

이화여자대학교 대학원  
2018학년도  
석사학위 청구논문

An Intuitive Robot Trajectory  
Simulation Interface for Unstructured  
Environment using See-Through AR

컴퓨터공학과

Hyunjung Kim

2019

# An Intuitive Robot Trajectory Simulation Interface for Unstructured Environment using See-Through AR

이 논문을 석사학위 논문으로 제출함

2019 년 6 월

이화여자대학교 대학원

컴 퓨 터 공 학 과 Hyunjung Kim

# Hyunjung Kim 의 석사학위 논문을 인준함

지도교수 김 영 준 \_\_\_\_\_

심사위원 박 현 석 \_\_\_\_\_

오 유 란 \_\_\_\_\_

김 영 준 \_\_\_\_\_

이화여자대학교 대학원

## Table of contents

<b>I. Introduction.....</b>	<b>1</b>
1.1. Motivation.....	1
1.2. Research Goal.....	2
1.3. Challenges.....	3
1.4. Main Contributions.....	4
<b>II. Related Work.....</b>	<b>5</b>
2.1. Robot Programming Techniques.....	5
2.1.1. Online Programming.....	6
2.1.2. Offline Programming.....	8
2.2. Robot Simulation Interface with Augmented Reality.....	10
2.2.1. Augmented Reality in Robotics.....	10
2.2.2. Offline Robot Programming using AR.....	11
<b>III. System Overview.....</b>	<b>12</b>
3.1. System Architecture.....	12
3.2. Gesture Interfaces in HoloLens.....	14
3.3. Coordinate System Transformation.....	15
3.4. Network System.....	16
<b>IV. Teaching by Augmented Reality.....</b>	<b>17</b>
4.1. Mapping Real World to Virtual World.....	17
4.1.1. Calibrating the Virtual Robot.....	17
4.1.2. Environment Mapping.....	18
4.2. Input Interfaces in AR.....	23
4.2.1. Moving the Tool pose .....	23
4.2.2. Intuitive and Rich Input Feedback.....	24

4.2.3. Setting the Start and Goal States.....	26
4.3. AR Simulation.....	27
<b>V. Implementation.....</b>	<b>29</b>
5.1. Implementation Platform.....	29
5.1.1. AR Platform.....	29
5.1.2. Robot Platform.....	29
<b>VI. Experiment.....</b>	<b>30</b>
6.1. Task and Environment.....	31
6.2. Participants and Procedure.....	32
6.3. Measures and Analysis.....	33
6.4. Results.....	34
<b>VII. Conclusion.....</b>	<b>39</b>
<b>Bibliography.....</b>	<b>40</b>
<b>Abstract (in Korean).....</b>	<b>44</b>

## List of Figures

<b>Figure 1.</b> The American Occupational Safety and Health Administration defines three methods for programming the robot [9].....	5
<b>Figure 2.</b> KUKA smartPAD [10].....	6
<b>Figure 3.</b> Teaching by Demonstration.....	7
<b>Figure 4.</b> Graphical user interface [21].....	9
<b>Figure 5.</b> System Components.....	12
<b>Figure 6.</b> System Architecture.....	13
<b>Figure 7.</b> Air tap. Left: finger in the ready position. Right: press finger down to tap or click.....	14
<b>Figure 8.</b> Coordinate frames used by the interface.....	18
<b>Figure 9.</b> Overlaid with a virtual robot.....	18
<b>Figure 10.</b> Spatial mapping.....	19
<b>Figure 11.</b> Frame mapping.....	19
<b>Figure 12.</b> Mapped virtual environment.....	21
<b>Figure 13.</b> Planning in virtual space.....	22
<b>Figure 14.</b> Pose control interface.....	23
<b>Figure 15.</b> Rotation control gesture.....	24
<b>Figure 16.</b> Robot 7 joint frames.....	25
<b>Figure 17.</b> AR simulation of the path between the start pose and goal pose of the tool.....	27
<b>Figure 18.</b> AR simulation.....	28
<b>Figure 19.</b> Robot execution.....	28
<b>Figure 20.</b> Two interface conditions.....	30
<b>Figure 21.</b> Experimental environment.....	31

