a coordinate with high precision not easy as the other methods. This method is prone to human error for accuracy and repeatability of a task. Some kinesthetic teaching can offer numerical coordinate information input to readjust the demonstrated motion. (But this requires an additional programming layer)</li> <li>• Can lead to operator fatigues and safety issues as direct contact with the robot is necessary.</li> <li>• This method is limited to a small cluster of robots (Cobots) (large and heavy robots can't be easily adapted to this method)</li> </ul>

Table 3 Advantages and disadvantages of walk through method

## 2.2.4 Simulation/Offline Programming

Offline programming or simulation-based programming is often used in advanced task demands. This method allows the robot to be programmed using a virtual mock-up of the robot, robot cell, and the task. With this method, advanced control algorithms can be tested for a correct and fail-safe operation before deploying it into a real robot. This method is gaining popularity and acceptance as the computational resources available in modern times increase exponentially.

This method is desirable in industries where downtime in production is not accepted and where the robot will undergo multiple reconfigurations. The programming process exists outside the production cycle. Furthermore, this method often comes with an intuitive form of the user interface where an idea can be tested and varied in the simulation before moving it to the real robot.

One of the key aspects of this method is the use of CAD models to generate robot trajectories automatically. This a great solution for a task that is heavily object-orientated like rapid prototyping of complex shaped objects with CNC machines. Furthermore, in this method, the precision between simulation and real-world is also a key element. It is much more time-consuming to manually program (Teach Pendant) all the robot trajectories to manufacture or process an object surface with a high level of complexity. However, with a high precise CAD model of that object, the Offline Program can calculate and generate the necessary robot motions to carry out the task fully autonomously.

Advantages	<ul style="list-style-type: none"><li>• Reduces the downtime. The robot operation only needs to halt while the new program is being uploaded to the robot controller and tested</li><li>• The intuitive approach of programming. (The CAD model of a robot in a simulation environment can be moved with drag and drop techniques)</li><li>• An easier and faster way to test many different approaches to the same problem which is time and energy-consuming in Online methods</li><li>• Potentially dangerous movement of the robot and clashed with the environment can be simulated and avoided in Offline programming</li></ul>
Disadvantages	<ul style="list-style-type: none"><li>• The programs still need to be altered with an additional layer of information filters or correction mechanisms to be applied to the real robot</li></ul>

	<ul style="list-style-type: none"> <li>• Depending on the processing hardware the programming process can be time-consuming since it needs to run various simulation and post-processing tasks to generate the final version of the program</li> <li>• Can run into simulator issues that are not related to the production challenges themselves. (Quality and the precision of generated trajectories heavily depend on the simulation set-up and capabilities of the overall Offline software)</li> </ul>
--	--

Table 4 Advantages and disadvantages of simulation

### 2.2.5 Learning System based Robot Programming

Automating the robot programming process utilizing artificial intelligence and machine learning is gaining more and more attention in the field of industry. It opens a new opportunity to increase profits for the manufacturer. Minimum or even no human interventions are needed for the robot to carry out a task in a previously unknown environment.

The main idea of this approach is that machine learning and reinforcement learning system are introduced into the programming loop. This enables the robot to self-explore the environment through sophisticated sensors and learn to accomplish a task based on its data gained from the learning phase.

The modern robotic system incorporates a sophisticated sensor system in its architecture. This data is from, joint force sensors and vision sensors, for example, that can provide continuous and real-time data to be integrated into robot motion planning.

These new capabilities widen the robot programming autonomy in various new fields. Good examples are autonomous varying payload detection and pick and place tasks with computer vision.

Programming by Demonstration with machine learning is another good example. A user performs the task manually, leaving the robot to observe, follow and learn the human demonstrations in real-time. This enables a user who may not have any robotic programming skills to program a robot.

Advantages	<ul style="list-style-type: none"> <li>• Allows a robot to intelligently adapt to unknown situations, tasks, and objects</li> <li>• Enables an autonomous path planning with more efficient and minimum movement of the joint</li> <li>• enables a user who may not have any robotic programming skills to program a robot</li> </ul>
Disadvantages	<ul style="list-style-type: none"> <li>• Still at the research stage and ideal only for the simplistic situation with less variation in environment parameters</li> <li>• Additional effort needed to obtain the sample data needed for learning</li> <li>• Noise in the data collected due to variations in human demonstrations or sensor reading</li> <li>• Multiple demonstrations are required for a higher quality of performance</li> </ul>

Table 5 Advantages and disadvantages of learning system-based method

### 2.3 Pick and Place Robot Task

Robotic manipulation tasks refer to the process of moving or rearranging objects in the environment. To perform a manipulation task, a robot establishes physical contact with objects in the environment and subsequently moves these objects by exerting forces and moments. [2]. One of these robotic manipulation tasks is the 'pick and place' task.

Sometimes a picking up an object can become complicated due to the task scenario. For example, the difficulty is dependent on the form of the object and its physical characteristics. Moreover, the choice of gripper plays a important role.

Gripping with a robot is not as simple as it is for human hands. There are several uncertainties due to the lack of actuation and sensing information. The Gripper cannot feel the object. These uncertainties contribute to computational challenges in motion planning. Motion planning is divided into two parts to simplify the various aspects of the problem into something computationally tractable [2].

- Rough motion planning
- Fine motion planning

The rough approach involves reasoning about the manipulator motion on the macro scale and considers its overall global movement strategy. The fine approach considers how to deal with uncertainty to accomplish the precision that is required by the process robustly. Furthermore, the primary constraints on the allowable motions are:

1. The robot manipulator must not collide with any obstacles in the environment while reaching for, grasping, or releasing the part
2. The robot must adequately grasp the part while transporting it
3. The robot manipulator and movable part must not collide with any obstacles in the environment while the part is being transported
4. If the part is not being transported by the robot, it must be placed at rest in some stable intermediate placement
5. The part must end up at the desired goal location in the workspace

The two modes of operation result in the formulation of the hybrid system. Either the robot moves alone or the robot moves while grasping the part. In this thesis, motions of the robot holding the object are called transfer paths, and motions of the robot while the part stays at a stable placement are called transit paths.

## **2.4 State of the Art Mixed Reality Technology**

In a Mixed Reality scene, computer-generated entities are embedded onto the real-world view to enhancing the user's real-time interaction with the real world. These computer-generated entities can be in the form of texts, graphics, 3D models or animation. The MR technology creates the illusion of virtual objects belonging to the real world. This can be interpreted as the 'hologram effect' or 'magic lens effect' [3].

### **2.4.1 Definition**

The definition of mixed reality needs to be outlined and the synonym between the term augmented reality (AR) and mixed reality (MR) is need to be clarified. Augmented reality is the perception of reality with computer-generated perceptual information enhancement. [3] The term MR is based on a reality-Virtuality continuum as a continuous scale ranging between the real world and a completely virtual environment, similar to virtual reality (VR). The continuum encompasses all possible variations and compositions of real and virtual objects and consists of both augmented reality (AR), where the virtual augments the real and augmented Virtuality (AV), where the real augments the virtual. MR provides a certain area of different combinations of real and virtual content [4]. In this thesis, the term of AR and MR is considered as synonyms because the scope of the thesis does not demand a strict classification between the amount of combination of the real world and the virtual world

## **2.4.2 Mixed Reality Working Principle**

The MR technology system can be broken down into 5 main subsystems to produce the MR scene output [3]:

- 1.Video Capturing
- 2.Tracking
- 3.Registration
- 4.Placement
- 5.Output

### **Video Capturing**

The first step usually involves capturing a video image or more precisely, a video stream of the viewer's surroundings. This is done with any camera (webcam, smartphone camera, TV camera, etc.). The important thing here is that the camera has been calibrated appropriately beforehand. There are other types of augmented reality, for which a camera recording of the environment is not necessary like projector-based AR. However, in this thesis, such methods are not taken into consideration due to high hardware requirements.

### **Tracking**

Tracking is generally understood to be the calculation or rather the estimation of position and orientation. In the case of MR, it is always necessary to capture the viewer's point of view relative to the surrounding environment as accurately as possible. However, since the perception of reality is mostly in the form of the captured video image, one mostly estimates the position and orientation of the camera instead. Tracking provides a transformation from the user or camera coordinate system to the coordinate system of the virtual environment.

### **Registration**

Registration or more precisely, geometric registration refers to the anchoring or correct fitting of artificial virtual content into real environment. This means that based on the position and orientation estimation of the tracking the coordinate system of the individual virtual contents and the observed reality are related in such a way that virtual contents appear firmly located (registered) in reality. This means an artificial object that is not moving in the virtual world also have a fixed location in the real world, independent of the changing viewpoint of the observer.



## Rendering

Based on the transformation resulting from the geometric registration and the respective camera perspective, the virtual contents are rendered. In this process, the recorded video image is correctly superimposed by the virtual content in terms of perspective, which results in the actual augmentation. For the most seamless overlay possible, the resolution and sharpness of the virtual image may also need to be adjusted. As an alternative to superimposing the video image, the viewer's perspective can also be superimposed directly.

## Output

Finally, the augmented video images (or the augmented video stream) are output via a display to which the camera is also attached. This can be a handheld device such as a smartphone, a tablet, or data glasses. In principle, the output can also take place on a separate monitor or using a projection where the camera is not attached to the display unity itself. However, the impression of a seamless extension of reality is only limited to the viewer. The output devices will be further analyzed in the section 2.4.4

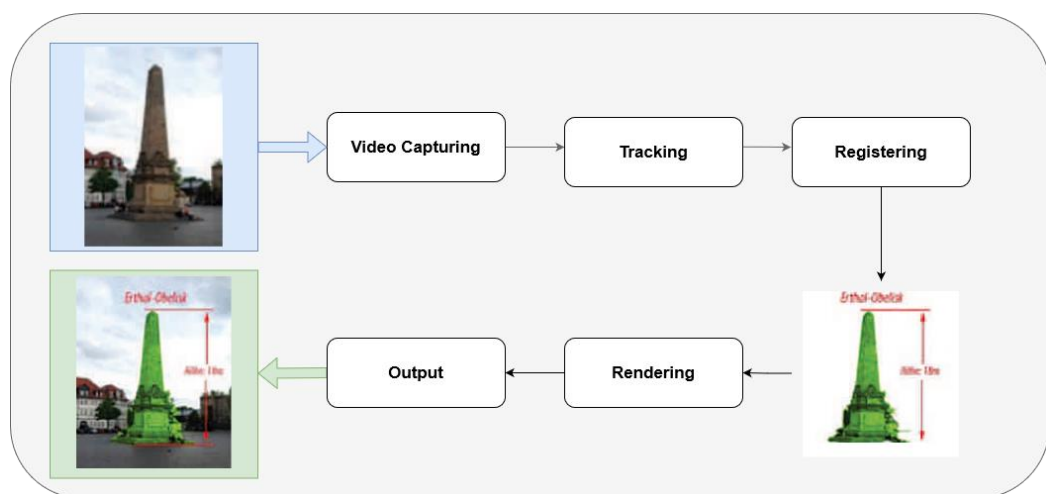


Figure 2-4 The working principle of mixed reality technology

### 2.4.3 Pose Estimation in Mixed Reality

Accurate registration and tracking are the main benchmarking criteria of the MR system.

#### RGB based Pose-Tracking

The basic functionality of the mixed reality is based on a computer vision algorithm where a previously known cluster of points (marker) is being identified in the stream of new visual information. This visual information is in form of the RGB-Data of the camera. The tracking takes place in four steps. If a target is found, the position and orientation of the camera to the marker is calculated from the pose of the corner points. These targets are normally being 2-dimensional image targets or 3-dimensional model targets. Model targets by mean can be simple geometry like a box, cylinder, sphere, or a complicated geometry form like the CAD-model of an object.

The image target and the object target in an MR ecosystem work with this algorithm. It continuously searches for a predefined reference image target or object target. From this reference target, unique elements are extracted out, like the high-contrast spots, curves, or edges that will not be altered significantly when it is the view from different angles. This unique element is recorded as the feature points in the algorithm. The main criteria for a good target area to have plenty of such feature points in it. The target recognition technology saves the relative's positions of all the feature points with each other. In return, it has a database of the shape of constellations of these feature points.

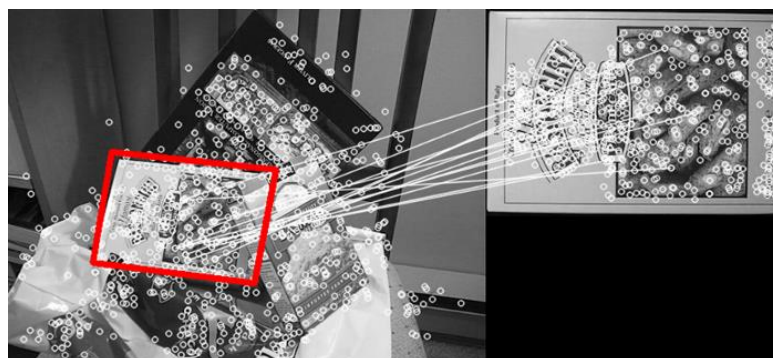


Figure 2-5 Recognition of feature points from RGB data [5]

During an augmented reality session, the algorithm continuously searches for any resembling cluster of feature points that resembles the shape of the constellation on every RGB-frame of the camera feed. If most reference features points are

registered in the camera frame, then this path of the camera frame is recognized as the marker. The 6 degrees of freedom pose of the marker is calculated by comparing relative positions of a reference "constellation" with recognized "constellation". [5]

## **Hybrid Solution based Pose-Tracking**

This solution is a combination of the IMU readings from sensors and RGB pose tracking. Sometimes it is referred to as the visual-SLAM. Position estimation in mobile environments with the aid of sensors is now considered standard in the field of mobile devices such as smartphones and tablets. Usually, a combination of three different sensor types is used which are magnetometers gyroscope, and inertial sensors. With this hybrid solution, the quality of the tracking is profoundly increased. Furthermore, the tracking marker pose is not broken even though the marker itself is not in the camera viewpoint anymore.

## **Machine Learning-based Pose-Tracking**

This is a new form of object pose detection method which utilizes a ML algorithm to predict the pose of the target, which can be an image target or an object target. One of the advantages of this pose tracking method is that one or more objects can be recognized and tracked from all angles. This enables a 360-degree viewpoint range utilization of the MR application [6]. This feature is not available in the mentioned image target or object target because those targets only work at certain viewpoints where the key features recognizable and visible.

### **2.4.4 Mixed Reality Output Devices**

The mixed reality output devices are used by the user to experience the MR environment. These devices can be broken down into two major categories:

1. Handheld devices (smartphone, tablet, and laptop)
2. Head mount displays

These categories arise due to the two different type of MR experience each device deliver to the user. These MR experiences are further explained in the following points:

#### **Video See-Through-MR (Magic Lens Effect)**

In the video see-through technique, first, the real environment is captured using a video camera. The video image is then superimposed with virtual content in the correct perspective and then dispalyed on an output device.

This is called the magic lens effect. The viewing direction and viewing angle of the video camera and the output (the display unit) matches. It gives an impression of a see-through window or portal which enhances the view of the reality with additional virtual objects and animation. This display method is promising and easily implementable, as the number of tablets and smartphone which are being used is increasing exponentially. Furthermore, most of these devices are equipped with high-performance capable CPUs, GPU, and IMU sensors. This sets a very promising scenario for a huge scale deployment of an MR application.



Figure 2-6 "Magic lens" type MR effect done with handheld device [4]

### **Optical See-Through-MR**

In contrast, a video recording of the real environment is not necessary for so-called optical see-through MR. Instead, the real environment is always perceived directly by the viewer. For this purpose, virtual content is optically superimposed on reality by the output device. This requires an output device with a semi-transparent display so that on the one hand the reality behind it and on the other hand, the additional virtual content can be perceived. For the perspective of the real environment and the virtual extension to match, the viewer's point of view with the display must be known. It is necessary for each eye to have a separate display. If both eyes look at the same display, a stereoscopic display can be used so that the perspective can be correctly adjusted for each eye.



Figure 2-7 HMD Device example: HoloLens 2

### **2.4.5 MR Development Approaches**

The MR technology has matured over the past on both the hardware and software sides. Currently, there are many software development kits (SDKs) to start developing an MR application. Most popular are ARCore from Google, ARKit from Apple, and Vuforia from PTC. These SDKs enables a developer to design and deploy an idea of an application that needs to make an object appear to blend into the real world. These SDKs offer all the key functional components like 3D object tracking, image recognition and visual SLAM.

## **2.5 Related Works on Mixed Reality for Robot Programming**

In the following paragraph, projects which are related at some degree to these thesis focus field are studied and analyzed. Some of these works are closely related while some only in a similar environment, i.e. only distantly related. The works are broken down into several clusters to make the analysis more systematic.

### **First Wave: AR Stylus Concepts**

There is a common trend that can be observed from the survey of work done between the periods of 2006 till 2016. The first wave occurred when the AR Toolkit library was introduced in 1999. This library empowers an AR-marker based method. The basic idea was to design a handheld device with one or more AR Markers attached to it. This handheld device is used as the path planning tool to draw the trajectory line. This drawn line is converted into G-Code using standard inverse kinematic modules. This can be seen in the work of Chong et al. [7], Fong et al. [8], Pai et al. [9]

In the work of Fong et al. [8] and Chong et al. [7] , an immersive robot programming method was developed in AR where the waypoints can be programmed using a handheld probe with an AR marker and a sphere at the end. The authors used a video-based tracking method in AR Toolkit and rendered a virtual robot model on the real world with a see-through head mount display. The

author developed the system with an inverse kinematic module which will plan the path for the end effector to reach the waypoint. This work shows that by using a pointer with a sphere attached at the end of it, a smooth arc waypoint teaching can be programmed more intuitively than the traditional way. This is especially useful for welding tasks on an object's surface. The overall system had difficulty in the accuracy of the inverse kinematic module, and it is limited to control of a virtual robot only.

In the work of Pai [9], a CNC machine operating system is developed in the AR scene. Here a desktop monitor and an external camera is used to achieve the 'magic lens' effect. As for the AR tracking and registration, the same AR Toolkit and OpenGL method is utilized. The actual robot is not controlled directly in this application and only a virtual robot model is programmed to move based on the G-Code generated. This AR Toolkit approach achieved good performance with a mean tracking error of 2,78 mm. This value includes errors from human hand jitter, lighting condition, and marker design.

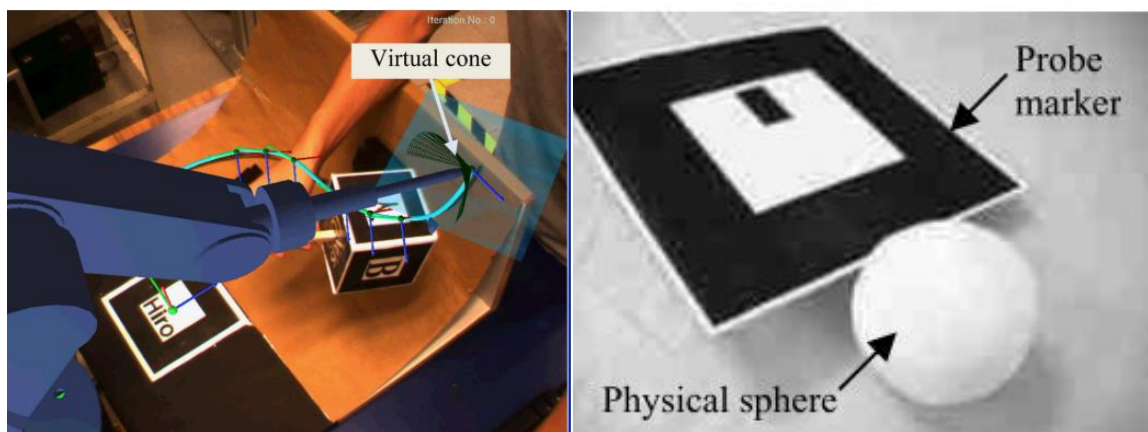


Figure 2-8 Marker-based AR stylus to demonstrate the robot path line. [4] [6]

## Second Wave: HoloLens and Mature Computer Vision Library

For the start of the second wave is when the market launch of the HoloLens in 2016 can be considered. MR make it possible to place a digital entity within the environment and thus boosted the number of researches done in this field. HoloLens has solved the limitation faced in the first wave configuration where tablets and laptops were often used to display AR content which prevented two-handed operations. The HoloLens enables the user to walk hand free and the projection of the AR content is in line with the operator's perception. Furthermore, the maturing of computer vision algorithms alongside the development of hardware and sensor capabilities played a vital role in these works. This can be seen in many scientific papers [10], [11], [12], [13],[14]. [15]

## Accuracy of Pose Estimation

From the research, the accuracy of any type of MR application, SDK, or even equipment is hard to be pin down to an exact dataset. This is because there are multiple numbers of influencing factors that affect the accuracy performance of the system. For example, the lighting condition, the camera calibration, the quality of the target, the distance of the target, device motions and the most uncontrollable variable is the human interference itself.

Form the previous research some benchmarking accuracy of the overall applications which have been developed, are stated in table 6. Some of this benchmark is influenced by the distance from the camera and the viewing angle of the target, the hardware set-up and also the type of AR algorithm which has been used. The author Blankmeyer et al. [11] have found out that the best viewing angle of a 2 Dimensional image target is around 40°- 60° at a distance of 80 cm which yields an accuracy of 1mm to 2mm. In their investigation Vuforia was used as SDK to register and track an image target.

HoloLens uses Iterative closest point (ICL) from the point cloud library (PCL). ICP (Iterative Closest Point) is the default method used by the HoloLens to anchor the digital content in the scanned environment. The HoloLens scan the environment through the in-built depth cameras and generate a point cloud map. However, this approach with the use of additional targets or the usage of a matured AR SDK outputs a more desirable accuracy performance. Previous works utilized this approach because the HoloLens and an Image Target combination yield better performance. However, the use of an Infrared marker produced the highest accuracy as this kind of marker eliminates the environmental lighting condition and target visibility. The following table 6 shows results of the experimental results.