**Figure 22.** Comparison of RViz and our interface based on user data.....38

## List of Tables

<b>Table 1.</b> Assessment item.....	33
<b>Table 2.</b> User study data.....	38

## Abstract

A collaborative robot was created as a result of recent developments in robotic technology. Its advantages of small size and easy to handle frequent re-programs, without the need for a robotic expert, are useful in small and medium sized enterprises (SMEs). However, there is a risk of collision with the robot, and to overcome this danger an interface system that enables the operator to safely, easily, and intuitively work with collaborative robots is needed. In this dissertation, we present an intuitive robot simulation interface using untethered see-through augmented reality (AR) to improve the human-robot interaction (HRI) of a non-expert user in an unstructured environment. The intuitive robot simulation also supports safe and effective robot programming. To achieve this, we propose a new concept of teaching robots: teaching by AR, which uses AR to teach the robot from the user's demonstration through a virtual robot without contact with the real robot. We used offline robot programming using the AR approach and provided intuitive visual feedback to inform the validity of the user input in real-time in AR. Through AR head-mounted displays (ARHMD), we supported a user to see a view matching the virtual simulation interface on the user's physical workspace, and we provided mapping of the work cell environment for collision-free path planning. Furthermore, we conducted a user study in which participants were asked to teach a robot a program using our simulation interface and RViz. The results of the study demonstrate that our interface is intuitive, time efficient, and helps the user to teach the robot.

# I. Introduction

## 1.1. Motivation

Due to recent developments in robotic technology, various safe and high-performance robot hardware have been developed. A new robotic concept has been created called a collaborative robot, or cobot as it is also known. As industry 4.0 progresses, the collaborative robot market is growing rapidly, and it is expected to become the core engine of future industrial robots. The collaborative robot refers to a new type of industrial robot that can be used in the same space as a worker without need for installation of a safety fence, which must be applied to a conventional industrial robot.

An early collaborative robot has been developed to solve the reality of SMEs, whose production environment has multiple types of small-volume production forms, small workspaces, and small capital, in contrast to larger enterprises. Existing industrial robots have been unable to handle the frequent replacement of work programs. In addition, they require a wide workspace to install safety fences and are therefore costly. However, the collaborative robot developed for SMEs can work in the same space as the worker and can be deployed immediately without any major changes to the workspace being required. Even if there is no robot engineer for programming the work instruction, in the case of a simple job, the robot can be used for new work within a few minutes. These features make the collaborative robot highly attractive.

Good safety standards are required for people working with collaborative robots. Many collaborative robots are equipped with the mode of operation in which the robot immediately stops when a certain amount of power or force is detected. However, there is still a risk of collision, and since the collaborative robot is still in its early stages of introduction, there are currently insufficient safety guidelines. Therefore, an interface system is needed that enables the operator to safely, easily, and intuitively input task information into the collaborative robot

in order to execute the task. This could be resolved if the worker can obtain the status of the robot and situation awareness feedback from the interface, and if a safe, easy, and convenient method of task execution is supported.

Many studies have pointed out the promise of AR in robotics. The advantages of AR characteristics include improving SA by reducing cognitive workload and facilitating appropriate decision making by combining virtual information into a real-world situation [1], [2], [3], [4]. Recent advances in sensor technologies, display technologies, etc. have resulted in a see-through AR device that enables the user to leverage all the potential of AR as HRI interfaces. This frees AR from 2D-restricted devices such as tablets, monitors, projectors, etc., which have hardware limitations [5], [6], [7].

## **1.2. Research Goal**

Our goal is to propose an intuitive robot trajectory simulation interface for the collaborative robot using untethered see-through AR to improve the HRI of non-expert users in an unstructured environment, and to support safe and effective robot programming.

Since it takes considerable time to learn how to program a robot, we facilitated intuitive and helpful robot programming using the AR technique for a user who had no knowledge of robotics. We tested given input from the user to evaluate whether it is valid or not by checking that the input is within a reachable space of a robot and confirming if collision-free path planning is possible with the given input. In addition, we gave the user intuitive visual feedback for a validity test in AR, showing the movements of the robot corresponding with the given input to improve the SA of the user. When the user had decided on the desired start and goal states of the robot in the former step as inputs, we provided an AR simulation that is a path through the given states.