<b>Hardware and Registration Algorithm</b>	<b>Accuracy of target registration from 50 cm distance [mm]</b>
HoloLens 1+PCL-ICP* [16]	320,00
HoloLens 1 + Infrared Red Marker [17]	0,76
HoloLens 1 + Vuforia Image Target[11]	39,3
RGB-Camera+ Vuforia[16]	30,00

Table 6 Accuracy performance of various MR-ecosystem

Furthermore it is stated in [12] and [11] that a continuous target recognition and tracking eliminates the error of model drifting form the initial registration pose.

They concluded a continuous registration is better than a one-time registration algorithm although it consumes more computational resource.

## **VR vs AR**

The advantage of using VR-based robot programming instead of AR is its scalable modeling capability of the entire environment where a robot operates. However, an accurate VR environment requires dedicated content development to represent the actual working scenario. Another issue that needs to be addressed is the delay between the VR display of the movements of a remote robot and its physical movements

VR is mainly used in the focus field of virtual training for example how to operate a CNC machine [9]. The focus here is that the user experience with the machine should match almost exactly the interaction with the actual machine. For these additional reality simulation functions, visual effects and additional external sensor to replicate the reality are needed. This configuration is far less demanding compared to an AR scenario. The overall tracking performance obtained from a hardware set-up of a VR Headset and a Leap Motion device in the work of Cousins et al. [18]., is at an error of  $\pm 3 \text{ mm}$ . In this project, the authors tried to control a Baxter robot remotely using a VR approach.

## **2.6 Laboratory Environment**

Fraunhofer IFAM has allocated a typical industrial scenario for this project as lab environment for this thesis. These place is part of a huge industrial floor where the lighting condition is controlled by heavy industrial light systems. Furthermore, the working space is shared with various types of industrial robots, machinery, and inventory closets.



## 2.6.1 Universal Robot 10

Fraunhofer IFAM has allocated Universal Robot 10 for the work of this thesis. UR10 belongs to the robot class called Cobot. The key metrics of the UR10 are listed below:

Degrees of freedom (DOF)	6 rotating joints
Maximum payload	10 kg
Reach	1300 mm
Speed of tool, typical	1 m/s
Repeatability	$\pm 0.1$ mm
Materials	Aluminum, ABS plastic, PP plastic

Table 7 UR10 data sheet

The UR10 has in total 6 Joints. The figure below shows the labels given to the main components of the robot. The UR10 is set up on top of a specially designed table and connected to a power supply.

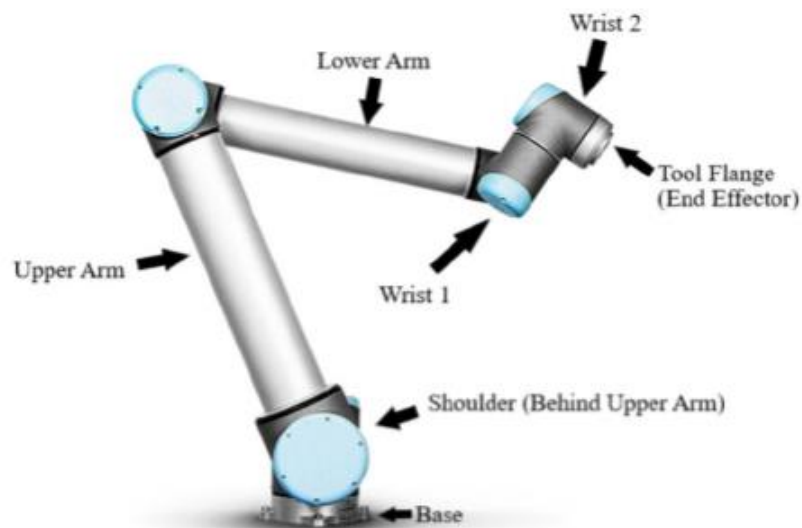


Figure 2-9 UR10 joint designations

## 2.6.2 Workstation PC

For this thesis Fraunhofer IFAM has provided a workstation pc with the following system specification mentioned in table 8:

Operating System	Windows 10
processor	Intel Core i7-7700K (4.20GHz)
GPU	NVIDIA GeForce GTX 1080
RAM	16 GB
Camera	Creative HD Webcam and depth camera D435

Table 8 Workstation PC system specification

## 2.6.3 Unity IDE

For this thesis an integrated development environment (IDE) form Unity is provided by Fraunhofer IFAM. Unity IDE is a cross-platform game engine where the applications are developed under the C# language. The platform is also used for engineering platform as it provides an easy and flexible developing environment for highly programmable simulation engineering and software engineering. The unity project guideline for this thesis is stated in the appendix B.

## 2.7 Summary

From the gathered information, it is clear that an existing robot programming method is either in the virtual environment or in the real environment. A method that exists in the combination of the virtual and real environment is still under research level and hasn't been introduced as a reliable method yet. The mixed reality technology can act as a starting point for this method.

The development of the idea is alive and actively researched upon in this decade. It can be also seen that the development process is closely related to the maturity of computer vision technology, the available MR devices, and the easy programming of the robot controller. These three factors act as t catalyst for the development of the concept.

The fundamental concept of MR technology is outlined and the possible development methods are identified. The selection of the type of target plays a vital role in the overall accuracy of the system. The accuracy factor is a very important aspect of robot programming. Therefore, the most optimum and ideal target configuration needs to be identified in the development process. The infrared marker gives the most accurate performance. However, an additional layer of hardware set-up and algorithm needs to be deployed compared to the image target approach. Due to this reason this set-up is not taken into consideration.

The given laboratory set-up is outlined and taken as the boundary conditions for this thesis. A concept that can be deployed under this boundary condition will be developed in the following chapter.

Finally, to contribute a better solution than the previous work, the focus should be fixed on the intuitiveness the practicality of the overall application. This means the minimum usage of hardware and straightforward implementation of software algorithms will promote a better advancement of the system.

## 3 Concept and Design

This chapter presents the concept and design development for the mixed reality based robot programming application. The main goal of this MR application is to intuitively enable a user to perform a pick and place task with a robot. The workflow of this chapter is illustrated in the following figure 3-1.

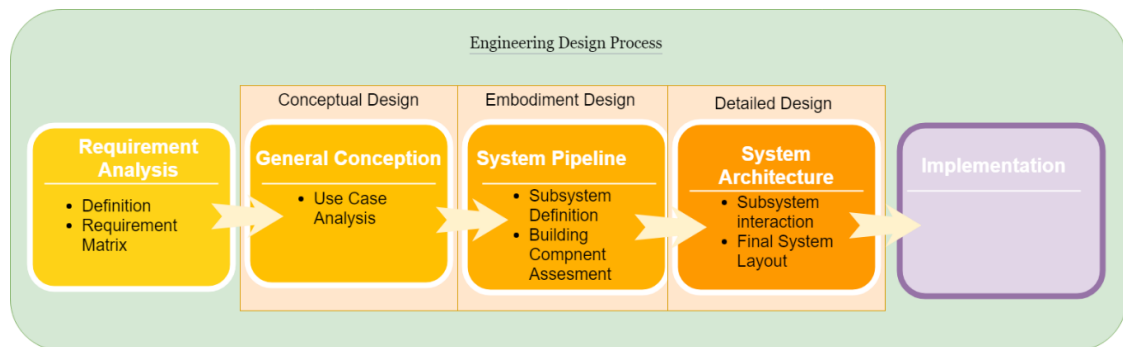


Figure 3-1 Workflow of the design process

### 3.1 Requirement Definition and Analysis

In this section, the design requirements are formulated. The primary requirements that need to be fulfilled are given from the Fraunhofer IFAM. The secondary requirements are derived from the state-of-the-art analysis and previous work insights in chapter 2.

The primary requirements 'must be fulfilled' by the solution to conduct a valid experiment process and derive valuable insight. The secondary requirements 'should be fulfilled' if it is possible within the time scope. Tertiary requirements 'can be fulfilled' but are of very low priority. They can be neglected if they involve a lot of unnecessary work.

These design requirements act as boundary conditions and they control the design of the product and the process being developed in chapter 4. They will also be used as input in the verification process in chapter 5.

#### 3.1.1 Primary Functional Requirements

These requirements are derived from the main goal of this thesis which is set by the Fraunhofer IFAM. Fraunhofer IFAM acts as the primary stakeholder of this work. One of the goals of this thesis is to develop a mixed reality-based robot programming method. Using only this application a pick and place task should be carried out with a UR10 robot.

The performance of the final application can be measured and analyzed with an experimental approach. Based on chapter 2.3 which explains how a pick and place task is performed with a robot, the following primary requirements are formulated and given identification tags in table 9.

A	Primary Functional Requirements
A1	The MR application can set more than one waypoint just like in the teach mode
A2	The MR application can record waypoints
A3	The MR application can initiate the robot to follow the recorded waypoints
A4	The MR application can activate the gripping end effector
A5	The MR application can assist a successful grasp of picking an object
A6	The MR application deactivates the gripper end effector

Table 9 Primary functional requirements

### 3.1.2 Secondary Functional Requirements

The following requirements are derived mainly from the analysis part of this thesis. These requirements lead to the development of a MR application that provides the user an intuitive and easy form of robot programming especially for a pick and place tasks. This is important as one of the main goals of this thesis underlines the end application should promote an easier and intuitive form of robot programming. From chapter 2.2, the existing main robot programming methods are identified and analyzed.

The advantages and disadvantages of each robot programming method are listed out in table 10. In fact, MR technology integration does play as a good alternative method of robot programming compared to the existing methods. This is because the MR application does give a solution to some of the disadvantages faced in the existing methods like a high complexity of the programming workflow, a high demand for programming knowledge and a non-flexible programming environment. The following table 10 lists the possible potential gains which can be achieved from developing a MR based robot programming method and the degree of potential for a solution is illustrated by a color gradient. The dark green illustrates more gains can be achieved through integration of MR technology and vice versa for the light green.

Existing Methods	Disadvantages	Potential Solution from MR based Robot Programming
Teach Pendant	The 'Teach Mode' contributes to downtime	Offline-based MR programming approach
	Programming methodology Knowledge is required	MR-UI which promotes easy understandable and intuitive programming steps
Teaching by Demonstration	Moving the robot to a point with high accuracy is not easy	MR solution which gives live feedback on the point coordinate which being demonstrated
	Promote operator fatigue and danger of direct contact with the robot	Solution which is able to manipulate the End-Effector location without touching it
Offline Programming	Remodeling the Robot work cell is time-consuming and difficult	Implementation of the simulated robot in the real environment and the environment does not need to be remodeled
	The robot work cell needs to be reconstructed if there is physical changes and this slows down the simulation process result	Direct and instant feedback on how the robot interact within the environment
Autonomous Robot Programming	High level of computer programming knowledge	Human cognitive skills like vision-based judgment and sensory organ-based judgment can easily be introduced in the programming loop
	Able to set up and design sophisticated sensor architecture	MR solution with a Human-centric computation approach

Table 10 Possible advantages of MR based robot programming method

From this table 10, it can be seen only some of the disadvantages which are associated with programming by demonstration method and the teach pendant method, could be directly fixed using the integration of the mixed reality technology. The mixed reality UI layer will act as an additional layer on top of this Programming by Demonstration (PbD) method. Normally PbD is executed with a human touch contact or touch panel interaction. With this mixed reality solution, both of these interactions should be made optional. However, the robot waypoint can be taught without being touched and the user does not need to know how to use the touch panel to record the waypoints.

This is important to achieve the intuitive form of robot programming. This approach has already minimized the number of steps that need to be performed to achieve the same goal. Furthermore, the mixed reality UI layer should prompt the user with useful and insitu relevant information to assist and simplify the waypoint teaching. It also needs to be mentioned that through a PbD method the user is constantly learning the art of programming with various forms of possibilities. This also enables room for innovation and creativity for robot task and not restricted to the limited information projected on a 2-dimensional screen.

Furthermore, the MR Application should perform a pick and place task. In chapter 2.3, the overall process of a pick and place task with a robot is outlined. Successful completion of a pick and place task is also defined in this chapter. The simplest way to execute a pick and place task is using a teach pendant method approach. This methodology is very similar to the programming by demonstration. Each waypoint is taught in by the operator in the teach pendant until the end effector with a grasping mechanism, is in a suitable position to grasp an object. This scenario is taken as a benchmarking as it is easy to mimic and has lesser influencing factors around it.

Normally the quality of the pick and place task performed in this manner is dependent on the user's perception of the world. Unlike the learning-based approach or computer vision aided approach of a pick place task which utilizes sophisticated sensory data to produce the pick and place action autonomously. However, PbD does need a lot of pre-calibration and complicated hardware set-up and software configuration.

Therefore, the MR application will be measured against the most simplistic approach of executing a pick and place task with a robot, which is the teach pendant compared to the other approaches. This is illustrated in figure 3-2. With this compression, the qualitative attribute of intuitiveness of the MR based robot programming methods, against the teach pendant method to perform a pick and place task could be easily validated based on features of PbD.

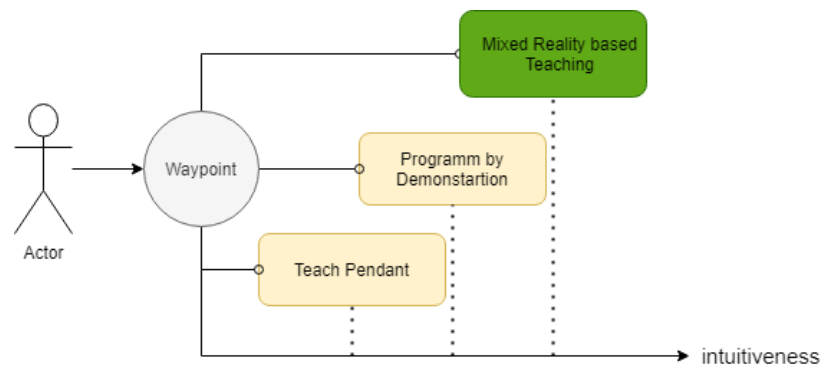


Figure 3-2 Intuitiveness level associated with robot programming methods

In the table 11, the secondary requirements are for this concept is outlined. These features are chosen as secondary requirement because their core functions are related to guiding the user to perform the pick and place task more efficiently. They also provide the flexibility and robustness to the MR base robot programming approach. However, the pick and place task can be done even without these features and for this reason these requirements should be present within the system but not be taken as the main requirement. This is done for a better process of implementation of the system.

B	Secondary Functional Requirements
B1	The MR application can delete the recorded waypoints
B2	The MR application can change the location of waypoints
B3	The MR application can change the pose of the waypoint in all 6 degrees of freedom
B4	The MR application show path line from the constructive waypoints
B5	The MR application shows the individual waypoint Tags
B6	The MR application can show the distance between each constructive waypoint



B7	The MR application can show or illustrate a preview of the robot moves along with these waypoints
B8	The MR application can show an updated preview of robot movement when one or more waypoint is deleted
B9	The MR application can show an updated preview of robot movement when one or more waypoint's pose is modified
B10	The MR application can remove all the recorded waypoints with one action
B11	The MR application can make a robot execute the same movement from the recorded waypoints

Table 11 Secondary functional requirements

### 3.1.3 Non-Functional Requirements

The non-functional requirements define system qualities. These requirements also impose constraints or restrictions on the overall design of the system. These requirements should not be overlooked as it affects the overall evaluation of the system in the final stage of the development. However, these requirements are set as the tertiary requirements for this thesis. If any of these requirements can't be fulfilled due to technology or resource constraints, it can be omitted as long the primary requirements are not being affected by this action.

Most of the non-functional requirements are also derived from chapter 2. The analysis of the state of the art technologies involved in this thesis provides a framework of technology limitation and fundamental performance attributes. The analysis of the previous work related to this thesis focus field provides the key features that the system should have and the mistakes which should be avoided in the designing section. The types of non-functional requirements which are identified for this system are clustered as follows:

<b>C</b>	<b>Performance (Non-Functional Requirements)</b>
C1	The MR application provides additional calibration setting to improve the MR experience
C2	The MR Application should start up within 60 seconds after initiation
<b>D</b>	<b>Scalability (Non-Functional Requirements)</b>

D1	The MR application can be deployed in UWP, Android, and iOS platform
D2	The System can be easily installed deployed into another hardware in a different location
D3	The system supports multiple users to control the robot
<b>E</b>	<b>Reliability (Non-Functional Requirements)</b>
E1	The MR application able to complete 10 cycles of operation without malfunctioning
E2	The MR Application could produce a robot movement with an error margin of 0,1 mm

Table 12 Non-functional requirements

## 3.2 General Conception

In this section, the general concept of the system will be introduced. This general concept should be in alignment with the requirements of the system. The general conception is a high-level idea that describes how these requirements are going to be satisfied. The general concept does not pay attention to the technical aspect of the idea in detail. However, it sets the flow towards the detailed development of the concept.

### 3.2.1 Programmable Digital Twin

A digital twin is a virtual entity that has the same characteristic as the real counterpart of it in the real world. Digital twin concepts consist of three parts:

- The physical product
- The virtual product
- The connection between these two products

The virtual product is a digital mirror of the real product in the virtual environment. The connection part handles the data transfer of the real product to the virtual product and the transfer of the available information from the virtual product to the real product. These data are obtained from sensor reading which is integrated into the real product. The information received from the virtual product to the real product can be the output generated from processing the sensor data with certain algorithms.

This concept can be integrated as part of the robotic programming system with the help of the additional MR layer. A digital twin of the UR10 robot can be used to view all the joint states in real-time. This can be done by collecting the data streaming from the sensors on the UR1's joints and projecting this information on a 3D-Model of the UR10.

These real-time sensor readings can be used to update the digital twin robot state if there are any movements performed in the real robot. With this approach, a digital form of the UR10 robot, which mimics the exact movements of the real counterpart is set to be available.

In order to integrate the robot programming part in this concept, the real robot should be made to update its current state when there is any form of movement in the virtual robot. When the joint configurations in the virtual robot are changed in the virtual environment domain, there should be a way for it to be reflecting into the real robot. There should be an additional subsystem that captures the changes in the joint state of the virtual robot and generate the necessary G-Code for the real robot to change its state in the same way. A G-Code is a native language of the robot control.

This concept already exists in the offline robot programming method. Where the real robot can be controlled indirectly by manipulating its virtual twin model in an offline simulation environment. However, this process has a clear borderline between the real world and the virtual world (offline simulation ecosystem).

The mixed reality technology can act as the bridging block to merge these two environments into one, where the human operator has a new form of control over the robot by having the ability to change the virtual robot state directly. Through mixed reality technology, the digital twin robot can be projected to the real environment and can interact with it too. Furthermore, matching the placement of the virtual twin on the exact pose of the real twin can yield the superimpose effect. This effect promotes the user an illusion of a programmable digital twin.

The concept of the programmable digital twin robot is further refined in the following steps:

- Step 1: The digital twin robot should be registered on the exact location of the real robot to achieve a superimposition effect
- Step 2: The registered digital twin on the real robot location can be seen by the user. This where the mixed reality rendering method (refer to chapter 2.4.2). can be used
- Step3: An interactive GUI based tools is prompted to the user to manipulate the state of the digital twin robot

- Step 4. A method to command the real robot to mirror back the digital twin's new state is implemented

In figure 3-3, the basic idea is outlined, and some concrete components needed for this solution are identified.

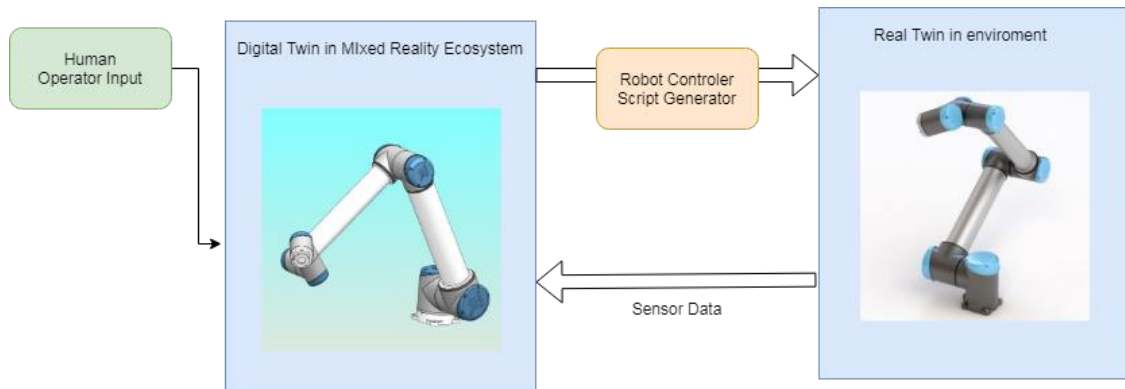


Figure 3-3 Programmable digital twin concept

### 3.2.2 Coordinate System Descriptions

There are three main Euclidean coordinate systems that should be taken into consideration.

- The UR10 robot base
- The camera sensor used for the MR application
- The end-effector of the UR10

The interrelationship between these three-coordinate systems should be outlined. The picking mechanism takes place at the end-effector of the manipulator. Therefore, having control of transforming the location of the end effector within the working space enables to perform a pick and place task with a high degree of freedom. This concept is similar to kinesthetic teaching, (lead through programming method). However, the operator does not need to physically move the end-effector to the desired location. By enabling the user to manipulate the end-effector of the digital twin of the robot through the mixed reality layer, the user can make the robot move to a specific location. Therefore, the transformation between the previously mentioned coordinate system needs to be understood for the application development phase.

First of all the relationship between the UR10 robot base  $(X_R, Y_R, Z_R)$  and the end-effector  $(X_E, Y_E, Z_E)$ . are obtained from the internal kinematic model of the UR10 controller. The actual position of the end-effector is saved and updated in real-time as Tool Centre Point (TCP). Secondly, the coordinate system of the camera sensor of the MR application  $(X_C, Y_C, Z_C)$  is obtained respectively to an MR target

which can be recognized in the reality. The pose of the camera is constantly updated with the respective transformation between the registered and tracked marker.

In order to achieve a superimposition of the digital twin and the real robot, the UR10 robot base  $(X_R, Y_R, Z_R)$  should be registered and tracked with the means of the MR target. This completes the loop between the three coordinates system. Therefore the relationship between the camera sensor of the MR application  $(X_C, Y_C, Z_C)$  and the end-effector  $(X_E, Y_E, Z_E)$ . can be calculated.

### 3.2.3 Integration of Robot Operating System

For the development of a UR10 digital twin concept, the following things need to be available:

1. A very accurate 3D model of the UR10
2. Access to the real-time joint states of the UR10
3. The capability to alter the UR10 joints state

The mentioned requirements for the development of the digital twin concept can be fulfilled with middleware called Robot Operating System (ROS). ROS provides a collection of software frameworks for the development of robotic applications.

In the digital twin concept, a subsystem is needed where the path that the robot needs to take to reach a waypoint should be generated. This generated path should be transferred to the UR10 controller system to be executed. Furthermore, the sophisticated robot path generation needs high computational resources for a faster output of the script. Separating the path generation algorithm from the MR ecosystem will enable a better user interface performance on the MR interface side. This can be achieved with the ROS approach where all the necessary robotic-related algorithms can be run on a separate CPU.

The ROS has basic architecture as shown in figure 3-4, A ROS master layer act as a server that can run ROS nodes. A ROS node can be a specific robot program algorithm, a communication protocol, or a graphic display system. They are running under C++ or python script. The communication layer is consisting of custom ROS topics that can be published and subscribed among the running ROS nodes. This is the fundamental idea of the ROS system where a complex robot program can be implemented through a various stack of ROS nodes and ROS topics.

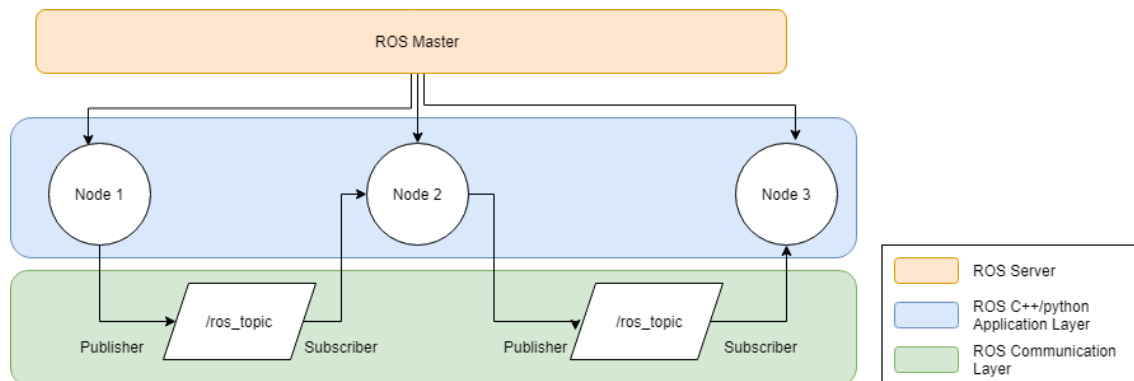


Figure 3-4 Basic ROS system architecture

The ROS online community is a very active platform where many kinds of frameworks for robot application nodes are made available to the public. These nodes' applications are compiled with their dependencies and source codes in form of ROS packages. These ROS packages can be distributed in public with open-source licensing. These ROS packages can be downloaded and integrated into the robot software framework in form of ROS nodes. Some of these packages are developed and maintained by well-known industries like Siemens, Fraunhofer, and many more [19]. Based on the survey on the ROS system library, the following ROS packages are identified and fixed as building blocks for the system. The working mechanism and implementation techniques of these packages will be further explained in chapter 4.

## ROS Sharp Package

ROS Sharp package is an open-source software library and tool. It is developed in C# and C++ languages. The main function of this package is to enable bi-directional communication between a ROS ecosystem and the applications which are being built in the Unity IDE, namely .Net application [20]. The author of the ROS Sharp package also has made it possible for retrieving information directly from the ROS ecosystem like the actual joint states, end-effectors pose, and robots movement velocity. Vice versa, the information like user input from the application developed in Unity can be passed to the ROS ecosystem.

This package can be used to open the communication channel between the ROS server and the MR application. This package enables the development of an MR application specifically targeted for the ROS software framework. Furthermore, ROS Sharp also provide a URDF model importer.

## **ROS-Industrial Universal Robot Package**

This package is part of the ROS-Industrial program, where the ROS framework is extended to industrial relevant hardware and applications [21]. It contains packages that provide nodes for communication with Universal's industrial robot controllers which includes the Universal Robot 10. ROS-Industrial takes to consider the low-level reliability and safety of industrial robot controllers. In return, the developer has an industrial graded application to be launched under the ROS server.

Furthermore, this package also provides universal robot description file or Unified Robot Description Format (URDF) models of all the Universal Robots. The URDF model is an XML specification to describe a robot's kinematic and dynamic, visual representation for robot with part's CAD files and collision model of the robot This means an accurate UR10 URDF model can be accessed from this package readily. This URDF model could act as the digital twin on the MR UI layer. It also provides direct controlling of the actual Universal Robot through a command algorithm written on the ROS Application layer.

## **ROS-Movelt Package**

Movelt is a special ROS package that enables the motion planning of a robot manipulator. It can generate robot trajectories path through motion planning plugin which can be either out of the box or custom made [22].

Furthermore, the ROS-industrial package for Universal Robot provides a ready-made and industrial graded Movelt add-on. With this add-on, the full implementation of Movelt for UR10 can be done along with an industrial graded and tested, Movelt-universal robot motion planning plugin. Through the Movelt package, the path planning to reach a specific end-pose could be achieved without the need for the teach pendant. This autonomous path generation capability complements the digital twin concept of this thesis.

### **3.2.4 Pick Object and Gripper Type Selection**

In this section, the type of gripping end-effector and the object that going to be picked is discussed in terms of its physical characteristics. A suction cup gripper is made available by Fraunhofer IFAM for this project as shown in figure 3-5. The vacuum provides the needed normal force to have enough static friction to lock the part in place. A constant supply of vacuum sources is attached to this gripper to make it work. This selection gripper also has practical advantages to the grasping gripper categories. The mechanism is simple and requires only two calling functions in order to utilize it, namely, turning on the vacuum source and turning off the vacuum source. The simpler grasp gripper also has similar two main calling functionality however additional adjustment between the object form

and the gripper hand needs to be programmed well for a successful pick procedure. Furthermore the suction gripper has an adjustable spring element in its vertical axis with 5 mm offset. This mechanism offers a safe and flexible surface contact, if the gripper is moved to far down in the z-axis direction.

A successful pick with a suction gripper only happens when a full contact between the suction cup and the surface is established. Therefore, choosing an object with a large plane surface is optimum. For this thesis, small box-shaped car models are chosen as objects to be picked as shown in figure 3-5. This model is handmade and has a large surface area where the operator can navigate the suction gripper to perform a successful pick with the least amount of effort.

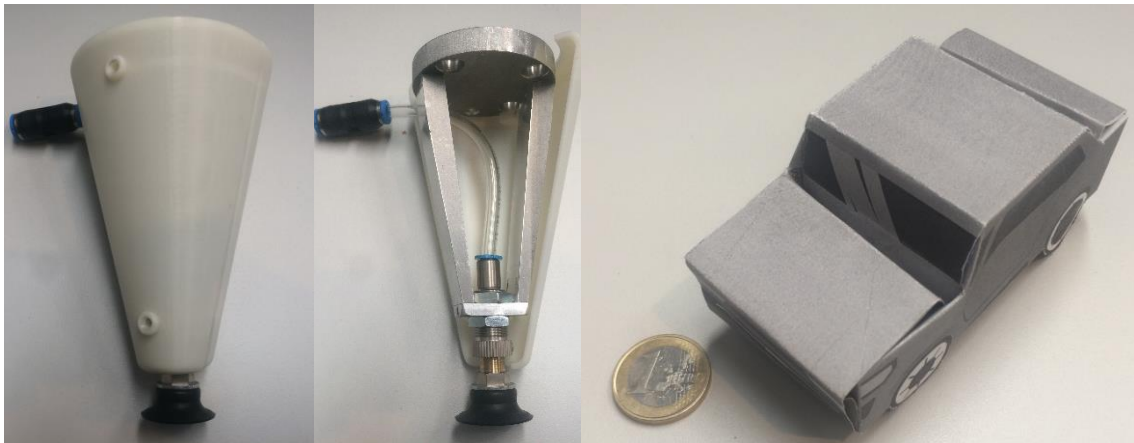


Figure 3-5 Suction gripper structure and the pick object

### 3.2.5 Pointer Type Selection

The user can teach in a pose of the end-effector with a pointer mechanism. The pointer will assist the user to translate his desire of pose in the real world. The decision between a real pointer and a virtual pointer needs to be taken. The table 13 shows the advantages of using a virtual pointer instead of the real pointer.

	Real Pointers	Virtual Pointers
Hardware	Monochrome object, MR Target	None
Technique	Additional MR Tracking system	None
Manufacturing	3D-Printing, CNC	None

Table 13 Selection between real pointer and virtual pointer



### 3.3 Use Case Study

The use case is constructed such all the primary requirements of the system are fulfilled at the end of the whole process. This use case is developed in reference to programming by demonstration method namely the kinesthetic teaching. The mixed reality UI layer will act as an additional layer on top of this PbD method. The MR layer can minimize the number of steps that need to be performed to achieve the same goal and prompt the user with useful and insitu relevant information to assist and simplify the waypoint teaching. The effect of this PbD simplification is illustrated in the use case diagram in figure 3-6.

After a successful connection with the ROS Server, the user should be prompted with the digital twin of the UR10 robot. This digital twin should be already overlaid on top of the existing robot with the exact joint states. In the same MR interface, the user should be shown an instruction to move the end-effector of the digital twin just like in the kinesthetic teaching. The user should be shown a quick preview of the robot movement when the end effector is moved to another location. If the user is satisfied with the robot's movement, the operator can execute the movement in the real robot. By following these steps, the operator can bring the end-effort of the robot, which in this case the suction gripper, close the pick object which is situated anywhere in the reach of the robot.

After the gripper is binged close enough for a successful pick procedure, the vacuum source can be activated through the MR UI layer. Then the operator can repeat the process of changing the end-effector position to a new place, for example, the pacing bin. Upon the arrival of the end effector at the desired end goal position, the operator can deactivate the vacuum source and place the picked object.

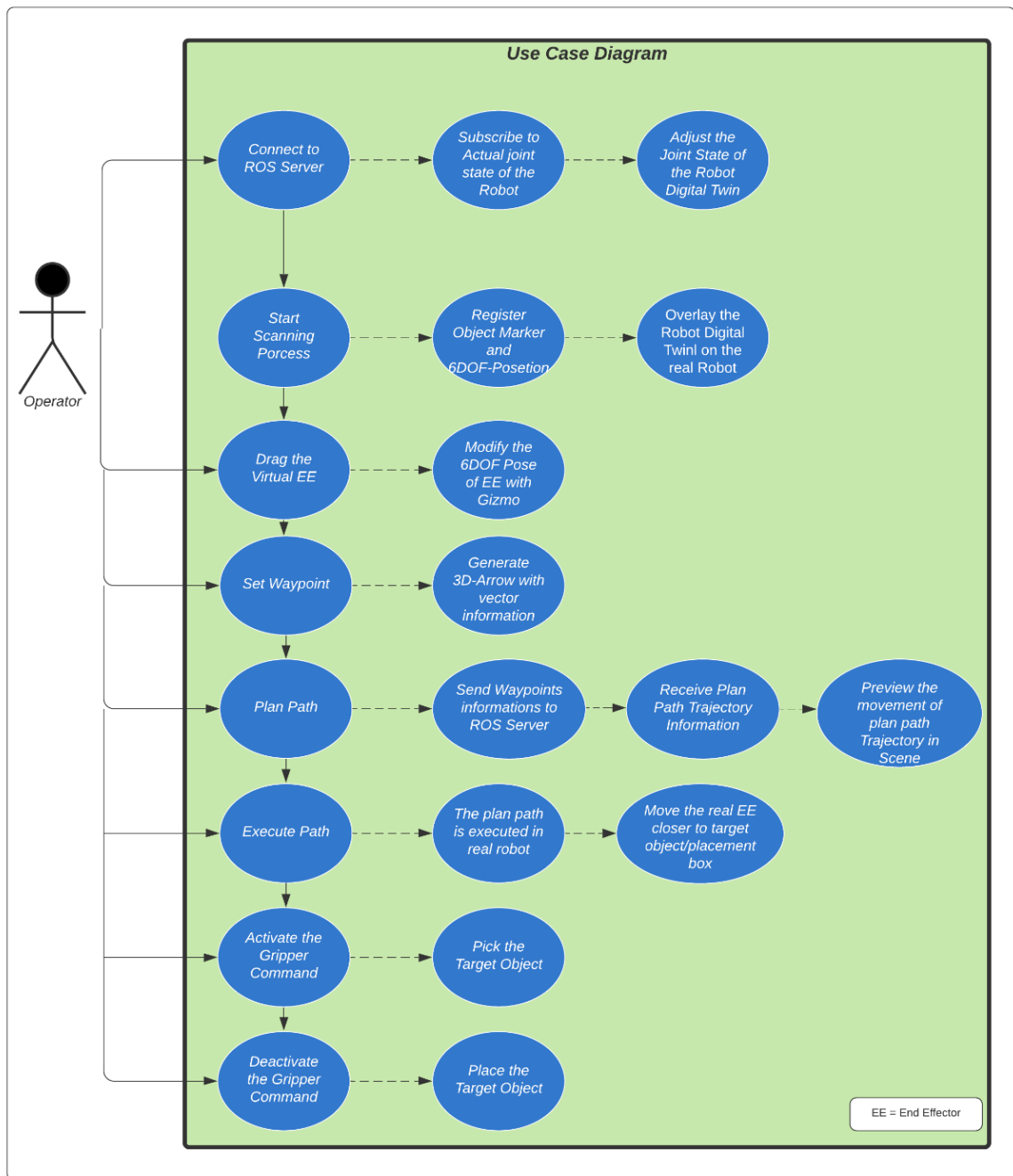


Figure 3-6 Use case study with the combination of the general concepts

### 3.4 Summary

The final concept of the system design is illustrated in figure 3-7. The required components are identified and clustered into three subsystems:

1. Server (Green Box)
2. Client MR application (Blue Box)
3. Robotic system

The server will act as the main broker layer where the custom robot applications and commands will be processed and sent between the MR application and the robotic system.

The Client MR application will act as the main Interaction layer where the human can input commands and instruction for the robot programming. Finally, the UR10 robot set-up will act as the robotic system layer for this thesis. It consists of the UR10 manipulator and the robot controller box. The teach pendant should be used only for a higher function like turning on the robot, selecting the program, and the function of their button. The communication between these parts is done with network communication. The detailed description of the whole system is further described in chapter 4.

The digital twin concept acts as a MR based extension of the programming by Demonstration, namely the kinesthesia teaching. This approach has practical reasoning, as the programming procedures are not unfamiliar and new to the veteran advanced user. Operators who have experience in PbD can easily adapt to this new method and perform more intuitively with it. The operators who have no prior knowledge of PbD can learn the concept faster as it is mainly structured with rich graphical information and models.

In this chapter, three main ROS packages are identified as potential solutions for developing a mixed reality-based robot program generator. In the following chapter, the approaches of integrating these packages and making them communicate which other will be shown in more detail. A simple use case study has been deployed to identify important interactions of the operator to achieve a

successful pick and place task has been deployed to identified important interaction of the operator to achieve a successful pick and place task.

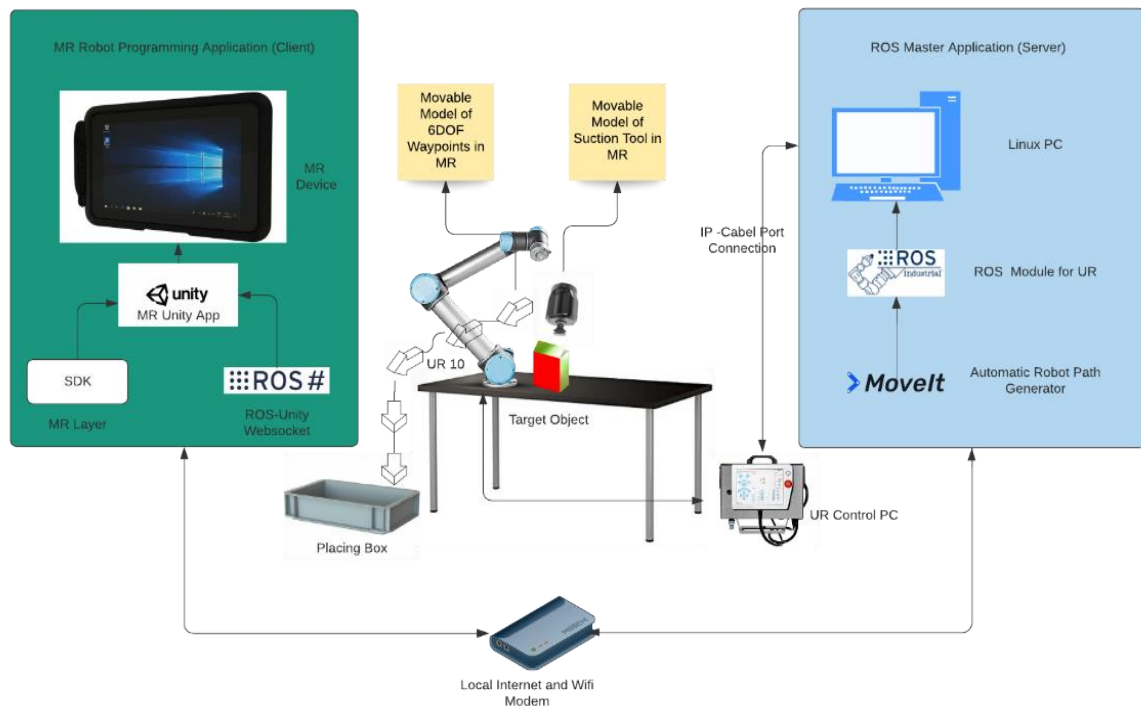


Figure 3-7 Final concept of the system

## 4 System Implementation

In this chapter, the steps to realize the concept of the MR based robot application is presented. The development process and the application flow are outlined. The final product of the implementation is presented as well.

### 4.1 System Architecture

In this section, the system architecture is presented. The system architecture is broken down into three sub-systems as explained in the previous chapter. The ROS server application, the MR-based client application, and the robotic system. There is a clear distinction between these three subsystems therefore separating them and developing them parallel will ensure a faster implementation and avoid heavy functionality dependency within the system architecture.

Each of these subsystems is made of components that handle a specific functionality in the system. The integration of these components enables the overall system to output the desired results. The interdependency of the components within each subsystem is illustrated in figure 4-1.

The system works under the principle of server-client architecture. The server application is built on top of the ROS operating system. The ROS server application act as the central part of this system architecture and runs specific robot application in form of ROS Nodes. It will manage the communication, the generation of a robot program, the reading of the robot sensors, and the input of the human operator through the mixed reality user interface layer. The server-side components demand high computational resources. For this reason, the server is deployed within the Workstation PC.

On the client-side, the MR application which displays an intractable digital twin of UR10 robot is deployed on MR capable hardware. The MR application needs to establish a stable and low latency connection with the ROS server. Through this communication channel, the actual sensor readings of the UR10 robot should be received from the ROS server and the input commands of the user through the MR application should be published back to the ROS server.

The robotic system is made of the UR10 manipulator, the UR control box, the touch panel, and the ethernet connection to the workstation pc. This subsystem should be the least modified and changed among the three subsystems. The reason for this is that the developed system should have the characteristic of a 'plug and play solution. The MR robot program generation concept should be able to be used on any UR10 robot. The solution should perform with high independence from the configurations of the robotic system layer. This ensures the scalability aspect of the application. The main modification in the subsystem should be opening a communication channel between the ROS Server and the

UR control box. Through this communication channel, the joint sensor reading can be published to the ROS server. Furthermore, the post-processed ROS server action files from the input commands from the MR application can be sent to the UR controller to be executed.

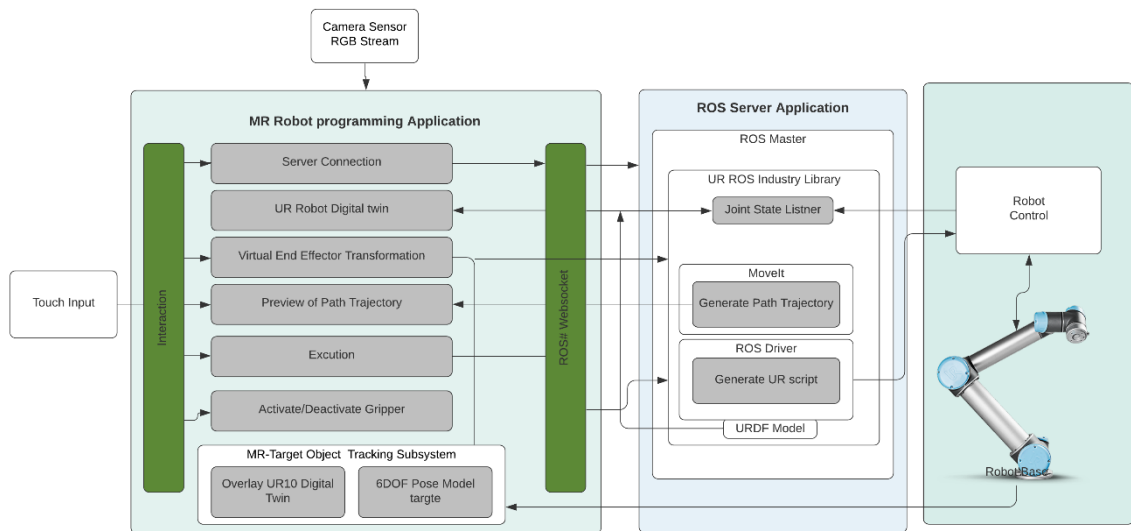


Figure 4-1 Final system architecture

## 4.2 System Sequence

The system sequence diagram is derived from the use case scenario in chapter 3 and can be seen as a more systematic description form of the use case. The actor of the system is defined as the application user or the robot operator. It outlines each action sequence that should take place on each respective subsystem, on which the whole concept is built on. This actor will initialize the action which leads to an inter-subsystem action sequence and also on the lab environment objects. From this analysis, the following things have been identified as points to be clarified before the detailed development of each subsystem could start:

- Number of input actions from the user
- Number of message channels
- Cycles of request and response messages
- Identification of continuous and non-continuous communication channel

A complete action cycle sequence means a user action triggers an action in the system which sends back another form of reaction towards the user. From figure 4-2, it can be seen that there are two complete cycles of an action sequence.