Due to the attention required when looking alternately at the separated output spaces, simulation interface, and work cell where the robot exists, the cognitive workload of the user is

increased. To reduce the user's cognitive workload, we provided the user with a virtual interface that provides a matching view of the virtual simulation interface onto the user's physical workspace. We achieved this by using marker tracking through see-through ARHMD.

Furthermore, a user in an unstructured environment, where there is a frequent need to change the worktable environment and re-program the robots, needs to update the spatial information in every change. To reduce the user's workload, we automatically mapped the environment of the work cell through spatial mapping of ARHMD.

### 1.3. Challenges

***AR simulation:*** we needed to show the simulation set by the user of the robot trajectory simulation from the start state to the goal state of the robot.

***Real-time visual feedback for valid user input:*** we needed to show the user's input feedback to the user in real-time to inform them whether the input is valid or not.

***Mapping the surrounding environment:*** we needed to know the surrounding environment information of the workspace for robot simulation. When the user sets a desired start and goal state and wants to plan a path, collision information of the work cell is required to plan well.

***Real world to virtual world transformations:*** we needed to match the coordinate frame of the virtual interface to the coordinate frame of the actual work cell to overlay the virtual robot onto the real robot.

***System integration:*** an integrated communication system is needed for sending and receiving data between three separate system elements: ARHMD, which is a user interface device to interact with a robot, the server running on Robot Operating System (ROS), which is for processing user requests between ARHMD and a robot, and a 7-DoF industrial robot.

## **1.4. Main Contributions**

This dissertation makes the following main contributions:

First, we propose a new concept of teaching robots: teaching by AR, is a similar concept to teaching by demonstration in terms of moving a robot in a real work cell, but is different in that it does not have any contact with the robot itself.

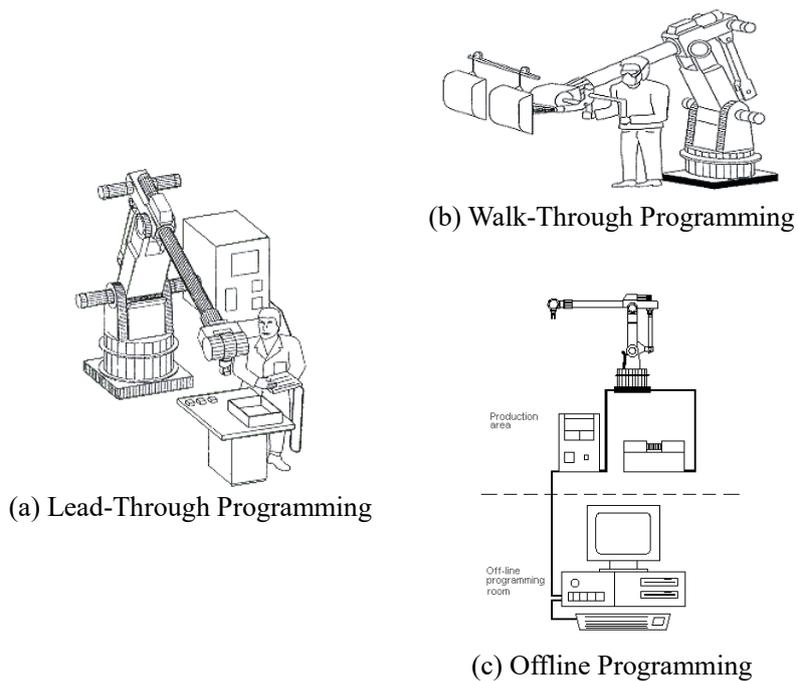
Second, we introduce a prototype of a new AR robot simulation interface based on the untethered see-through ARHMD, Microsoft HoloLens, to improve the HRI of a non-expert user in an unstructured environment, and to support safe and intuitive robot programming.

Third, we demonstrate that our interface improves the user's HRI through the results of a user study, which proves that our simulation interface is intuitive, easy to use, and time efficient.

## II. Related Work

### 2.1. Robot Programming Techniques

Today's industrial robot programming techniques can be broadly divided into online and offline programming [8]. Frequently used online programming techniques are lead-through programming and walk-through programming (Figure 1). In this dissertation, we follow the definition that classifies robot programming techniques from The American Occupational Safety and Health Administration.



**Figure 1.** The American Occupational Safety and Health Administration defines three methods for programming the robot [9]

### 2.1.1. Online Programming

Online programming occurs when the robot is online or in an operational mode. Online programming is done at the production site and requires access to the equipment and the robot.

In lead-through programming, the control of the robot occurs when it is in a teach mode and this programming method uses a proprietary teach pendant (e.g. a joystick or a touchpad) (Figure 2). To program the robot, the operator uses the buttons on the pendant to move the robot to point-to-point and store each position separately (Figure 1-(a)). When the whole program has been learned, the robot can play back the points. This teaching method is the most traditional, familiar, and basic method.

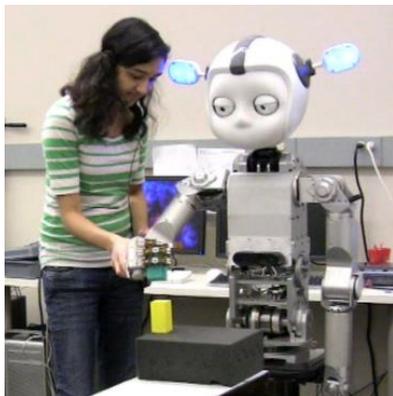
However, with this method, the operator needs to teach every single point individually, making it very cumbersome and time-consuming [11]. In addition, it is unintuitive because of the modality of the teach pendant. It is not straightforward to adjust a robot arm that has 6-degrees of freedom (DoF) or higher with a 2-DoF joystick or touchpad. Thus, lead-through programming is only reasonable if the program is not changed very often, and if it is relatively simple, with only a few points.



**Figure 2.** KUKA smartPAD [10]

In walk-through programming, the operator is in direct physical contact with the robot and moves the robot to the desired positions (Figure 1-(b)). While the robot is operating, its controller scans and stores the movements of each joint at frequent intervals. When the robot is later placed in the automatic mode of operation, these values and other functional information are replayed, and the program run as it was originally taught.

According to the robot programming definition of the American Occupational Safety and Health Administration [9], teaching by demonstration (TbD), specifically kinesthetic teaching, is also included in walk-through programming, from the perspective that TbD is a programming technique where the operator moves the robot through physical contact (Figure 3). It is faster and more intuitive than using the teach pendant in lead-through programming and requires minimal programming skills. TbD has acquired much research interest as it is easy to use, requires few skills, and has recently become widely used with the help of a fully compliant robot manipulator that has advanced force-torque sensing with variable stiffness [12], [13], [14], [15], [16], [17].



(a) A robot interacting with a teacher [13]



(b) An operator interacting with the robot and piloting it to a coordinate [11]

**Figure 3.** Teaching by Demonstration

Although the TbD method allows intuitive robot programming, it is not good for tasks that involve moving the robot correctly along the desired trajectory. Moving the robot by hand would be arduous and inaccurate for such a task. Furthermore, since the user is physically in contact with the robot as well as within the robot's scope of the work, the operator is exposed to the risk that the robot may collide with the operator who is not protected by the safeguarding devices being deactivated. These safety issues may cause operators to worry about their work.

### **2.1.2. Offline Programming**

Offline programming (OLP) can overcome the deficiencies of online programming, a time consuming and out of production state [18]. OLP is the process of producing a program through simulation before it is run on an actual robot using a virtual mockup of the robot and task (Figure 1-(c)). OLP allows the operator to develop the robot programs without having to access the robot directly. OLP, or simulation, is most often used in robotics research to ensure that new control algorithms are working correctly before moving them onto a real robot. In addition, OLP can reduce on-site programming time and downtime of manufacturing equipment [19]. It can be a particularly useful method for SMEs, where the robots are more likely to be reconfigured multiple times compared to mass production environments. With OLP, it is also easy to test several different approaches to the same problem, which would be inefficient for online programming methods. However, in OLP, there is a mapping problem in moving the physical working environment to the virtual world to simulate the robot. This makes it difficult to do so where the working environment changes frequently [6].

For OLP, it is important to build an easy-to-use interface (Figure 4) to help the user program the manipulator [20]. To do that, a couple things must be considered when implementing an interface. Firstly, how to make the interface easier for the user to program without seeing the actual robot, and secondly, how to make sure that the user can specify the robot program well by communicating with the computer graphic elements in the interface using a pointing device such as a mouse or light pen. These need to be designed well to help non-programmers. If the simulation interface is intuitive to use, this can be a quick way to test an idea before moving it

to the robot.