After the user made a successful connection with the ROS Server the system should receive the actual joint state constantly and update it back to the MR application layer. The second complete cycle can be seen from the tracking of the MR target object within the environment. The information from the MR tracking algorithm is used to update constantly the registration pose of the digital twin in the MR application layer. Therefore, these two cycles need to run continuously and without any interruption. If there is a break in the main functionality of the application, this cycle will be interrupted. Therefore, the frequency rate of the joint-state channel and the computational process of the MR tracking, need to be optimized between all the involved components to avoid a huge latency.

A half-action cycle sequence, on the other hand, means an action from the user which triggers an action in another subsystem or object. This can be seen as a remote-action scenario, where the user's action is transformed into another form, in action, in another system or object. In the diagram 4-2, it can be seen that there are three half-action cycle sequences. The three of them are associated with the change in the pose of the manipulator's end effector and the activation and deactivation of the gripper. As this is a half-cycle action, the communication channel will be opened only when needed as they are a non-continuous mode of communication. This approach will free up bandwidth resource which may be used for the two complete action cycles of the system.

### Mixed Reality Robot Programming for Pick and Place System- Sequence Diagramm

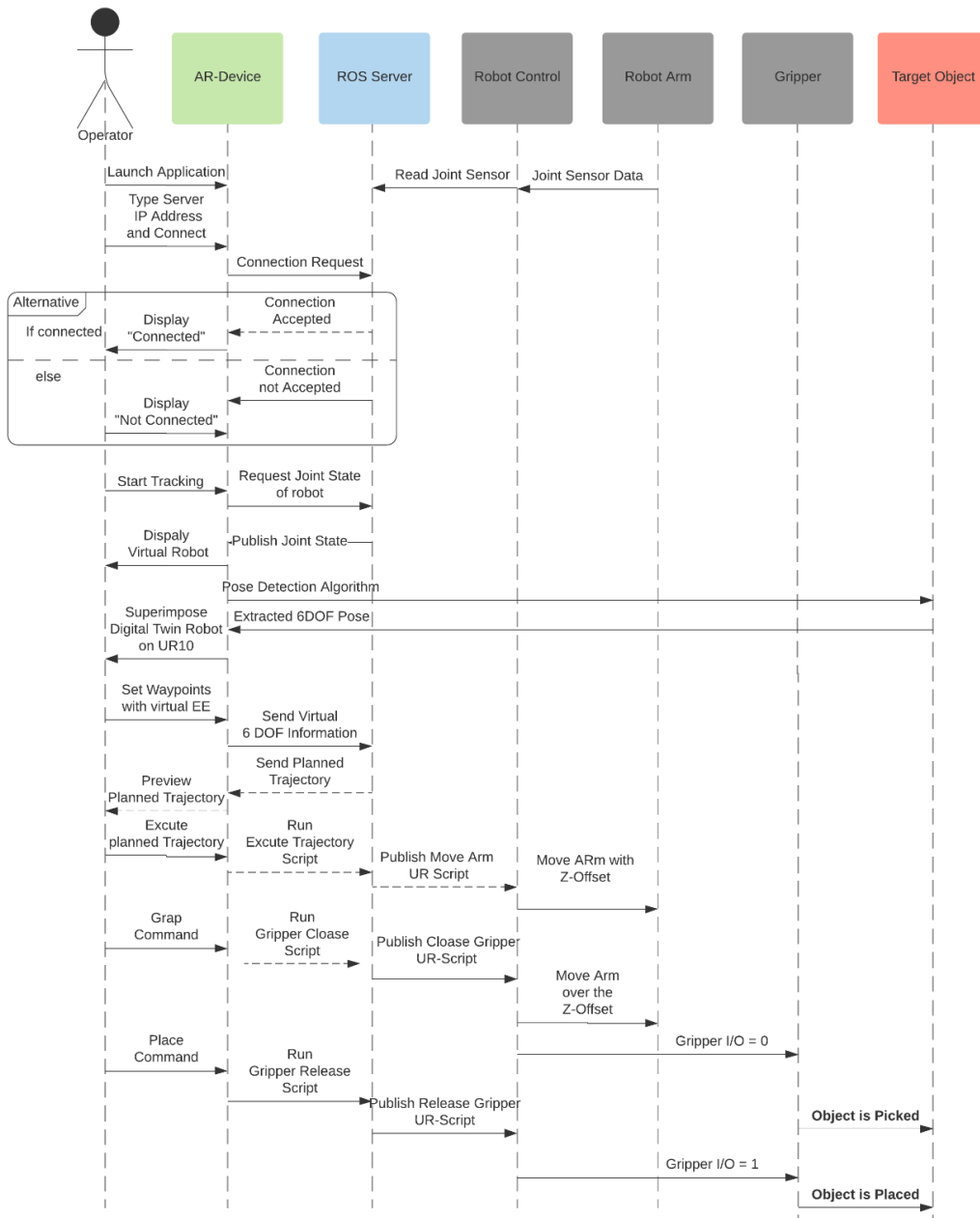


Figure 4-2 System sequence diagram



## 4.2.1 ROS Ecosystem

Setting up a ROS-based Ecosystem is one of the most important steps of the implementation. The workstation pc in the laboratory was running with Windows 10 at the time of this project. Windows Subsystem for Linux [23] (WSL) is installed on top of Windows 10. WSL is a compatibility layer for running Linux binary executables natively on Windows 10.

Afterwards, the ROS framework is installed on top of the Ubuntu 16.04 running on the WSL. After the installation, then a ROS workspace is created. ROS Workspace refers to the main folder which holds all the necessary source files to build the ROS server application. Furthermore, external ROS packages can be installed within the source file folder of this workspace. After installation, each individual external ROS package can be unpacked and integrated into the ROS server application. The following external ROS packages are integrated into the ROS workspace.

1. ROS Sharp package [20]
2. ROS-industry Universal Robot package [24]
3. Universal robot ROS driver package [25]
4. ROS MoveIt package [22]

The Universal robot ROS driver functions as an add-on UR10 driver layer for the ROS-industry Universal robot package. Although, the ROS-industry Universal robot package has a prebuild UR10 driver, several problems were identified with them during the implementation process. Furthermore, this add-on package has the following features which have been identified as a potential benefit for the project:

- Factory calibration of the UR10 inside ROS to reach Cartesian targets precisely
- Real-time-enabled communication structure which can handle the 2ms cycle time
- Speed-scaling functionality where the speed of the trajectory is controlled according to the movement complexity. This eliminates a jerking movement which was seen with the default driver

When these four ROS packages are installed in the source file folder of the ROS workspace with all their respective dependencies, the implementation of a custom robot program is possible with either C++ or python scripts. Furthermore, the ROS server workspace can be converted into a ROS package itself with all the important dependency files. This enables the application to be shared and deployed on any server running the ROS system.

## **4.2.2 UR10 Custom Configuration**

The Universal robot manufacturers provide a ready-made communication interface through an ethernet protocol. The communication between the PC which is hosting the ROS Server and the UR10 controller is set-up using an ethernet cable. In order to test a successful connection, the IP address of the UR10 robot (recorded from the teach pendant) is pinged back from the WSL terminal. If the ping does not work, it means the WSL terminal needs administrative permission to pass through the firewall configurations.

Further alteration needs to be done in the UR10 subsystem, in order to use the Universal robot ROS driver package. An external URcap needs to be uploaded to the UR 10 controller. URcap can be seen as an external app. They need to be installed manually. This can be done through the USB port of the teach pendant. Unity IDE Project Setting. This procedure is further explained in appendix C.

## **4.2.3 Unity IDE Project Setting**

The unity environment is set up for the development of the standalone application first. The standalone application uses a webcam and a monitor to project the MR scenario. Therefore, a webcam is necessary for this standalone application version.

Thereafter, the ROS Sharp unity package is imported into the unity project. For enabling the MR capability of the application, the suitable AR SDK will be imported into the project file. A copy of a project is made for each import of AR SDK. This avoids any compilation error during the building process of the project or compatibility issue within the Unity IDE. The AR SDK which is tested for this thesis will be outlined in section 4.4.3. For this thesis, two types of application builds are deployed:

- 1.Windows 10 Standalone application
- 2.HoloLens 2 UWP application

The HoloLens version of the app can be built with certain modifications within the Unity project of the standalone application. The reason for a HoloLens version is to prove the flexibility of the developed system being deployed in a see-through mixed reality hardware set-up. Furthermore, the intuitive grade and the performance of the two types of MR, the magic lens and the see-through MR, can be analyzed further.

### 4.3 Modular Design Implementation

For the implementation process, a modular design-based principle is taken into consideration. The concept of the system is divided and clustered into parts called modules. These modules are created through functional partitioning of the whole system. Each module is designed and implemented in a way that each of them has its own main functionality. In the end, each of these modules is interconnected to deliver the designated output of the whole system.

This approach enables system customization on a portion of the system rather than overhauling the whole system. Each module can be modified, replaced, deactivated independently without disrupting the whole system flow. This flexibility enables the system to be redesigned and tested with various forms of components and SDKs. Furthermore, it ensures easy scalability and extension of the system in future developments of the system.

Figure 4-3 outlines the modular components of the whole system. The system is made of 6 main modules where each of which is responsible for distinct functions within the system. The implementation of each modular is scripted in a way they are independent for each other to deliver the result and have ports of the communication channel to exchange data with each other. This ensures also an easy debugging approach to trace back any error in the system. The following section describes the implementation of each module in detail.

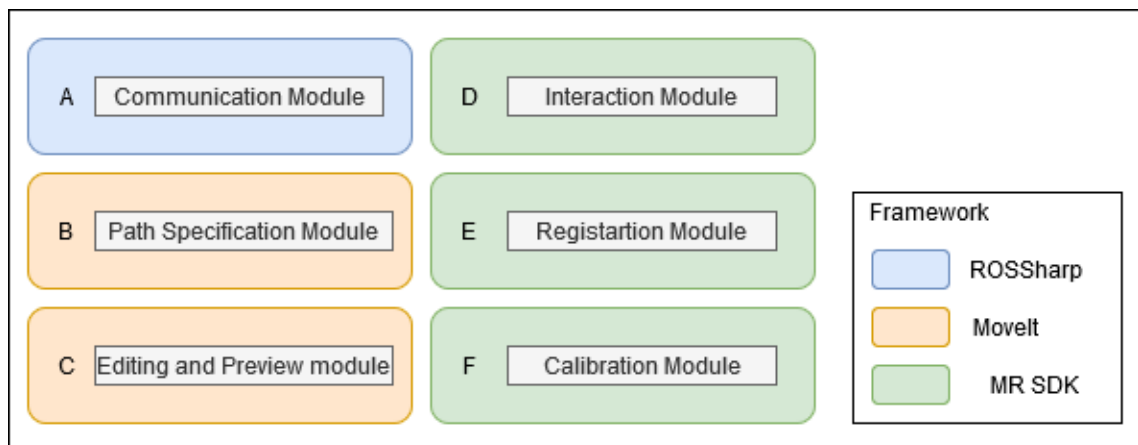


Figure 4-3 Modular components of the system

### **4.3.1 Communication Module**

The ROS Sharp package handles the establishment of a communication channel between a ROS server and an application that is developed in unity. This package uses websockets, a computer communication protocol which enables a two-way ongoing conversation between the client and the server over a common wifi network.

The ROS Sharp package is installed within the ROS workspace. Now a ROS node with ROS sharp functionality can be integrated into the ROS Master. This ROS node will open a websocket with the IP address of the host (workstation pc) which is running ROS. A custom port of the connection can be further defined within the code for example port 8080. The developer need to makes sure that the defined port is not being used by other application.

On the client-side, the ROS Sharp plugin is installed within the MR application through the import of the ROS Sharp unity package. The IP address of the workstation pc and the custom port is typed into this plugin. Afterwards, The ROS Sharp client websocket will be activated in the background when the app is running with a constantly listening connection from the ROS Host side. Both ROS Sharp node and ROS sharp plugin in the unity application establish a websocket connection. A successful handshake occurs, if the given IP address and port match. The publication and subscription of the ROS topics are allowed afterwards.

The data exchange between the different layers is possible through the serialization and deserialization of provided protocol buffers within the ROS Sharp stack on both host and client-side. This means the client MR application can be developed for various hardware configurations.

#### **The Transfer of URDF Model**

One of the main functionalities which of ROS Sharp provides is the transfer of the URDF model directly from the ROS Server side to the unity application after a successful handshake. The imported URDF file will be converted as a game object in Unity 3D. This enables a URDF model of the robot to be used as a 3D simulation model within a unity scene. Therefore, the URDF model of the UR10 is extracted from the previously installed ROS-industry Universal Robot package at the ROS server side. The imported URDF model is converted as a unity game object and added with a material file to illustrate a digital twin of the real UR10 robot.

### 4.3.2 Interaction Module

This module handles how the user is going to interact and input actions within the system. It is important in determining the intuitive factor of the application. The immersive user interface which the MR technology provides is used to achieve a unique UI, unlike the commercial 2D display UI.

The main interface is to enable the user to shift the position and rotation of the end-effector of the UR10 manipulator. Furthermore, the user can activate and deactivate the suction gripper. The design and implementation of the user interface of this application should assist the user to perform the pick and place task successfully.

#### Virtual Pointer and Dragging Gizmo

A virtual pointer will be used to manipulate the end-effector of the robot. One of the special features of MR technology is that the user can interact and manipulate a virtual 3D object displayed in the real world. This feature enables the design and implementation of a virtual pointer. The user is able to change the position and rotation of this virtual pointer in the real scenario. The MR application will capture and record this pose of the virtual pointer and translate them into robot end-effector pose goals. This pose goal will be published back to the receiving ROS node at the ROS server side. This will be further explained in the path specification module.

The virtual end-effector will be a replica model of the actual end-effector. The virtual end effector should be dragged around the scene with the user input action. For this, a special user interface, a 3D-gizmo is designed for the MR-application. This gizmo promotes assistance to a user to move the virtual end effector as seen in figure 4-4.

1. Three linear axes ( $x, y, z$ )
2. Three rotational axes ( $x_r, y_r, z_r$ )
3. The hybrid control of all the axes

These three modes of the gizmo can be selected with a callback function which is attached to the user keyboard inputs. This approach provides the user a systematic and cross-axis movement constraint control to move the end-effector in a certain linear axis or rotational axis. The last hybrid mode enables the user to move the end-effector freely in the environment without any cross-axis movement restriction. This effect mimics the action of kinesthetic teaching.

After the user registers a waypoint with the virtual end-effector through another callback function called 'set-target:', a corresponding 3D arrow model is generated on that specific space location like in figure 4-4. This 3D model arrow shows the vector arrow of the teached point. Furthermore, the same gizmo layer is available to for use to manipulate the teached waypoint in the real environment.

This is necessary for the editing module functionality. However, the user needs to toggle the gizmo layer from the end-effector model to the waypoints, to avoid confusion and misuse of the gizmo between moving the end-effector model and the waypoints.

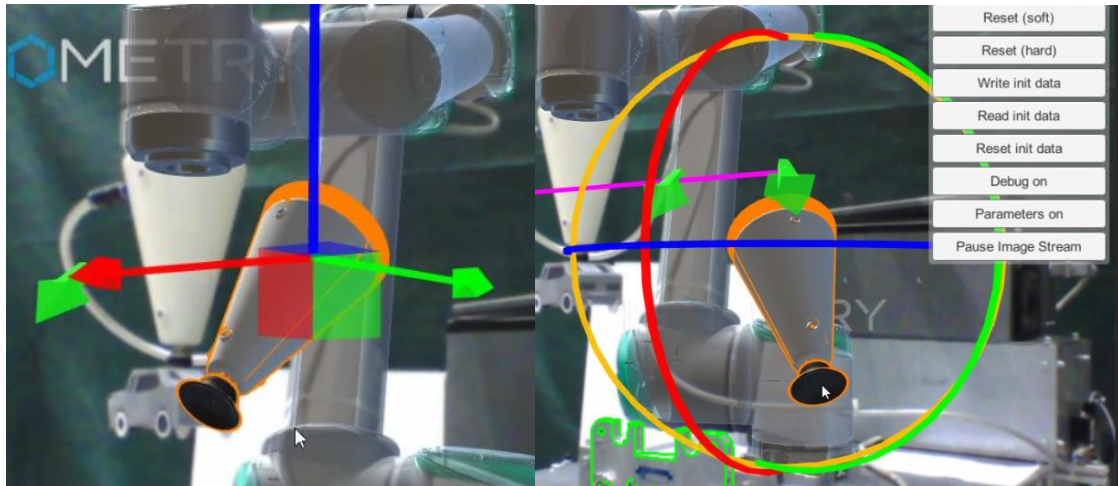


Figure 4-4 Virtual draggable end-effectors with custom gizmo

### Button Input action Interface

The final user interface of the MR-application is achieved with the usage of simple button inputs, which are anchored to the specified callback functions in the backend of the application instead of using an interactive virtual 3D object to activate the same callbacks. First reason, the buttons can be arranged in a way that they do not interfere with the user's view of the real robot. Overloading the viewpoint with unnecessary, unwieldy virtual 3D objects can lead to safety problems or confusion in interpreting the scene. Furthermore, this approach reduces the CPU usage, battery usage and the latency.

Figure 4-5 shows the button and their respective callbacks. The button cluster at the bottom right corner handles the waypoint teaching functions, the editing of the registered waypoints and, the activation and deactivation of the suction gripper. In the top left corner there is a callback button for a calibration layer of the application.

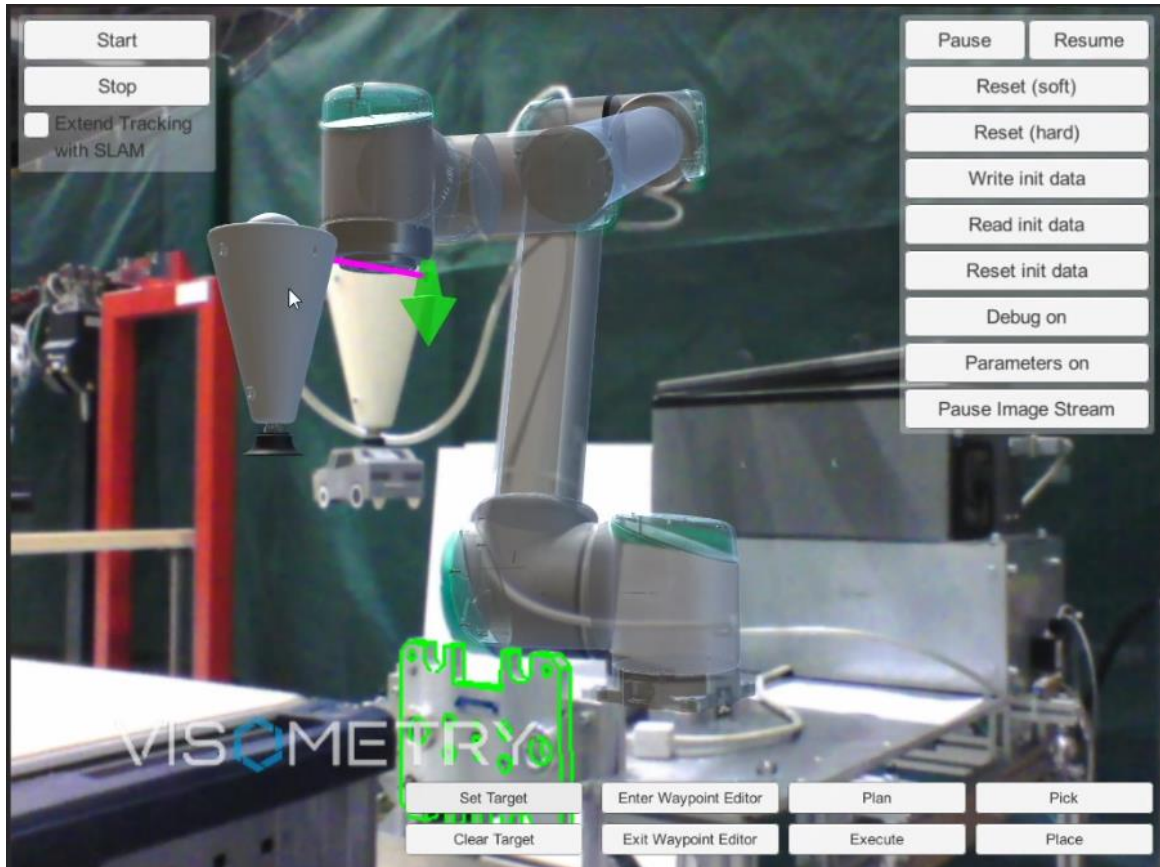


Figure 4-5 Final UI of the stand-alone application

### HoloLens Application UI

The UI interface for the HoloLens app version is slightly different from the stand-alone application because it was developed with the MRTK SDK. There is less functionality that needs to be implemented in the HoloLens version compared to the standalone app. This is because a new dimension opens up for user with a head mount display.

The main UI is composed of the button 'set target', 'clear target', 'plan', and 'execute plan'. This can be seen from the screen shot from the application in figure 4-6. However, the callback functionality remains the same as the stand-alone application. This demonstrates the flexibility of the modular design where a different type of UI can be implemented on top of the application according to the demands of the situation.



Figure 4-6 MR UI of the HoloLens version of the client app

### 4.3.3 Registration Module

This module handles how the digital twin is going to be registered and anchored on the actual base position of the real robot. It will be responsible for achieving the superimposition effect. The superimposition effect is achieved by merging the real-world coordinate system with the virtual world coordinate system. Therefore, an MR target needs to be deployed in the scene.

The performance of the registration module plays a vital role in developing a reliable and efficient MR robot programming application. This module should be implemented in order to register the virtual coordinate system as accurately as possible in the coordinate system of the real world. The following analyzes are



done in order to identify the best configuration of the registration module. First, the software development kit of the AR technology needs to be determined.