Figure 4. Graphical user interface [21]

## **2.2. Robot Simulation Interface with Augmented Reality**

### **2.2.1. Augmented Reality in Robotics**

AR is an interactive reality in which virtual information is overlaid onto a real world. AR differs from virtual reality (VR) in that, in VR the entire real world is replaced by computer graphics, and AR enhances reality rather than replacing reality. Azuma et al. [22] note that there are three main characteristics of an AR interface:

- 1) They combine real and virtual objects.
- 2) The virtual objects seem to be registered in the real world.
- 3) The virtual objects can be interacted in real time.

Green et al. [4] state that AR is an ideal platform for human-robot collaboration because it provides the following important qualities:

- 1) Ability to improve reality.
- 2) Seamless interaction between real and virtual environments.
- 3) Ability to visualize the robot based on the workspace (exo-centric view).
- 4) Spatial cues for local and remote collaboration.
- 5) A tool for enhanced collaboration, especially for multiple people collaborating with a robot.

These attributes enable AR to support natural spatial conversations by showing the visual cue needed for humans and robots to match each other's intentions and maintain situational awareness.

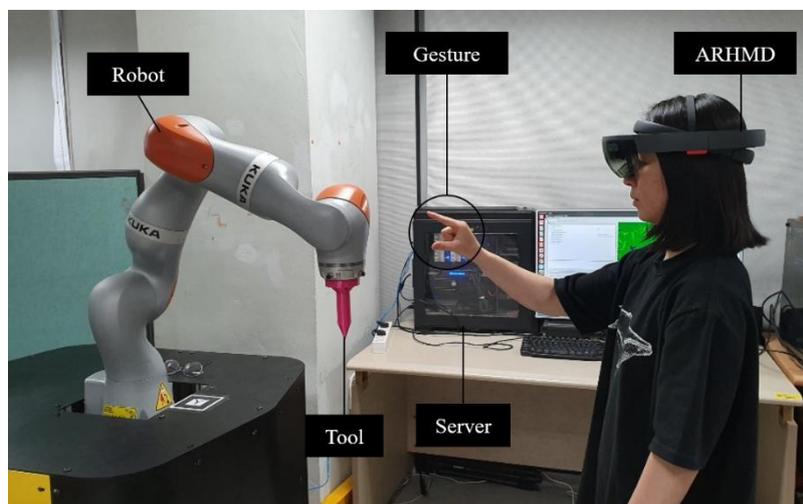
### **2.2.2. Offline Robot Programming using AR**

Many studies have paid attention to the merits of AR mentioned above and applied AR as a tool for HRI [1], [5], [6], [23], [24]. Among other topics, studies have been conducted on offline robot programming using AR [7], [18], [19], [25], [26]. OLP using the AR technique allows offline robot programming without the need to model environment elements in a virtual environment. Because OLP using AR can augment the virtual robot on the actual working cells, it is also useful when approach to the robot is needed. This approach can eliminate many technical difficulties associated with calibration issues between the virtual world and the real world [17]. OLP using AR delivers some of the OLP's unique benefits, such as the need to separate physical robots from production, while maintaining safety and operational benefits.

### III. System Overview

This section briefly describes our presented simulation interface system. More details are introduced in next chapter.

#### 3.1. System Architecture



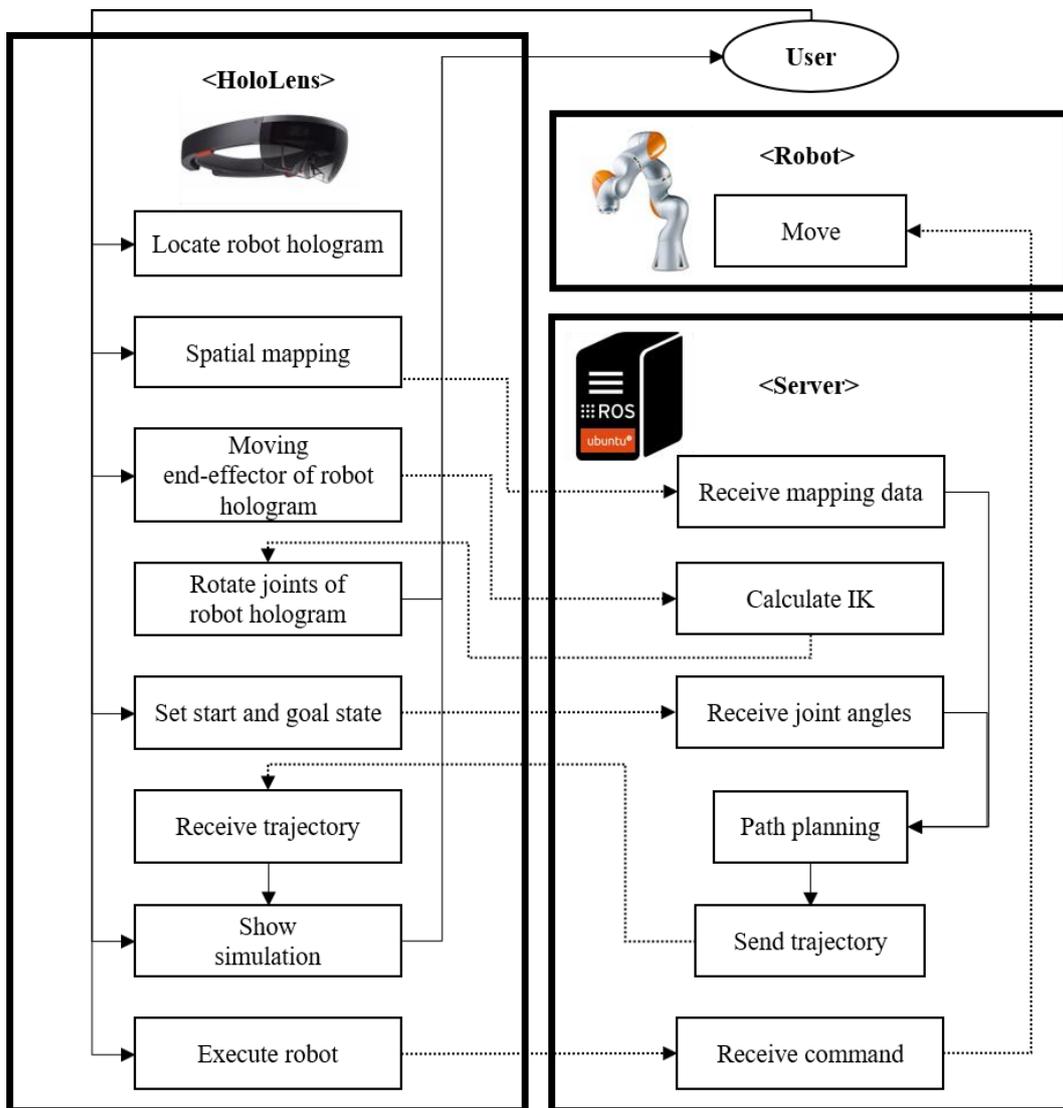
**Figure 5.** System Components

Our proposed simulation interface system interacts with the user as shown in Figure 5. The system is composed of a wireless ARHMD, Server, and Robot. The system takes the user's gestures as input and shows them virtual objects through the AR interface in ARHMD.

As shown in Figure 6, through the AR interface, spatial mapping of the surrounding environment is performed, and the user can then give the start and goal poses of the robot by moving the end-effector of the virtual robot instead of moving the actual robot. When the user moves the end-effector of the virtual robot, the pose of the tip of the tool attached to the end-effector is sent to the server. The server computes corresponding joint states of the robot to the

received pose and sends joint states to HoloLens. This process is repeated in real-time so that the user can see the movements of the virtual robot while moving its end-effector.

The server running on ROS calculates the trajectory based on the received poses and mapping data and sends it back to the HoloLens so that the user can see the simulation of the trajectory. With this simulation, the user can consider whether to execute the robot or not.



**Figure 6.** System Architecture

### 3.2. Gesture Interfaces in HoloLens

In HoloLens, hand gestures use head gaze as the targeting cursor. HoloLens recognizes the air tap, which is a tapping gesture with the hand-held upright, like a mouse click or select (Figure 7). With this air tap, the user can perform a tap and hold, which is equivalent to the click and drag action with a mouse. By combining tap, hold, and release with the movement of the hand, the user can drag and move, and resize or rotate, a virtual object from the user's hand movements.