## The selection of the AR SDK

From the survey, it is found that there are many MR technology SDK providers. Vuforia SDK from the PTC provides various type of MR target approaches than the other SDK. For this thesis, an SDK from the VisionLib GmbH is also taken into consideration. The VisionLib SDK provides a low-level coding opportunity unlike the Vuforia SDK and it provides more industrial relevant MR functionality like a calibration layer for the custom camera. For the development of HoloLens 2, MRTK SDK is used as it provides special HoloLens 2 development configurations like point cloud anchoring and hologram interaction. The type of MR targets which each SDK supports are shown in table 14.

Capabilities	Vuforia Engine [26]	VisionLib SDK [27]	MRTK [28]
Image Target	Yes	Yes	No
Object Target	Yes	Yes	No
ML Target	Yes	No	No
Platform	Stand-Alone UWP, Android,iOS	UWP, Stand-Alone	UWP

Table 14 Selection of AR SDK

## Target type selection

The following three MR Target types and a point cloud anchoring method were available.

- Image Target
- Object Target
- ML Target
- OCL-PCL Anchoring

Each target type is implemented and deployed with the registration module. However, all of these four targets performed differently in comparison to each other. Some of the differences are the point of accuracy, stability, and re-registration.

The image target can be used to place the digital twin on top of the robot. However, the pose of the digital twin drifts significantly. This results in the digital twin appearing to be not aligned with the real robot after a trajectory is executed. The superimposed effect is significantly inaccurate in this form of target. Then the ML-based target is implemented by using the UR10 base structure's origin as the anchoring point. The 3D model of the UR10 base structure is used to train a ML model which can predict the pose of the model with a live camera stream. The ML training can be done through the Vuforia advanced model target [29] software. From this software, a weight file is generated which can be imported within the unity project to be used. The advantage of this ML-based target is that the digital twin can be registered from all directions unlike the image or 3D-Model-based target where the registration only occurs at a certain registration viewpoint. However, the ML-based target doesn't produce a consistent successful registration after the application is restarted although the viewpoint is the same. The ML target need to be further optimized and refined through a redefinition of the embedded machine learning process within the Vuforia advanced model target generator software.

Finally, the Object target is chosen for the final application as it provides a more stable, easier and more consistent registration than the ML target. Furthermore, it has a wider field of viewpoint for the registration than an image target.



Figure 4-7 Digital twin registered with different type of MR-Targets

## Final Registration Set-Up

A valve structure that is situated on the side of the robot table is used as an object target. This valve model is big enough to be seen from a far-field of the camera and also rigid and static. Furthermore, the CAD model of the whole UR10 robot with the table was made available by Fraunhofer IFAM. Therefore, a very accurate offset between the base of the UR10 robot and the valve structure can be extracted from the CAD model. This offset data is further double-checked in the real set-up.

The VisionLib SDK is used instead of the Vuforia SDK although both provide a method for object targets. The VisionLib object target performed considerably better in terms of consistency and stability compared to the Vuforia object target.

In order to register the digital twin, the user should try to align the valve structure within the outline of the valve projected in the scene. Upon successful registration, the outline will turn green and if the registration accuracy is altered with changing viewpoint, it will turn into yellow. This signals the user to re-register the model target to produce more stable tracking. This is one of the features of VisionLib SDK which provides a better intuitive usage compared to the Vuforia object target functionality. Moreover, the user can use this guideline UI to know where exactly the registration checkpoint is in the real scenario. This can be seen in figure 4-8.

Furthermore, this is a novel MR target strategy used in this thesis. An object which has a constant position from base coordinate system of the robot is used as MR target. In previous similar projects, the used MR targets are from a marker or object which has a dynamically changing location from the base of the robot. This statement is valid also for the OCL-PCL anchoring approach which used in the HoloLens application. The advantage of using a MR target which is part of the UR10 robot montage structure is that the extra calibration needed to transform the tracked MR target coordinate to the robot base is eliminated. The reason is that the structure's offset is well known. Thus, the superimpose of the digital twin can be achieved by adding this offset value only. Therefore, using a model structure where it's exact position relative to the robot base is known, is one of the way to streamline the system's efficiency.

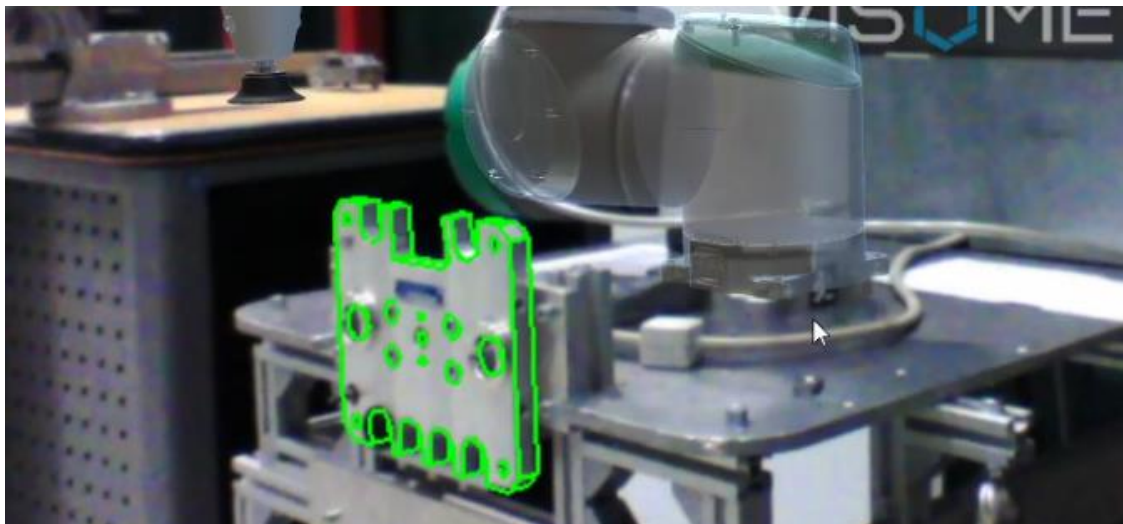


Figure 4-8 Valve structure used as model target

#### 4.3.4 Path Specification Module

This module contains the algorithms, which are responsible for teaching in the waypoints of the robot trajectory. The path specification approach should almost

mimic the method of kinesthetic teaching. As the difference the user transforms the position of the virtual end-effector instead of the real robot end-effector.

The waypoints are taught when the user activates the 'set target' callback function. This callback function inquiry the actual pose of the virtual pointer relative to the base coordinate system of the digital twin UR10 robot. The recorded pose consists of the  $x, y, z$  linear axis coordinate points and the  $x_r, y_r, z_r$  rotational axis angels.

Furthermore, an algorithm for a visual effect is also initiated when the 'set-waypoint' callback function is activated by the user. At the location of the recorded waypoint, a virtual 3D arrow model is spawned. The direction of this arrow model is aligned with the outward axis (z-axis) of the UR10 tool center point (TCP). However, the z-axis position of the arrow always coincides with the origin of the virtual end effector coordinate system used in the current robot session. This feature enables flexibility when another end-effector with different dimensions is mounted on the TCP of the UR10.

The pose information of the recorded waypoints is stored only on the MR application side. They are not published back to the ROS server, yet. This is because this action has been identified in the previous sequence diagram, to be a half-cycle action sequence. In return, the bandwidth of the websocket connection between the MR application and the ROS server is saved up for the full cycle action sequence.

There is no restriction on the maximum number of waypoints that can be taught in the environment. For each consecutively recorded waypoint, a line is illustrated visually as shown in figure 4-9. However, it should be noted that Important that this is just a connection between the points in the MR scene and not the real robot trajectory which will be planned later from the ROS MoveIt node. The combination of all the waypoints generates the trajectory path which the user wants the robot to perform. With the visual feedback of the waypoint and the trajectory path, the user can reprogram the movement of the robot intuitively, where the end-effector is successfully brought near the to pick an object. The same products will be done when determining the drop location of the picked object. Furthermore, visual feedback is illustrated through the MR effect. This gives the user an illusion of seeing instantly taught in waypoints and the trajectory path in the real world. This feature is not available in the existing robot programming method.

Pose information of all taught in waypoints, which are stored within the MR application, are published to the ROS server through the 'plan\_path' ROS topic. On the ROS server-side, a ROS node that is subscribed with this 'plan\_path' ROS topic, will receive this pose information and forward them to another ROS node. This ROS node is running a MoveIt algorithm. This MoveIt ROS node takes in waypoint's pose information and supplies it to the automated plan path generation algorithm. When a potential solution is found, it will return a trajectory plan. This Trajectory plan is composed of the required motor actions on each joint, in order

to reach this waypoint. The algorithm will iterate the generated trajectory plan for each additional waypoint being injected into it.

### 4.3.5 Editing and Preview Module

The editing and preview module contain two main functionalities. It is responsible for enabling the user to modify already defined waypoints and previewing the corresponding robot movement for those taught waypoints.

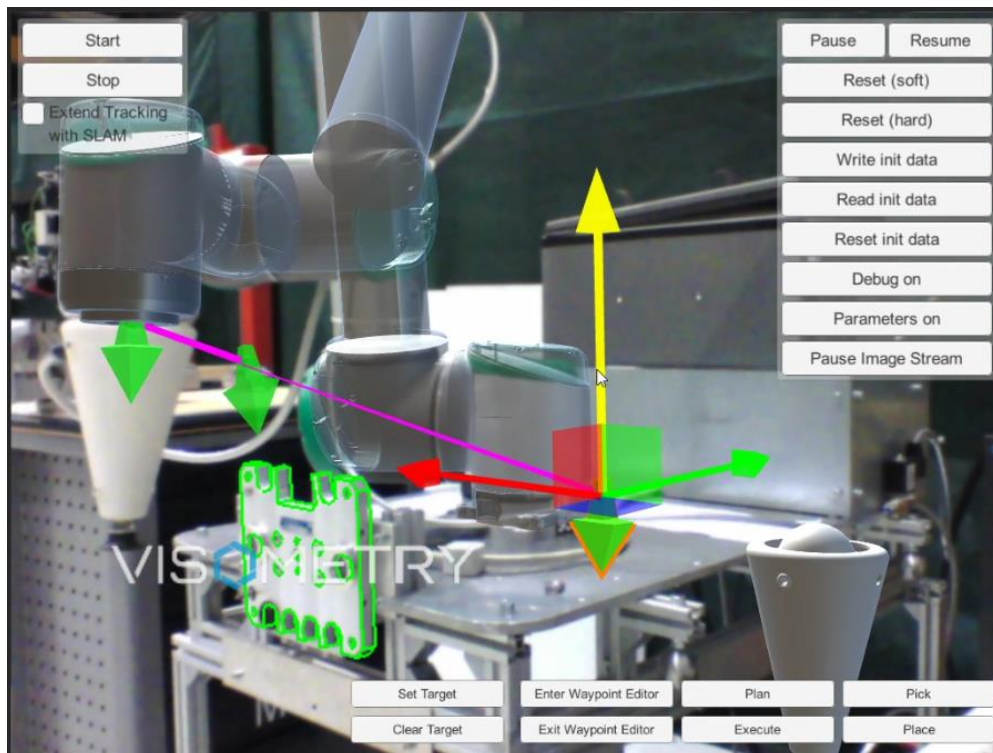


Figure 4-9 Waypoints represented by 3D arrow model

The editing function provides the user freedom and flexibility in adjusting the pose of the previously recorded waypoint and removing waypoints. The adjusting function is activated with the use of the previously mentioned dragging gizmo. The user can transform the pose of any recorded waypoint, by interacting with the 3D arrow model through the gizmo. The function of deleting waypoints can be activated with the 'clear target' callback function. On the other hand, the preview of the robot movement function is handled by the 'plan path' callback function. This callback function is invoked when the user presses the 'plan path' button.

A previously created ROS node within the path specification module stores the generated trajectory plan for the robot to execute. When the 'plan path' callback function is invoked, this stored generated trajectory plan is published back to the

MR application through the 'plan\_path' ROS topic. The received trajectory plan is further interpreted to derive the necessary parameters for a visualization of the subsequent movement with the digital twin. Those parameters are acceleration, velocity, and time steps.

Overall, the editing functionality and the preview functionality within this module run independently. The reason for this again is to save bandwidth for the websocket communication. The preview model is not constantly activated for each modification of the waypoints, only when the user is satisfied with the final modification, the preview of the new robot movement can be invoked.

### **4.3.6 Execution Module**

In this module the final call from the user is handled to execute the desired physical actions to be performed by the robot itself. This includes the robot motion and activation and deactivation of the suction gripper. The execution of robot motion is further categorized into two modes, autonomous mode, and semi-autonomous mode. These two modes need to be selected at the welcome screen menu of the MR application. After choosing the desired mode, the respective application view scene will be promoted.

#### **Autonomous Mode**

The autonomous mode utilizes the idea of the "follow the leader principle". The real robot end-effector will try to follow a virtual 3D sphere called the 'red-ball. In this mode, the preview module is deactivated and the path editing module is partially deactivated. This is because the displacement of the end-effector of the robot will be almost simultaneously with the displacement of the red-ball. This eliminates the need for a robot movement preview, or a waypoint teach in functionality. This mode is enabling a quick robot movement which does not require a lot of detailed planning of the robot trajectory path.

The user is prompted with a "red-ball" at the tip of the wrist of the UR10 robot in the MR scene. This red ball can be dragged in six degrees of freedom by the user through touch input. The red-ball and the buttons for the activation and deactivation are the only intractable object in the MR-scene as seen in the figure 4-10. When the red ball is shifted in the scene, the actual end-effector of the robot will move to the new pose of the red ball. This mode gives the user a quick usage of the application.

Furthermore, for safety reasons, the displacement of the red ball is locked only in the linear axis and the transformation on the rotational axis is restricted. The ball movement which contributes a self-robot collision path will not be executed as it is being restricted in the ROS MoveIt node by default. These three degrees of

freedom movement of the end-effector avoid any unforeseen joint movements from the MoveIt ROS Node, which may raise the safety concern of the user.

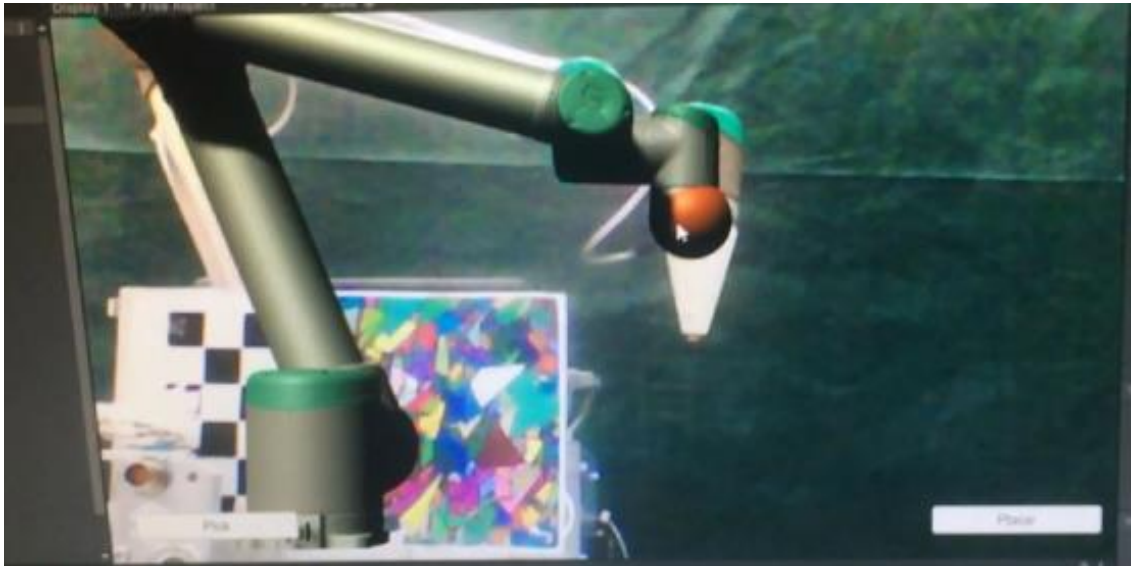


Figure 4-10 Movable red-ball in the autonomous mode

### **Semi-Autonomous Mode**

This mode offers the user full control and programming flexibility. It gives the full functional capacity to the path specification module and the editing and preview module. This gives the user a chance to supervise whether the planned path is useful or safe in the deployed real scene which requires careful obstacle assessments.

First, the user must teach the waypoints and preview the respective robot movement plan from the ROS server. If the movement of the robot is safe and appropriate, the user can execute this planned movement on the real robot by activating the 'execute plan' callback function. This callback function will publish a state message back towards the MoveIt ROS node running at the server-side. This state message will trigger the MoveIt algorithm to redirect the stored trajectory plan within its algorithm to the universal robot ROS driver node. Then this driver ROS node will generate a URscript with the necessary joint motor commands and uploaded to the UR 10 control system. On the UR10 side, the generated path plan will be received by the continuously running additional URcap which had been installed previously. When a script is received by this URcap, it will be forwarded to the internal control system and UR10 robot's controller and this generated URscript will be deployed to produce the planned movement.

## **Gripper Activation**

This section describes how the gripper is activated and deactivated. On the robotic subsystem, the gripper is attached to the digital “Input/Output” (I/O) socket. The gripper can be activated or deactivated through the toggle of the I/O port in the UR 10 control box. The Universal robot ROS industry package provides an option to toggle this I/O port of the UR control box remotely.

On the MR application side, with the press of the buttons, the user publishes the active or deactivate state of the gripper through the ‘gripper\_state’ ROS Topic. On the server-side, a ROS node is implemented which holds a callback function which activates and deactivate this port.

### **4.3.7 Calibration Module**

In this module, the calibration and optimization of the MR tracking subsystem is implemented. This is done because the lighting condition, the camera module, and the available computational resource contribute to the variation of the application performance and overall tracking quality.

This module is added with help of the VisionLib SDK, where numerous configuration parameters for calibration procedure are made available for the developer side. set up for the developer. This parameter configuration is integrated into a custom GUI for this thesis. This configuration parameter can make the tracking system to be more flexible and adaptable to environmental conditions and varying hardware configurations.

Furthermore, the VisionLib SDK provides the option to calibrate the camera system used for the application. This enables the program to be scalable by means of using various kinds of external cameras. For each calibrated camera a calibration file is automatically generated and can be saved in the local storage to be used when needed. This calibration feature is not a straightforward process when Vuforia is taken into consideration. Vuforia has predefined calibration files only for well-known smartphones and tablet models within their SDK: This is because Vuforia is mainly used for the development of AR applications for smartphones and tablets. This is a limitation for the hardware configuration flexibility which is common in an industrial use case scenario.

## **4.4 Summary**

In a nutshell, the described system is designed and implemented in a modular design approach. These modules are scripted in a such way that they are independent and can easily be activated and deactivated without needing to heavily modify the whole system. This feature is demonstrated in the toggle between the autonomous-mode and semi-autonomous mode of the robot motion



execution. This approach also simplified the implementation of the HoloLens 2 build where modification only happens within certain modules like the registration module. This approach will ensure easy maintenance and scalability in the future.

A ROS server is deployed with all the necessary components which supply the vital information, like sensor readings, the planned path from the MoveIt algorithm, and the toggle of the digital I/O, to the client MR application. With this key information, an MR application layer that can perform the robot programming can be successfully implemented. Furthermore, a specific robot task like the pick and place task can be successfully performed using this MR layer. On the robotic system side, no major modification has been done for the integration of this system. This proves the generalized applicability of the system itself.

In the end, a standalone MR application for desktop is successfully deployed. This application uses the monitor to display the MR scene and the camera to capture the real scene. This application uses the magic lens effect of the MR technology. The same application is remodified and deployed in a HoloLens2 successfully. This is done to show the adaptability of the system architecture for different types of MR hardware and to compare the difference between the see-through MR against a magic lens MR. The performance comparison of these two-application types will be furthered outline in the discussion section of the chapter 5.

The implementation and deployment of the system is done successfully in the test environment. However, the performance of this system needs to be analyzed and studied with a systematic method. This is done in the following chapter where a systematic study is conducted, and the insights derived from this experiment are discussed.

## 5 Experiments and Evaluation

In this chapter, the final output performance of the whole system is studied and analyzed through an experimental approach. The MR application enables a user to teach in a waypoint for robot programming through manipulation of virtual objects, which are anchored within the environment. The combination of a set of waypoints serves as building blocks of the robot trajectory line for a pick and place task. Therefore, the MR application should provide an accurate and stable waypoint definition within a working environment. The accuracy of the taught waypoint is taken as the key performance indicator (KPI) measurement of the system. This is because accuracy plays a fundamental role in robotic. A concrete insight and potential evaluation of this new MR-based robot programming can be derived by obtaining this KPI. The other factors like the speed of program implementation, the complexity and flexibility of the robot programming, the learning curve of the method are not investigated in a systematical manner for this thesis. The reason is that this factor depends heavily on the user's personal preference of application interaction and previous robot programming experience. However, this factor can be analyzed in a qualitative manner and the insight from it can be used for the future development of the whole system. In this chapter, such qualitative analysis is done based on the user experience from the final application and will be used to benchmark the intuitive aspect of the application.