**Figure 7.** Air tap. Left: finger in the ready position. Right: press finger down to tap or click

### **3.3. Coordinate System Transformation**

We used the game engine Unity3D to create an environment for developing the HoloLens application, and we used ROS for the robotic development platform. Since Unity3D uses a left-handed coordinate system unlike ROS, we had to deal with different coordinate systems between Unity3D and ROS. To address this, we converted the coordinate system of the data received from ROS in the HoloLens application from the right-handed coordinate system to the left-handed coordinate system. On the other hand, if ROS received data from the HoloLens, we changed the coordinate system from the left-handed coordinate system to the right-handed coordinate system.

### **3.4. Network System**

In our system, as HoloLens, the server, and robot are separated devices, we needed to set up a network for them to communicate with each other. There are two network connections where one is a socket communication between HoloLens and the server and the other provides communication between the server and the robot. Since we can take advantage of the fact that HoloLens is an untethered AR device, wireless communication was used between HoloLens and the server. As shown in Figure 6, several data are exchanged between HoloLens and the server. The spatial mapping data is passed by TCP socket communication for reliable transmission. The pose of the tool tip, joint states of Inverse Kinematics (IK) solution, the start and goal states, calculated trajectory, and command of real robot execution are all transmitted by UDP socket communication. For communication between the server and the robot, only the execution command is sent to the robot by the TCP communication of ROS nodes. We also considered different OS when setting up the communications. We used Windows based socket communication for HoloLens and Linux based socket communication for the server.

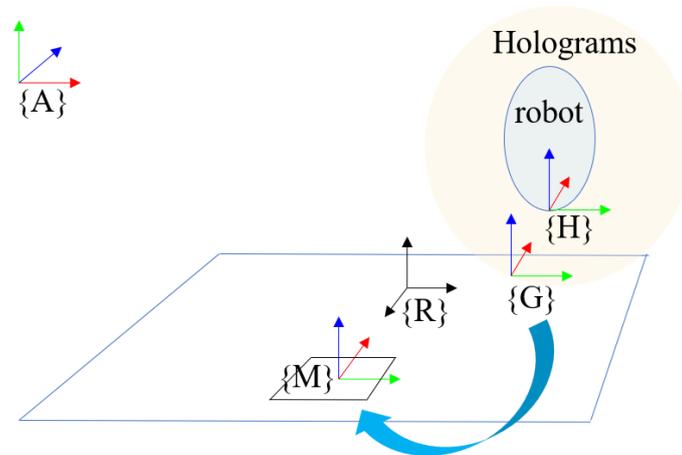
## **IV. Teaching by Augmented Reality**

### **4.1. Mapping Real World to Virtual World**

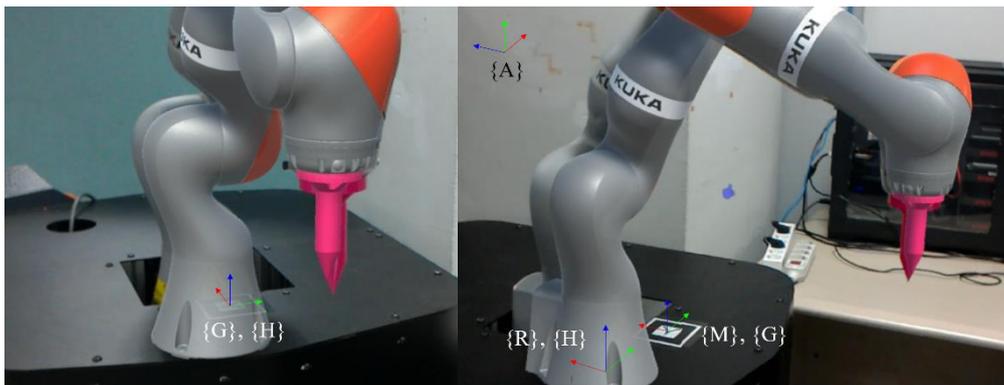
#### **4.1.1. Calibrating the Virtual Robot**

We calibrated the virtual robot to match the corresponding configuration of a physical robot in the real world by using markers (HoloLensARToolKit [28]). We attached a marker to the robot, which was detected by an image processing routine running in the background of the application through the front camera of HoloLens. We performed camera calibration to obtain the intrinsic parameters of the camera, camera matrix, and distortion coefficients, to compute the estimates of a 3D pose of the marker. As Unity3D is used as a development tool for HoloLens, all coordinate systems in a virtual interface of the application follow a left-handed coordinate system of the Unity3D.