In general, the difference between the planned or taught in end-effector pose and the executed end-effector pose by the robot is defined as the offset error,  $\Delta E$ . This offset is interpreted as the accuracy value. The following accuracy values are determined explicitly with a respective experiment in this chapter:

1. The accuracy and repeatability of the UR10 robot,  $\Delta E_{UR10}$
2. The accuracy of the ROS driver and the accuracy of the path generation with the MoveIt algorithm,  $\Delta E_{ROS}$
3. The accuracy of the waypoint teaching through the MR user interface,  $\Delta E_{MR}$

The offset error from the UR10 robot,  $\Delta E_{UR10}$ , is produced through the presence of inaccuracy within the mechanical and electronic system of the robot itself. The source of this error comes from the internal calibration error within the robot's controller. This value is normally provided by the manufacturer of the robot. However, this value needs to be double checked in order to exclude this source of offset error from the total offset error. Therefore an independent experiment needs to be conducted to find this value explicitly as shown in the expression (1).

$$\Delta E_{mean\_exp\ 1} = \Delta E_{UR10} \quad (1)$$

The offset error from ROS server,  $\Delta E_{ROS}$ , also plays a vital role in the overall performance of the system.  $\Delta E_{ROS}$  is the sum of the calibration error within of the Universal robot ROS driver package,  $\Delta E_{ROS\_Driver}$  and the accuracy of the

automated path generation from the MoveIt algorithm,  $\Delta E_{MoveIt}$ . This relationship is stated in the expression (2). Both the ROS driver and the automatic path generation from the MoveIt algorithm are not determined separately for this thesis. The overall accuracy from the generated UR script command from the ROS server is taken as the sum of the error from both ROS nodes outputs. However, these value needs to be defined explicitly, in order to investigate the reliability and stability of automatic robot joint movements commands generated to reach a certain waypoint. Therefore another independent experiment needs to be conducted to find this value. The relationship of this error can be found through the expression (3) and (4).

$$\Delta E_{ROS} = \Delta E_{ROS\_Driver} + \Delta E_{MoveIt} \quad (2)$$

$$\Delta E_{mean\_exp\ 2} = \Delta E_{UR10} + \Delta E_{ROS} \quad (3)$$

$$\Delta E_{ROS} = \Delta E_{mean\_exp\ 2} - \Delta E_{mean\_exp\ 1} \quad (4)$$

The final experiment need to be conducted to find the offset error value which will be generated from the deviation between the taught in waypoint through the virtual end-effector on the MR user interface and the executed actual robot's end-effector pose. This offset value is defined as  $\Delta E_{exp\ 3}$ . This value is an accumulation of the offset error from the UR10's system itself ( $\Delta E_{UR10}$ ), the offset error from the ROS server ( $\Delta E_{ROS}$ ) and offset error through the MR user interface ( $\Delta E_{MR}$ ). The final offset error from the MR user interface,  $\Delta E_{MR}$ , can be taken as the main KPI of the MR robot programming application. These relationships are defined in the following expression (5) and expression (6).

$$\Delta E_{mean\_exp\ 3} = \Delta E_{UR10} + \Delta E_{ROS} + \Delta E_{MR} \quad (5)$$

$$\Delta E_{MR} = \Delta E_{mean\_exp\ 3} - \Delta E_{mean\_exp\ 2} - \Delta E_{mean\_exp\ 1} \quad (6)$$

The explicit value of  $\Delta E_{MR}$  shows how well a user can teach in a waypoint accurately through the developed MR application. Thus the reliability of the system's MR function layer can be verified and analyzed. However, this offset error from the MR user interface,  $\Delta E_{MR}$ , is made of further two parts. First the

offset error from the MR registration, which originating from the inefficiency of the tracking algorithm ( $\Delta E_{MR\_only}$ ). Secondly, the offset error caused by human error when using the virtual pointer to teach in a waypoint, ( $\Delta E_{Human}$ ). This relationship is shown in the expression (7).

$$\Delta E_{MR} = \Delta E_{MR\_only} + \Delta E_{Human} \quad (7)$$

The offset error from the MR tracking system,  $\Delta E_{MR\_only}$ , has more than one source invoked. For example, the lighting condition of the lab environment, the efficiency of the used camera sensor, the efficiency of the MR SDK algorithm, and the efficiency of the used MR target. These error sources can be reduced through the calibration module which was implemented in the previous chapter. During the part of the experiment, where the  $\Delta E_{MR}$ , is being determined, this calibration module is used to initiate the best and optimum MR registration on the scene. This is to ensure the  $\Delta E_{MR\_only}$  is at its minimum influence on the final performance of the application. This step will be further discussed in this chapter later. Meanwhile, the other remaining part is the human source error,  $\Delta E_{Human}$ . This error originates from the human sensory system and human misjudgment. The MR application demands high usage of the visual sense of the human in order to use it properly. A common mistake that can happen through the visual sense is the parallax error. The user has the chance to misinterpret the exact location of the taught in waypoint in the real world due to false perception of the reality from a certain point of view. There is a chance for human parallax error when defining the waypoint. These errors can't be avoided fully because the system itself demands the interaction of the user to define the waypoint. Therefore, during the experiment, the steps of avoidance of this parallax error are taken to consideration as well. These steps will be further discussed as well in the methodology section.

Moreover, in this chapter a comparison between the format of the MR presentation, the magic lens effect MR and the see-through MR, is also outlined by analyzing the user experience for completing successfully a pick and place task.

## 5.1 Methodology and Set-Up

An experiment is designed and conducted in such way, that the previously defined offset errors like  $\Delta E_{UR10}$ ,  $\Delta E_{ROS}$  and  $\Delta E_{MR}$ , can be found explicitly. This is important for an independent and precise evaluation and validation of each main subsystem which is functioning within the whole system. From these values, a concrete insight can also be drawn from the reliability and justification of the

combination of the subsystems and helps to obtain insights for future development.

Fraunhofer IFAM has made an industrial graded laser tracker, model AT401, from the company Leica, available for this experiment. The laser tracker can track a point in the environment at an accuracy point of  $\pm 15 \mu m$  and with repeatability of  $\pm 7,5 \mu m$ . According to ISO9283, the benchmarking of a robot system accuracy and repeatability normally lays in the range of  $\pm 1 mm$  for accuracy and  $\pm 0,1 mm$  for repeatability. Therefore, having a measurement device with much higher precision will enable a precise pinpointing of the error source. The laser tracker model AT401 measures two angles and a distance value of a given target object. A retroreflective mirror will be attached to this target object and a laser beam will be projected by the laser tracker and received from the mirror's reflection. The distance value is derived from the time of flight of this returned laser beam and the two angles are calculated based on the tracker's own head angular orientation, through a special encoder. For this experiment, three spherical retroreflectors (SMR) are used. An SMR is a type of retroreflective mirror which is designed in a way that, it has a fixed offset distance with respect to any kind of surface to which it is attached too. With an SMR, the plane coordinate of the target surface can easily be obtained.

Three independent experiments need to be conducted for this thesis. The first part is conducted to find the  $\Delta E_{UR10}$ , the second part is conducted to find the  $\Delta E_{ROS}$  and finally, the last part is to find the  $\Delta E_{MR}$ . The first and second part are conducted without the MR application layer. In the first part, the pose of the end-effector is preprogrammed through the teach pendant. A total of 18 waypoints are registered in the teach pendant. These 18 waypoints are distributed in such way that variation of the accuracy from the reach zone of the robot is also addressed. This is because, the lightweight robot still can't offer a constant accuracy value for all the points within its maximum reaching zone. The reach zone of the robot is divided into three zones near range zone ( 400 mm from robot base) the middle reach zone ( 800 mm from robot base), and finally the far reach zone ( 1300 mm from robot base),. These three zones are further divided into two quadrants, the right quadrat(Q1) and the left (Q2). The 18 waypoints are placed in these two quadrants evenly and the position of the waypoints in Q1 mirrors the position of waypoints in Q2. Figure 5-1 illustrates these waypoints taught graphically. The final pose of the end-effector, after each waypoint's trajectory has been carried out through the teach pendant, is tracked by the laser tracker. The whole process is repeated three times which yields a total of 54 measurements. These

measurements are recorded straight from the laser tracker's post processor software.

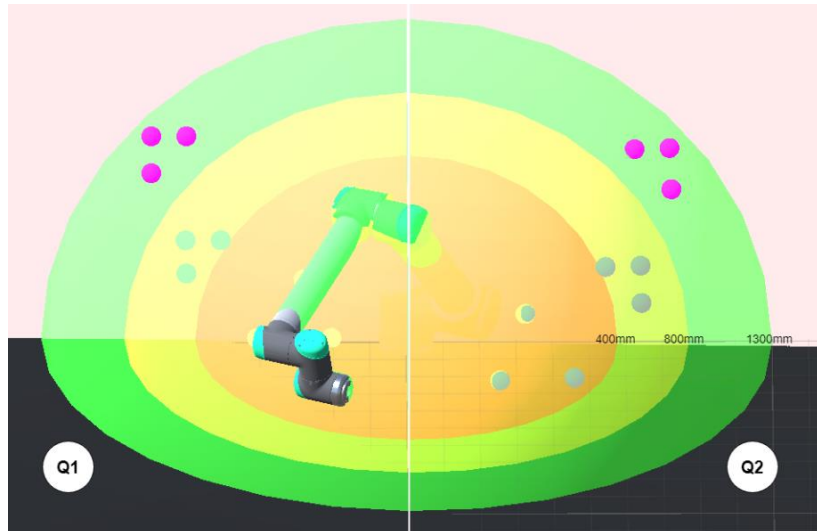


Figure 5-1 Measurement zone and quadrant separation

The basic idea for the first and second part of the experiment is to track the end-effector pose relative to the robot base coordinate system. In the first experiment, the waypoints are taught-in through the teach pendant. The taught-in waypoint coordinate is relative to the robot base coordinate system. Therefore, the coordinate of the robot base is derived first using three tracked SMR values. After that, the end effector poses are tracked precisely using the three SMR. A custom end effector is manufactured through high precise CNC machine as shown in figure 5-2.

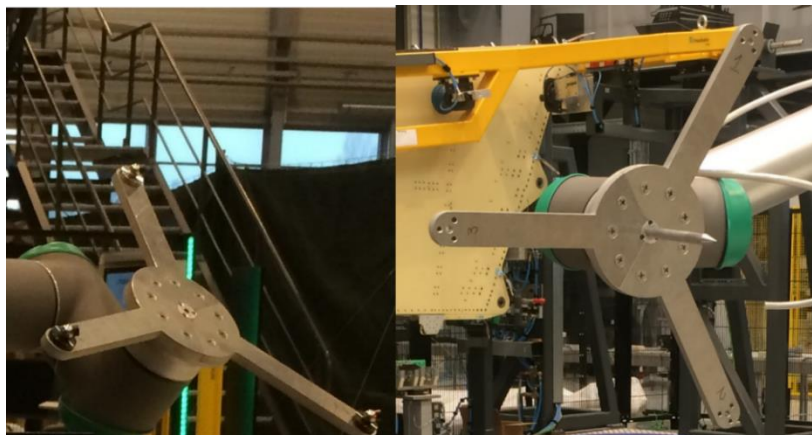


Figure 5-2 Custom end effector

This structure will hold the three SMRs in place during the tracking process. From the three tracked planes of the three SMRs, the center point and its pose relative to the robot base can be derived. From this value  $\Delta E_{UR0}$  can be derived based on ISO 9238 standards as shown in the expression (8) where  $\bar{x}$  is the mean position attained in the respective translational and rotational coordinate value and  $x_c$  is the command position which is programmed within the robot control system. The same expression is used for the remaining translational axis and rotational axis. The final  $\Delta E_{UR10}$  value is obtained as stated in the expression (9) and expression (10). Meanwhile the repeatability value of the UR10 can be obtained directly from the laser tracker's post-processing software.

$$\Delta E_{UR10_x} = \sqrt{(\bar{x} - x_c)^2} \quad (8)$$

$$\Delta E_{UR10\_trans} = \sqrt{(\Delta E_{UR10_x})^2 + (\Delta E_{UR10_y})^2 + (\Delta E_{UR10_z})^2} \quad (9)$$

$$\Delta E_{UR10\_rot} = \sqrt{(\Delta E_{UR10_{R,X}})^2 + (\Delta E_{UR10_{R,Y}})^2 + (\Delta E_{UR10_{R,Z}})^2} \quad (10)$$

The second part of the experiment is conducted with the same set-up as the first experiment. However, the approach of teaching the waypoints is altered to the ROS system where a customized MoveIt ROS node is running. Within this customized MoveIt node, the pose of all the 18 waypoints which are defined in the first part, are programmed as command input of the MoveIt algorithm. The algorithm will generate the appropriate robot joint movement command for each respective waypoint's pose input. The final pose of the end-effector after the execution process of each waypoint is also tracked by the laser tracker and recorded as measurement data. The calculation of the accuracy is repeated as done in the first experiment.

The final part is conducted with a modification on the custom end-effector. This third experiment is conducted to find the  $\Delta E_{exp3}$ . From this value, the estimation value of  $\Delta E_{MR}$  can be found as stated in expression (6). A custom pointer structure is manufactured with a high precision CNC machine and it is mounted on the previously custom end-effector as seen in figure 5-2. This pointer structure has an offset of 50 mm from the center of the end-effector after it has been mounted on it. The actual pose of this pointer within the real scene, relative to the robot base, is calculated through tracking of the custom end-effector center point pose with three SMR mirrors and addition of the 50 mm on the z-axis. Furthermore, a second pointer structure with the same offset value of 50 mm is

manufactured separately. This structure is fixed on a position within the experiment scene. The exact location of this pointer relative to the robot base is tracked initially with the laser track and this location is not altered throughout the experiment. The whole setup can be seen in figure 5-3.

In this experiment the basic idea is that the position of the structure's pointer will act as the absolute target point for the user. On the MR application, an identical virtual model of the custom end-effector with the pointer assembly is provided to the user. During the experiment, the user should alter the virtual end-effector pose within the MR scene, so that the pointer of the virtual end-effector reaches the location of the target point with pinpoint accuracy. The movement of the virtual-end-effector is restricted only in the translational plane. This is because it delivers the fastest and easiest approach to make the pointer of the virtual end-effector coincident with the pointer of the custom structure as shown in figure 5-3. The user may use the preview module to supervise the accuracy of the robot trajectory and edit the taught in waypoint as much as possible, until the user got the feeling that an accurate waypoint is taught in. When the user is confident with the virtually taught in waypoint, it will be executed by the real robot and the final pose of the end-effector relative to the structure pointer will be tracked using the laser tracker and also recorded as a measurement. A total of three measurements are done in this experiment.

Furthermore, it should be noted that for this experiment the 'magic lens' MR type is used. The scenario is captured through a webcam and a monitor display is used to show the user the digital twin of the UR10 robot with the custom virtual end-effector. Therefore, the point of view of the MR scene is only based on the webcam's perspective. The performance of the see-through MR is further discussed and analyzed in a qualitative manner in the next section.



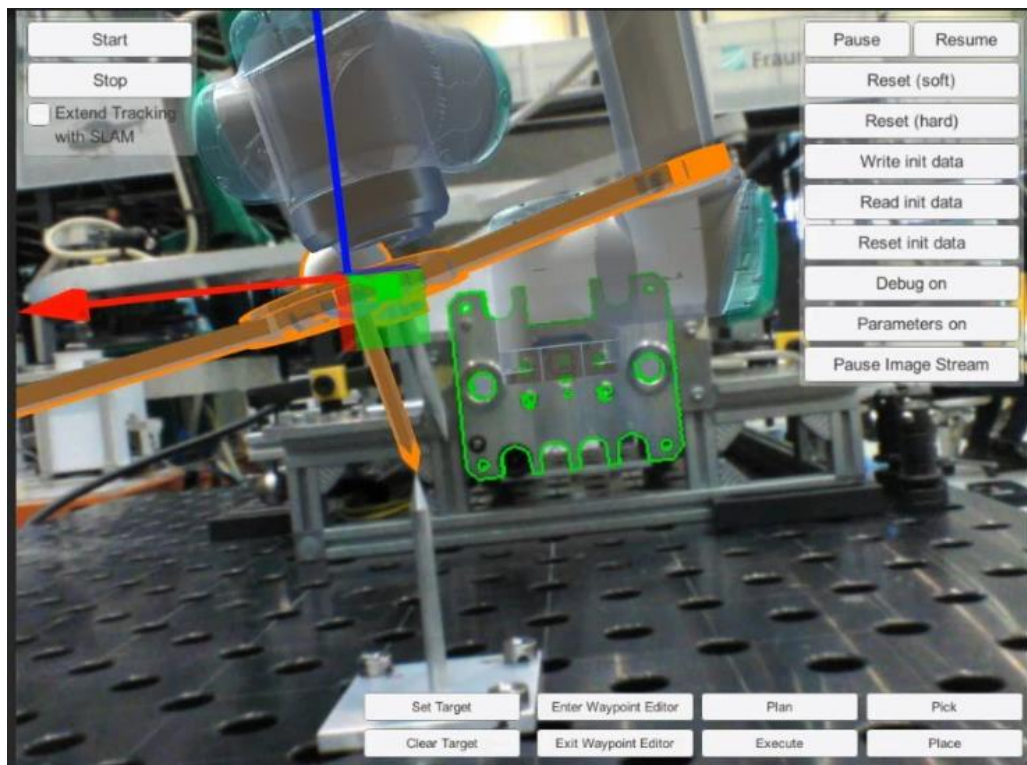


Figure 5-3 Pointer structure and virtual end effector pointer

## 5.2 Results

In this section, the results obtained from the three experiment parts are presented. The interpretation of the measurement data is illustrated in form of a diagram for better understanding which makes it easier to derive important insights. The section is divided into two parts. First, a quantitative performance analysis is done based on the collected data. Secondly, a qualitative performance analysis is done based on the overall user experience during the robot programming with the MR application. Furthermore, an example of a pick and place task is also presented. The overall quality of the pick and place task execution is discussed further.

## 5.2.1 Quantitative Performance

### Accuracy and Repeatability of UR10 and MoveIt

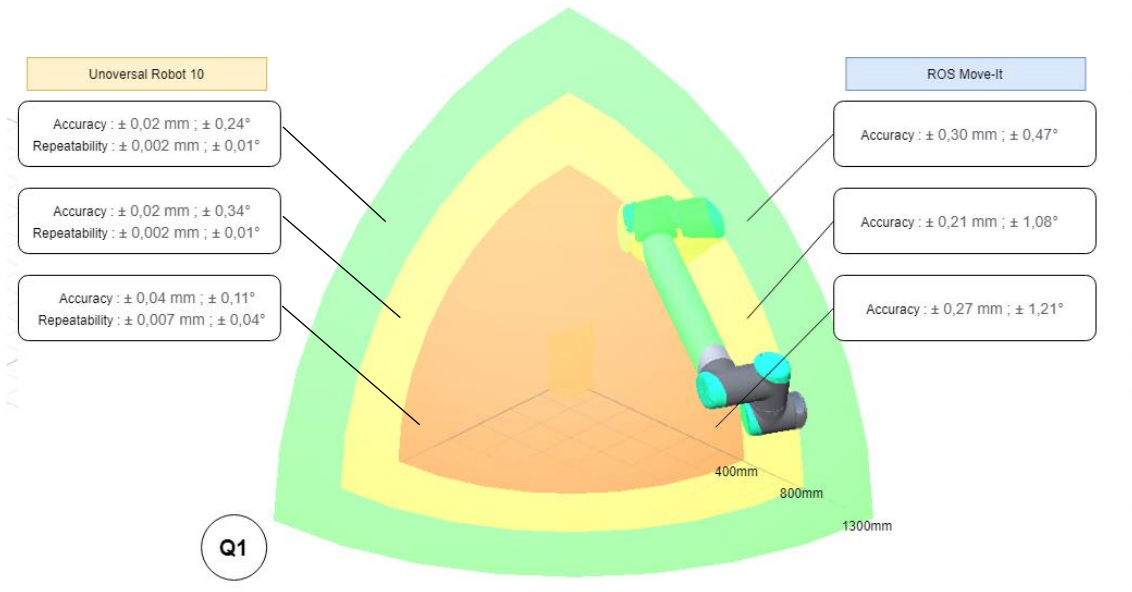


Figure 5-4 Results of the experiment 1 and experiment 2 on quadrant 1

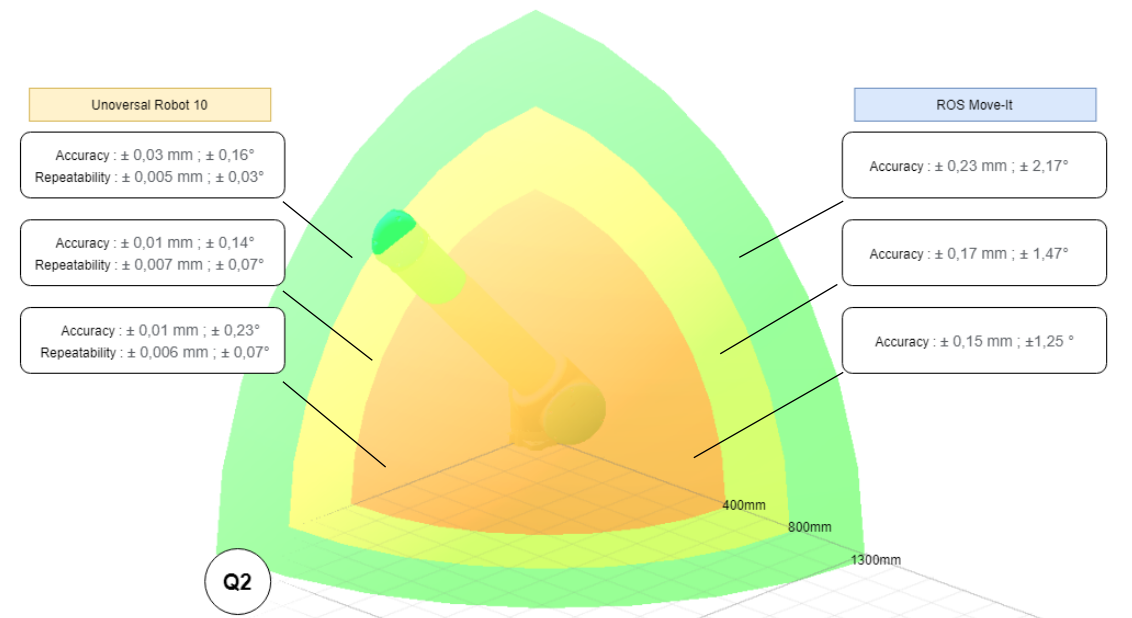


Figure 5-5 Results of the experiment 1 and experiment 2 on quadrant 2

In figure 5-4 and figure 5-5, the respective accuracy and repeatability value of the UR 10 robot and the automatic path generation of the Movelt algorithm on each reach zone are shown. Lowest registered value of the UR 10 robot is  $\pm 0,04 \text{ mm}$  translational and  $\pm 0,34^\circ$  rotational. Meanwhile, the repeatability is ten times as precise. It has a lowest precision of  $\pm 0,007 \text{ mm}$  and a lowest rotational precision of  $\pm 0,07^\circ$ . Both accuracy and repeatability of UR10 are almost constant for both quadrant and each zone. The measured lowest value of the accuracy and repeatability demonstrates the reliability and precision of the robotic system. This means the functional operation carried out by the UR10 contributes the least source of the error offset.

On the other hand, the accuracy of the ROS server side,  $\Delta E_{ROS}$  has a lowest translational precision of  $\pm 0,15 \text{ mm}$ . It registered a rotational as lowest as  $\pm 0,47^\circ$ . This measurement shows the stability and reliability of the ROS driver and the Movelt algorithm. It has overall an acceptable accuracy value for industrial scenario and does not vary significantly throughout each respective zone and quadrant. One of the reasons for this stable performance, is the choice of integrating the additional add-on package, Universal Robot ROS driver on the server-side as the default ROS driver. This package enables an industrial graded calibration file of the UR10 within the algorithm. This additional feature enables a more precise and smooth generation of joint motor commands on the ROS side. Overall, the usage of the Movelt algorithm as the main source of robot path generation is justified through this result. Furthermore, the offset error generate from the automatic path generation is also significantly low. Therefore, the final accuracy output of the application should be mainly influenced the MR functionality of the system,  $\Delta E_{MR}$ .

## Accuracy of the MR User Interface

Figure 5-6 shows the error offset caused by the overall system,  $\Delta E_{Total}$ . Three measurements are done successfully through the laser tracker. Based on the recorded measurement, it is clear that the  $\Delta E_{MR}$  is profoundly larger than the sum of the  $\Delta E_{Robot}$  and  $\Delta E_{MoveIt}$ . Therefore, expression (6) is altered as follows:

$$\Delta E_{MR} \approx \Delta E_{mean\_exp3} \quad (7)$$

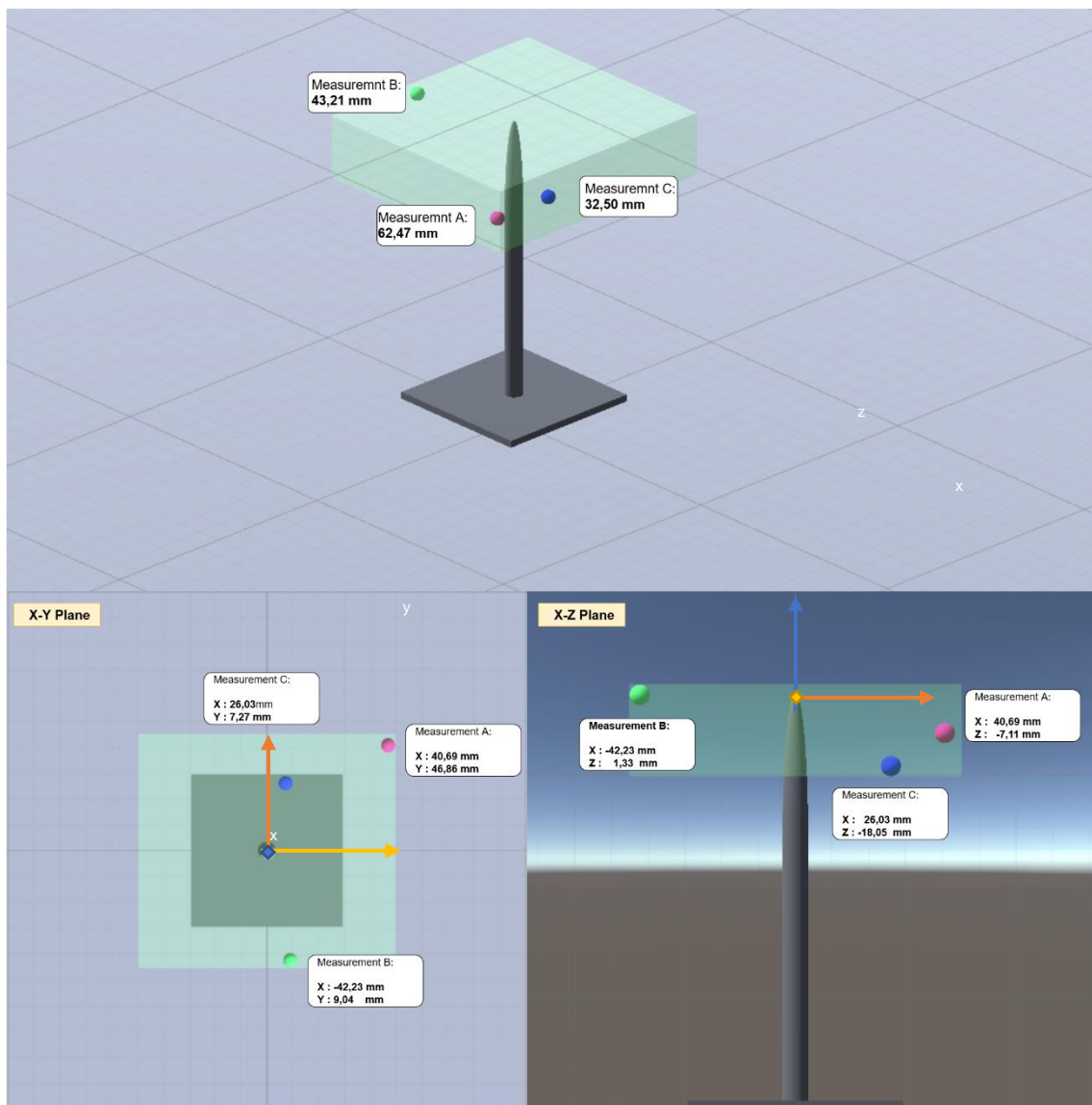


Figure 5-6 Final offset magnitude of measurements

The user had three trials to register a waypoint at the exact location of the pointer location. Measurement A shows the exact location of the first waypoint teaching attempt. It registered as the furthest waypoint from the actual goal at  $62,47\text{ mm}$  in the first measurement. The second attempt, measurement B is  $43,21\text{ mm}$  from target pose. Lastly the measurement C was the nearest where it was  $32,50\text{ mm}$ . Two important insights can be derived from these three measurements. First, the offset error from the goal point is decreasing with each consecutive attempt. This improvement of waypoint registration is due to the user's learning curve of the usage of the MR user interface and through intuitive strategy to implement the waypoint. This first insight will be further discussed in the qualitative performance analysis of this chapter.

The second insight is that the accuracy value of the MR application,  $\Delta E_{MR}$ , is at the rounded average of  $\pm 50\text{ mm}$  or  $\pm 5\text{ cm}$ . The accuracy does not meet the industrial standard for a robot operation. This value proves that the MR application layer undermines the precision and accuracy factor of the robotic system.

The chart in figure 5-7 shows the respective offset error of all three measurements on the 2-Dimensional plane, X-Y plane and X-Z plane. From the chart it can be seen that the variance from the respective average axis value measurement is lowest on the z-axis and highest on the x-axis. The variance on the y-axis is showing the trend of decrease and almost similar to the variance on the z-axis. One of the reasons for this is the parallax error caused by the limited perception of the camera point of view  $\Delta E_{Human}$ . It also includes the error of the MR registration,  $\Delta E_{MR\_only}$ . However, it is speculated that the  $\Delta E_{Human}$  is the major contributor as the methodology used for the MR registration like angle of the camera, light condition, distance of the MR target object are kept as constant as possible throughout the three trials. The variance on the x-axis is largest because, it is almost difficult to get a direct top view of the actual pointer poses during the experiment. The webcam position can be altered very well with an up and down direction and also in the left and right direction. However, a top and bottom view are almost impossible due to the hindrance of structure within the x-y plane. Therefore, it is more likely for a human parallax error to occur on this plane. From this limitation, the best approach to set a waypoint using a 'magic-lens' MR type is making sure the webcam poses at the isometric view of the pointer target location.

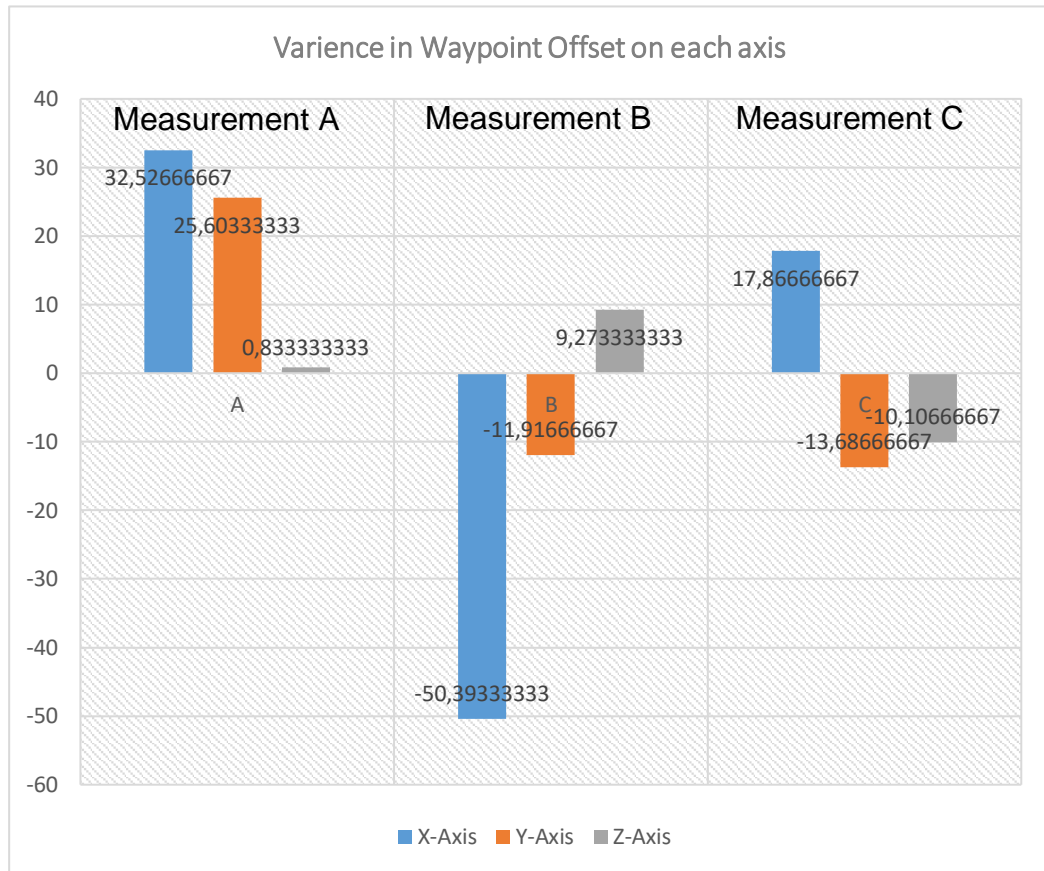


Figure 5-7 Offset variance from the respective mean value of each axis

Other than that, the figure 5-8 also shows a green box area which is enclosed by the maximum registered value of all three axes. This green box illustrates the use case when a target goal is an object with a planar surface instead of a pinpoint target. It also shows what kind of modification that the suction gripper needs to have in order to execute a proper and safe pick and place task. From the result, it can be seen that the MR application can pick up an object with a planar surface area  $79 \text{ cm}^2$  (the top view point surface) with a higher rate successful contact of the suction gripper from the top view. However, this success rate will decrease when a planar object with a heavier mass is set as a target. Furthermore, the suction gripper should have free movement with an offset of minimum  $-2 \text{ cm}$  in z -axis on its mechanical structure.

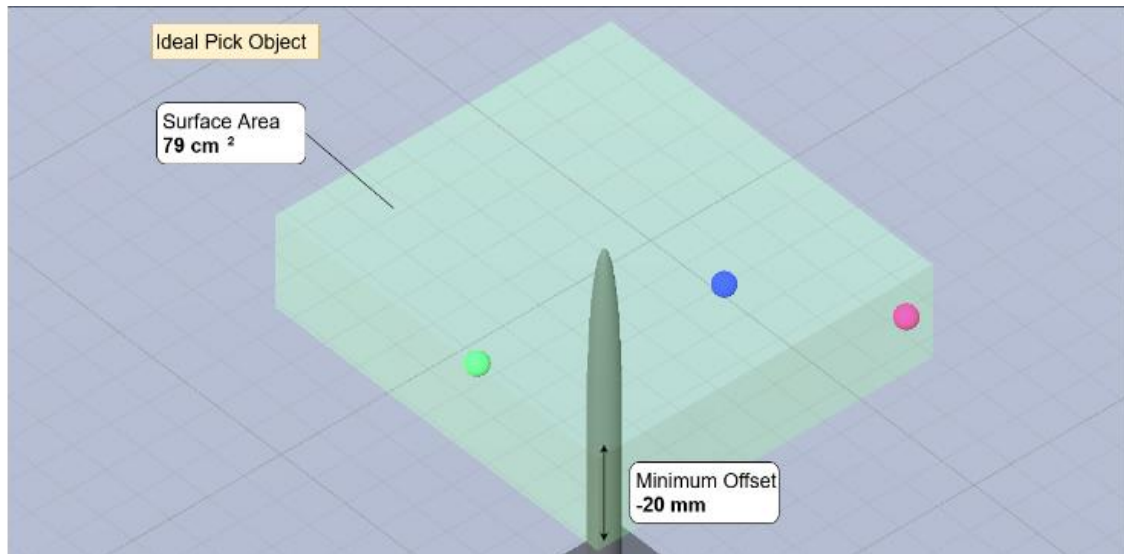


Figure 5-8 Estimation of pick object parameter

### Accuracy of the MR Registration

It has been noted in the previous section, that the accuracy of the waypoint registration varies strongly because of  $\Delta E_{Human}$ . However, the offset error of the MR user interface,  $\Delta E_{MR}$ , is also made of the MR registration error  $\Delta E_{MR\_only}$ . This can be seen on the digital twin robots drift pattern respective to the target viewing direction. During the experiment, the waypoint teaching procedure is only carried out when the user confirms that the digital twin robot is well superimposed on top of the real robot. This is done through visual inspection only. When the user recognizes that there is a significant drift in the registered digital twin, the MR registration process is restarted and recalibrated using the calibration module functionality. In this thesis, an explicit derivation of  $\Delta E_{MR\_only}$  is not obtained due to multiple sources of error and the difficulty to conduct a systematic study under the given lab environment. However, several insights are obtained during the phase of finding the optimum and successful MR registration of the digital twin robot. Those insights are outlined as follows:

#### Camera Pose Influence

The superimpose accuracy is higher with a direct view of the target object (the valve structure). The accuracy decreases when the angle is not directly on the target object. The reason for this is that the cluster of key point is registered and tracked more efficiently on the front view. The side view has less. The valve model has larger surface area on the frontal view than the side. This gives an insight for the usage of an object target with same dimensions. This is because the cluster of key points is significantly higher and well balanced on all the viewpoints. Alternatively, a multi-target system can be deployed where several arrays of target models are deployed within the MR scene. However, this

consumes computation as the CPU needs to stop the parallel track and recognize multiple targets and calculate the offset of the key point cluster.

### Lighting condition of test environment

The system performed differently according to the lighting condition of the lab environment. This insight is obtained from testing the MR application on different days with different lighting conditions of the lab environment. This factor shows that the system has high dependence on the lighting conditions because the lighting condition plays a role in how much light from the target module is being reflected back to the image sensor of the camera. The registration and pattern recognition of the key point cluster depends heavily on how the quality of perception of reality through the camera sensors is.

A potential fix is to eliminate the lighting parameter out of the system. This can be done by using a target system that emits an infrared spectrum. This spectrum should be captured back by an infrared instead of the RGB camera. This eliminates the problem when the target itself is the only entity emitting the infrared spectrum in the scenario. The registration and the tracking can be made quickly with less calculation algorithm, it can also produce a more stable and consistent tracking when the light condition varies through the whole process. The infrared marker target has a limitation when the robot is working in a high-temperature scenario. Therefore, Ultraviolet sensing technology can be investigated for a potential replacement.

This shows the importance of a more robust and reliable solution that is independent of the variation on the lighting condition which is very common in the industrial use case scenario. An example set-up is to use a custom target that can be made of arrays of infrared light-emitting diodes arranged in easily recognizable cluster geometry. The normal camera sensor can be replaced with a camera sensor that can sense both RGB and the infrared spectrum.

### The camera sensor

The performance of the system varies also when a different type of camera is integrated into the system. Two camera models were made available for this thesis, an HD webcam from Creative and the depth camera D435 from Intel. The intel depth camera's image sensor has a higher capability to capture the scene with a higher pixel definition than the conventional HD webcam. These pixels are converted to a one-dimensional array and submitted through the registration and tracking algorithm of the MR system. The target poses identified quickly and more accurately when a pixel array with distinct characteristics like higher contrast, brightness, and non-corrupted sensor reading is provided within the algorithm.

Therefore, the external camera used in the system should be chosen under consideration of the image quality which it can capture, and the quality of the camera build itself like the lens, camera sensor, and the data transmits technique.



The camera should be calibrated well, and the interests and extreme values should be supplied in the MR application. In this case, the VisionLib SDK's special functionality offers the best solution. The developer can integrate independent camera calibration values to enhance the tracking performance. This option is not available on the Vuforia and MRTK SDK.

### Target Set Up quality

It was found that, the target should be static and not in motion. The moving object creates an additional need for tracking the motion of the target object itself. This demands a higher rate of tracking algorithm loop within the MR system. A moving object can be used as a target given that the camera pose is static throughout the session. However, when both are in motion, an additional motion interpretation algorithm and technique should be implemented on top of the MR tracking system. This demands higher computational resource and produces less stable and constant MR registration.

## **5.2.2 Qualitative Performance**

In this section, the overall intuitiveness of the deployed system will be analyzed and discussed. In the previous section, the accuracy of the system is outlined clearly. However, the goal of this thesis is not only to focus on achieving an accurate waypoint teaching method, but also to design and develop a method that is easy and intuitive compared to the existing methods for completing a robot exercise like the pick and place task. In this section, several key points are discussed in order to pinpoint and benchmark the intuitiveness factor of the overall application.

### Less demand of robot programming knowledge and a simple MR UI

The intuitiveness of the application is benchmarked through a qualitative assessment. The user starts with a programmable digital twin within the scenario and can interact and manipulate the digital twin which in return carries out the desired changes in the robotic system. The user does not need to touch the teach pendant and program any form of code. Instead, the coding of the robot trajectory is interpreted from the manipulation of a virtual marker within the environment. The user is able to perform a pick a place task, having the control to teach waypoints which can produce the trajectory to reach and position to be handled. The intuitiveness of the programming is expanded with the capability of “on-demand preview” of the robot trajectory for the planned waypoints. This gives an opportunity for the user to make the correction and even improve the waypoint placement to plan a safe and effective robot trajectory. The user has the capability to activate and deactivate the end-effector of the robot through a simple button command within the MR.

These features are relevant for a user who has no knowledge of robot programming and wants to perform a robot task. The obstacle of the robot complexity is eliminated in the thesis developed system. However, it requires consistent usage to fully understand all the functionality of this system as seen in the measurement pattern of experiment 3. The learning curve is faster than a conventional robot programming method, when considering that the operator has little knowledge about robot or robots programming.

### Execution of a pick and place task

Figure 5-9 illustrates the successful pick and place task of the pick object, a model of a car, which was used for this thesis. For the first phase the user needs to place the virtual end-effector on top of the car module. The user needs to alter the camera viewpoint to eliminate any parallax error. It also needs to be noted, that the positioning of the virtual end-effector, in this case the suction gripper, should be positioned with an offset greater than 10 mm on the z-axis. This needed when the user executes the “pick “command the suction gripper will approach the object with an offset of 10 mm .This offset value is made possible with the spring mechanism attached to the suction cup.

It has been found that a minimum of three readjustment of the virtual end-effector are needed, due to the parallax error. The small surface area of the pick object also was a challenge to successfully place gripper. However, after the pick object is attached to gripper, the placement part of the task was carried out easier than the picking part. The car module is redirected towards a collection tray bin through teaching of waypoints within the MR. However, it should be acknowledged that the placement part will be difficult when the pick object needs to be placed at a pinpoint accuracy. For example, in an assembly process of many different parts.



Figure 5-9 Execution of a pick and place task with the MR application

### See-through MR vs Magic Lens MR

The level of immersion of the MR application does play a role in the intuitiveness factor of the system. The magic lens MR effect is achieved using the hardware configuration of a 2D display system and a webcam. The user experiences the MR effect only through a portal window size of the display. The interaction with the Virtual object occurs through the touch screen or mouse movement in the 2D plane of the screen. The user does not have the option to directly interact with the objects. One of the major limitations identified in the application deployed under the magic lens MR effect was the presence of the parallax error. Because magic lens MR provides the user with a 2-dimensional perception of the world the depth information is missing. This leads to multiple parallax errors during the waypoint teaching session. The user needs to shift his viewing point at least in two different viewing directions in order to perceive the depth and fix the parallax error. This not the case for the see-through MR as the depth is perceived directly from the environment. The waypoints can be freely and accurately placed within the environment compared to the magic lens MR effects. The detection of collision through the preview of the robot movement is faster compared to the magic lens effect MR. Furthermore, the user is able to grab and manipulate the 3D object within the MR scene with his bare hand through hand-tracking. This is a unique feature of the See-through MR provided by HoloLens 2. Furthermore, the user may start to perceive the digital twin as part of his reality faster than the magic lens effect. The user has more freedom to walk around and inspect the taught in waypoint within the scene and experience significantly less amount of parallax error. The magic lens effect frequently produces waypoints with parallax errors. This can be resolved by integrating a depth camera that can provide an accurate depth reading of the object.

### **5.3 Summary**

The developed MR robot programming system's performance was analyzed and studied using an experimental approach and using insights from observation remarks during the usage of the MR application. From the quantitative analysis of the system performance, it can be seen that the robotic system and the automatic path generation from the MoveIt algorithm have yielded a very reliable functionality. This also justifies the decision of integrating the MoveIt package as the path generation algorithm in chapter 3. The accuracy of the MR user interface is rounded to  $\pm 5 \text{ cm}$ . This value shows the inadequacy of the MR application to perform robot programming for tasks which demand pinpoint accuracy. It does not enhance or supports the robotic system's accuracy and repeatability factor. However, when the accuracy registration is reduced to a 2-dimensional plane, it has a reliable operational functionality for specific pick and place tasks. The developed MR application can be used for a pick and place task involving an

object with a relative larger surface area. With a larger surface area, the lack of accuracy demand can be compensated. It also needs to be noted that this depends on the type of gripper used as well. This compensation will not be valid if a form-based gripper is used instead of a suction gripper. In a nutshell, the MR application is reliable and functional to be used for a simple pick and place task that demands less accuracy and precision. The system is not ready for task which demand pinpoint accuracy and precision like part assembly processes.

On the other hand, the benchmarking of the intuitiveness factor of the application showed that the MR application does simplify the process of robot programming significantly. During the usage of the application, no specific form of robot programming knowledge or coding syntax was need as an input by the user. The user is able to command the robot to a specific place with only interacting with the MR layer. This is mainly due to the intuitive MR-based user interface tools developed specifically for a pick and place task. The MR application also provides a dynamic preview option where the user is able to see the programmed robot trajectory before it is executed on the real robot. The user is also given the control to modify and arrange the waypoints using visual feedback only. Moreover, there were no physical contacts which need to be made with the robotic system throughout the process. This is an ideal situation, when the robot is within an environment where the user can't be close with the robot and touch it. In a nutshell, all these features reduce the complexity of the robot programming significantly compared to the existing robot programming methods.

However, the previous statement manly valid for the user with no prior robot programming knowledge or experience. The intuitiveness should be further benchmarked in future studies, where a group of novice robot operators performs the same pick and place task using the MR system and the traditional method. The performance of the task should be benchmarked with the effectiveness of carrying out the task. Different pick and place scenarios with varying complexity can be planned. This will return benchmarking parameters like the programming speed and the number of successful pick and place tasks. This is important to find out whether the developed application can be seen as a complete replacement of the existing method or just as an additional assistant in the robot programming phase.

Overall, the following milestones are achieved with the final output of this thesis:

- A robot is successfully commanded with a MR application
- A pick and place task is conducted successfully with a small car model
- The setup is designed and deployed with minimum requirements and resources
- The system is designed in a way that it can be further modified and scaled easily

From this milestone achievement all primary functional requirements are fulfilled. The secondary functional requirements B1, B2, B3, B4, B7, B8, B9, B10 and B11 are fulfilled through the specific design and development of the path specification module, interaction module and the editing and preview module. Only some of the non-functional requirements for the overall system are tested and validated completely. Those requirements are C1, C2 and E1. The final client application was deployed as a Windows 10 standalone application and as a HoloLens 2 application. This means the requirement D1 which states that the MR application should be deployed in various platforms, is partially fulfilled for the platform UWP only.

In the following conclusion chapter, a concrete final verdict about the general idea of integrating the MR technology into a robot programming method will be outlined. The further steps and ideas for the improvement of the developed system performance and the system architecture are also presented.

## 6 Conclusion

Profound insights are obtained throughout this whole thesis project regarding the justification of MR technology integration in the robot programming field. These insights are obtained through addressing the problem statement, the lack of intuitiveness in existing robot programming methodology. An MR-based robot programming solution that can execute a pick and place task, is proposed and developed to address this problem statement. Important insights are drawn from validating and analyzing the performance of the final developed system through an experimental study.

As a final verdict, the mixed reality technology should be treated as a new form of displaying technology that can be used to minimize the complexity of the robot programming procedure. It should not be treated as a tool that enhances the accuracy or precision of the existing robotic system. The complexity of the robot programming is significant, but it can be reduced through interactive and information-rich three-dimensional visual feedback from the mixed reality technology. The integration of mixed reality technology within robot programming is valid only for user experience enhancement and should be used as a guide layer only. Therefore, with the existing state of the art of the mixed reality technology, robot task which demand pinpoint accuracy and precision, like welding, precise assembly, drilling, and machining, can't be performed. However, robot tasks which compensate low values of accuracy and precision, like simple pick and place tasks of box containers, surface vacuum cleaning, object hand-over task and educational demonstration, can be done more intuitively with MR technology integration in comparison to existing methodologies.

Overall, the only bottleneck which is hindering the full usage of the MR technology potential for robot programming is the accuracy and the stability of the mixed reality registration and tracking system within the environment. The accuracy and stability value from the state of the art are enough for entertainment and interactive educational fields, but not sufficient enough to meet the industry-standard yet. When this hindrance is addressed properly through a potential workaround, then the success factor of the developed MR robot programming system depends highly only on the user's skills and the degree of difficulty of the task.

Moreover, the aspect of accuracy and stability of the mixed reality technology is improving steadily through the decade with each new iteration of hardware set-up, sensor technology and software algorithm which the technology is build up on. Therefore, investing energy and resource in the idea of mixed reality technology for robot programming is a promising action, which can yield an innovative solution that can trigger a new revolution in the robotic field. Furthermore, this also compliments the concept of industry 4.0 where the user and the robotic system can share a digital symbiotic system and perform a complex industrial task through the developed MR application layer.

## Outlook

Several improvement proposals are laid out for future developments of this thesis work. These developments should focus on the performance aspect mainly, because it is identified as the persisting and unresolved challenge from this thesis. Other than that, several redesigns on the system architecture itself are proposed with consideration of the scalability and marketability of the concept.

An object registration and tracking technique which is very accurate and stable can profoundly increase the reliability of the system to carry out robot tasks profoundly. Instead of RGB-based mixed reality registration, a more industrial graded technique can be developed. For example, a depth sensor or laser tracking system to extract the pose of the object target at a high level of accuracy can be integrated. Alternatively, a custom MR marker object which delivers the most reliable and stable tracking can be designed and deployed within the system.

The existing user input action for the waypoint teaching produces a significant amount of parallax error especially when the application is deployed as a magic-lens type MR. The MR UI can be redesigned in such a way that it gives more information like the actual position of the virtual pointer tool. This information can be helpful for avoiding parallax errors. Alternatively, the see-through MR type application should be further developed to deliver a more immersive approach for the waypoint teaching. Furthermore, when an actual point cloud mesh of the environment can be obtained through an additional depth sensor, this point cloud can be integrated within the MR ecosystem and can be used to detect and prevent possible robot collision during the planning phase. A point cloud mesh can also be pushed back towards the Movelt algorithm to enable robot path generation with obstacle avoidance.

The final system is developed in such a way that it can be further developed for a scalable solution. In the developed system, the server-side handles all the robotic programming generation and sensor reading publishing. The client holds an MR application layer that captures the user input and promotes the viewing of the digital twin registered to reality. With further development on the server architecture, an “one to many” use case can be achieved, where multiple users on the client endpoint can use the one source of the server. This can be seen as robot program generation as service or RPaSS. The server is built on top of a Linux ecosystem and utilizes only a Wi-Fi network to communicate with the client. This means the application can be easily converted into a container schema like the Docker system and deployed into a cloud service. This is a potential scenario for an OPC unified architecture paradigm.

## References

- [1] T. Lozano-Perez, "Robot programming," *Proc. IEEE*, vol. 71, no. 7, pp. 821–841, 1983.
- [2] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer Science+Business Media, 2008.
- [3] R. Dörner, W. Broll, P. F. Grimm, and B. Jung, *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität*. Berlin, Heidelberg: Springer Vieweg, 2013.
- [4] D. Schart and N. Tschanz, *Augmented und Mixed Reality: Für Marketing, Medien und Public Relations*, 2nd ed. Konstanz, München: UVK Verlagsgesellschaft, 2018.
- [5] PTC, Vuforia Developer Library. [Online]. Available: <https://library.vuforia.com/>
- [6] Vuforia, Advance Model Target fro Augmented Reality. [Online]. Available: <https://library.vuforia.com/articles/Solution/trained-model-target-datasets.html>
- [7] S. K. Ong, J. W. S. Chong, and A. Y. C. Nee, "Methodologies for immersive robot programming in an augmented reality environment," in *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, Kuala Lumpur, Malaysia, 2006, p. 237.
- [8] H. C. Fang, S. K. Ong, and A. Y. C. Nee, "Novel AR-based interface for human-robot interaction and visualization," *Adv. Manuf.*, vol. 2, no. 4, pp. 275–288, 2014, doi: 10.1007/s40436-014-0087-9.
- [9] Y. S. Pai, H. J. Yap, and R. Singh, "Augmented reality–based programming, planning and simulation of a robotic work cell," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 229, no. 6, pp. 1029–1045, 2015.
- [10] J. Guhl, S. Tung, and J. Kruger, "Concept and architecture for programming industrial robots using augmented reality with mobile devices like microsoft HoloLens," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1–4.
- [11] S. Blankemeyer, R. Wiemann, L. Posniak, C. Pregizer, and A. Raatz, "Intuitive Robot Programming Using Augmented Reality," *Procedia CIRP*, vol. 76, pp. 155–160, 2018.
- [12] 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)*, 2018, pp. 1–9.



- [13] Camilo Perez Quintero, Sarah Li, Matthew KXJ Pan, Wesley P. Chan, H.F. Machiel Van der Loos, Robot Programming Through Augmented Trajectories in Augmented.
- [14] D. Bambušek, Z. Materna, M. Kapinus, V. Beran, and P. Smrž, "Combining Interactive Spatial Augmented Reality with Head-Mounted Display for End-User Collaborative Robot Programming," in 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), 2019, pp. 1–8.
- [15] Mulun Wu and Shi-Lu Dai and C. Yang, Mixed reality enhanced user interactive path planning for omnidirectional mobile robot.
- [16] A. Cao, A. Dhanaliwala, J. Shi, T. Gade, and B. Park, "Image-based marker tracking and registration for intraoperative 3D image-guided interventions using augmented reality," Aug. 2019. [Online]. Available: <https://arxiv.org/pdf/1908.03237>
- [17] C. Kunz et al., "Infrared marker tracking with the HoloLens for neurosurgical interventions," Current Directions in Biomedical Engineering, vol. 6, no. 1, 2020, doi: 10.1515/cdbme-2020-0027.
- [18] M. Cousins, C. Yang, J. Chen, W. He, and Z. Ju, "Development of a mixed reality based interface for human robot interaction," in Proceedings of 2017 International Conference on Machine Learning and Cybernetics: Crowne Plaza City Center Ningbo, Ningbo, China, 9-12 July 2017, Ningbo, China, 2017, pp. 27–34.
- [19] ROS. [Online]. Available: <https://www.ros.org/>
- [20] ROS Sharp. [Online]. Available: <https://github.com/siemens/ros-sharp>
- [21] ROS Industrial. [Online]. Available: <https://rosindustrial.org/>
- [22] MoveIt. [Online]. Available: <https://moveit.ros.org/documentation/concepts/>
- [23] Microsoft, Windows Subsystem for Linux. [Online]. Available: <https://docs.microsoft.com/en-us/windows/wsl/install-win10>
- [24] Universal Robots, Universal Robots ROS Driver. [Online]. Available: [https://github.com/UniversalRobots/Universal\\_Robots\\_ROS\\_Driver](https://github.com/UniversalRobots/Universal_Robots_ROS_Driver)
- [25] ros-industrial, ROS-Industrial Universal Robot meta-package. [Online]. Available: [https://github.com/ros-industrial/universal\\_robot](https://github.com/ros-industrial/universal_robot)
- [26] Vuforia PTC, Developer Portal. [Online]. Available: <https://developer.vuforia.com/>
- [27] VisionLib, Software Development Kit Documentation. [Online]. Available: <https://visionlib.com/>
- [28] Microsoft, Mixed Reality Toolkit. [Online]. Available: <https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/>

## Appendix A: Guideline for ROS Workspace Setup

### Prerequisites

First of all, the project needs these supporting packages to be installed. The instruction assumes Catkin workspace has already installed in the ROS. The ROS ecosystem used for this thesis installed on top of the Ubuntu 16.04

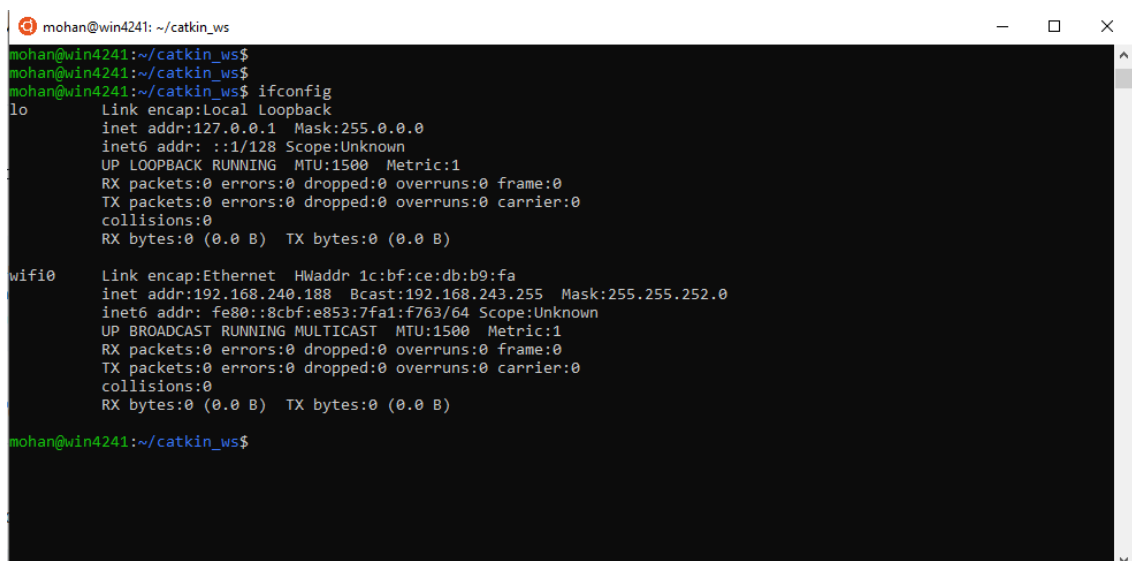
1. Install the source folder (src) in the catkin\_ws
2. This folder can be found in the DVD at following directory path:  
*Software/ROS packages/*
3. Then open a bash terminal.
4. Then input the following commands in the bash terminal

```
• cd catkin_ws
• rosdep update
• rosdep install --from-paths src --ignore-src --rosdistro kinetic
• catkin_make
• source devel/setup.bash
```

### The ROS Server IP Port Address Inquiry

After a successful installation of all the dependency from the ROS packages, the IP address of the local computer which running the ROS need to be obtained. Type the following command in a new bash

- ```
• ifconfig
```



```
mohan@win4241: ~/catkin_ws
mohan@win4241:~/catkin_ws$
mohan@win4241:~/catkin_ws$
mohan@win4241:~/catkin_ws$ ifconfig
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Unknown
            UP LOOPBACK RUNNING  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wifio      Link encap:Ethernet  HWaddr 1c:bf:ce:db:b9:fa
            inet addr:192.168.240.188  Bcast:192.168.243.255  Mask:255.255.252.0
            inet6 addr: fe80::8cbf:e853:7fa1:f763/64 Scope:Unknown
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mohan@win4241:~/catkin_ws$
```

## Launching the ROS Server

In order to launch the ROS server open up 5 bash terminals. Run the following commands separately of the opened five bash terminals. Take note that the second launch command has the `robot_ip` input. Please remove this IP address and replace with your robot IP address. The robot IP address can be found in the teach pendant (refer appendix C)

- `roslaunch file_server ros_sharp_communication.launch`
- `roslaunch ur_robot_driver ur10_bringup.launch`  
`robot_ip:=10.43.49.`  
`kinematics_config:=/root/my_robot_calibration.yaml`
- `roslaunch ur10_moveit_config`  
`ur10_moveit_planning_execution.launch limited:=true`
- `roslaunch moveit_tutorials mar_master.py`
- `roslaunch moveit_tutorials gripper.py`

## Appendix B: Guideline for Importing the Unity Project

### Prerequisites

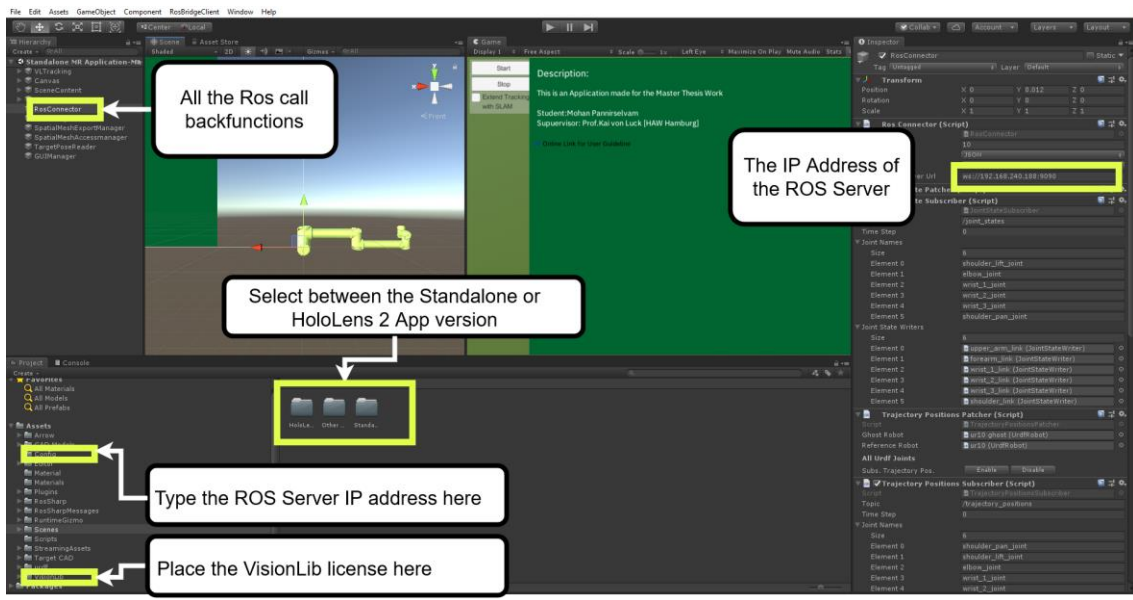
The following unity version and SDK version are needed for this project

|               |             |
|---------------|-------------|
| Unity version | 2018.4.26f1 |
| VisionLib SDK | 20.11.1     |
| MRTK SDK      | 2.4.0       |

### Opening the Unity Project

1. Open the unity project in the DVD at the directory path:  
/Unity Project File/MR Unity Application
2. In this project file, the VisionLib SDK is already preinstalled. Remove it if a newer version is wanted to be used.
3. Acquire the VisionLib license file and place it in the VisionLib folder
4. For the HoloLens 2 version, import the MRTK SDK file manually.

### Project Navigation



## Appendix C: Guideline for Configuring the Universal Robot

### Prerequisites

|                 |             |
|-----------------|-------------|
| Universal Robot | UR10        |
| Polyscope       | Version 5.1 |

### Installing a URcap through Teach Pendant

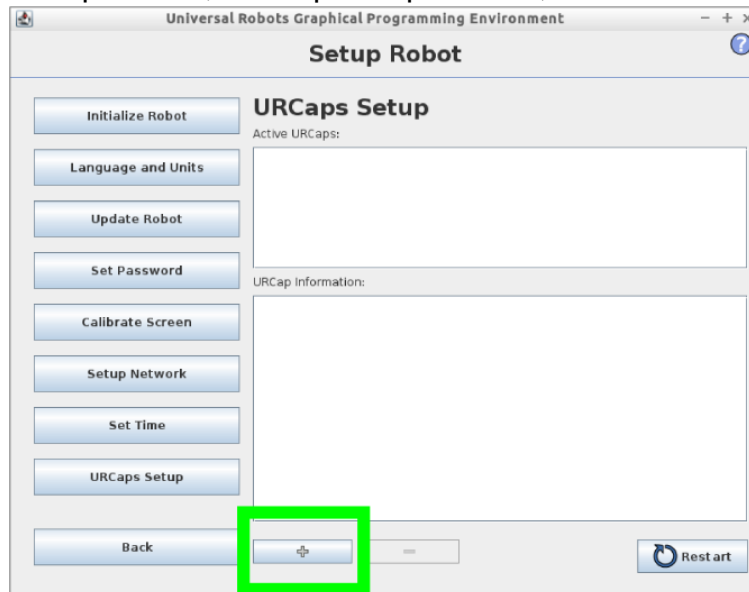
1. For using the *ur\_robot\_driver* with a real robot, the **externalcontrol-1.0.5.urcap** need to be installed. This can be found in the DVD directory path:URCap/
2. Copy the URcap file in a USB stick
3. Attach this USB stick in the teach pendant



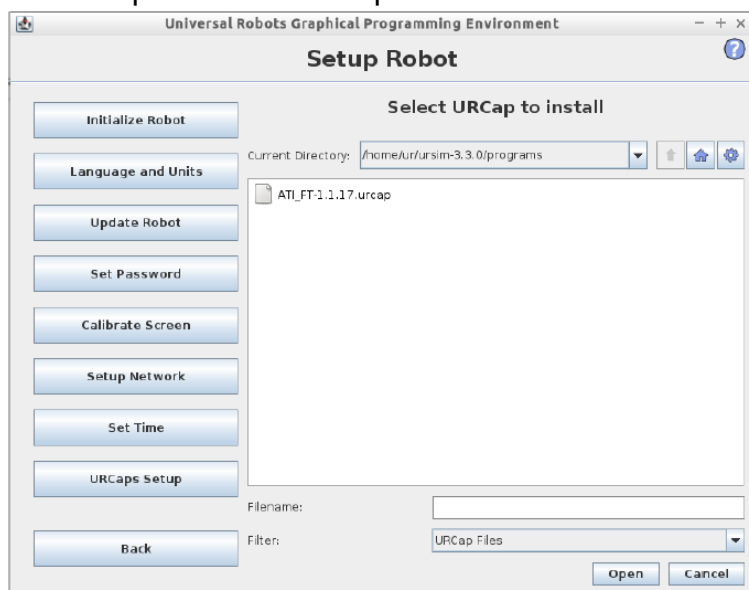
4. On the “Setup Robot” screen, click URcap Setup.



5. On the “Setup Robot”, “URcap Setup” screen, click +.



6. Select the URcap file and Click Open.



7. Click Restart after successful installation. Remove the USB stick.



## Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen.“

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.

Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

### Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: \_\_\_\_\_

Vorname: \_\_\_\_\_

dass ich die vorliegende \_\_\_\_\_ – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

*- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -*

Die Kennzeichnung der von mir erstellten und verantworteten Teile der \_\_\_\_\_ ist erfolgt durch:

\_\_\_\_\_ Ort

\_\_\_\_\_ Datum

\_\_\_\_\_  Unterschrift im Original