Frame A is an application-specific coordinate system and frame M, frame G, and frame H are coordinate systems with respect to frame A (Figure 8). Frame M is a coordinate system of a marker and it is located in the center of the marker. Frame G is a frame of a group of all virtual objects of our system. Frame H is the base frame of the virtual robot and is separate from frame G, as much as the actual physical distance between the marker and the real robot. Frame R is the base frame of the real robot. We performed a mapping from frame G to frame M. By transforming the coordinates from frame G to frame M, we were able to match frame H to frame R (Figure 9).



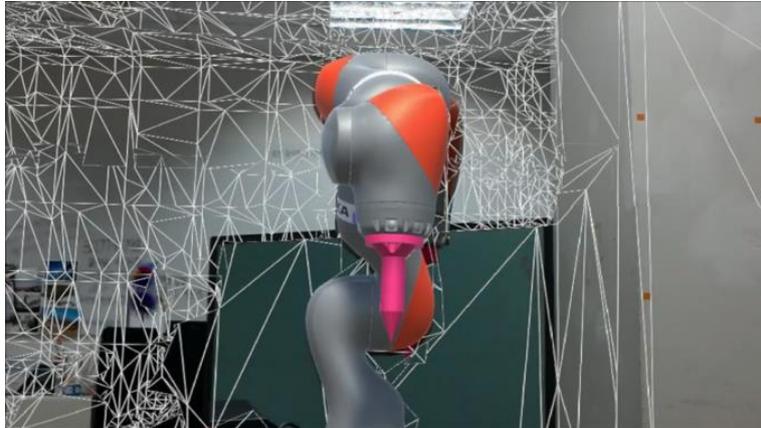
**Figure 8.** Coordinate frames used by the interface



**Figure 9.** Overlaid with a virtual robot

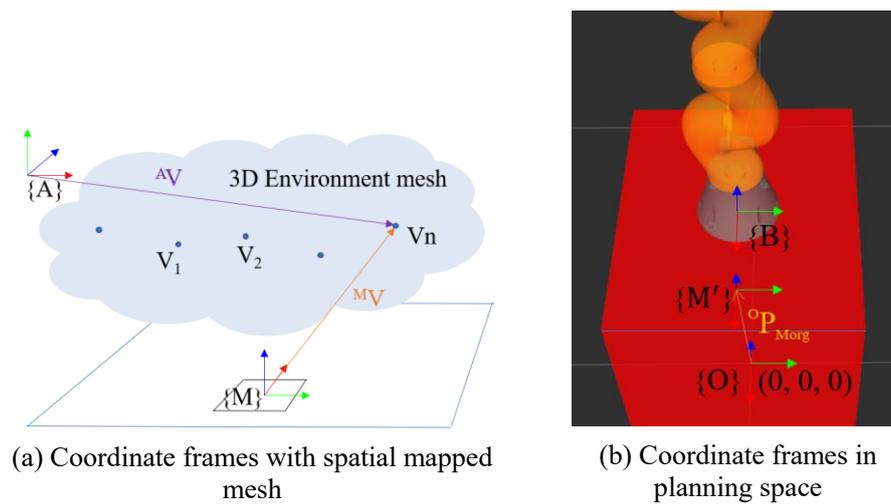
#### 4.1.2. Environment Mapping

We automatically built a map of the environment surrounding the robot and the user through HoloLens to gather the colliding information. HoloLens has a depth camera and four gray-scale environment tracking cameras to sense its environment and capture the users' gestures. With these sensing capabilities of HoloLens and built-in SLAM algorithms, we obtained 3D meshes of the environment (Figure 10).



**Figure 10.** Spatial mapping

When we do spatial mapping, the coordinate system of the 3D mesh data is with respect to frame A. Therefore, we needed a means of linking the coordinate system of this to match the mesh data with the real-world coordinate system. We used the marker pose, the coordinates of which are also with respect to frame A, and mapped the coordinate frame of the mesh data from frame A to frame M (Figure 11-(a)).

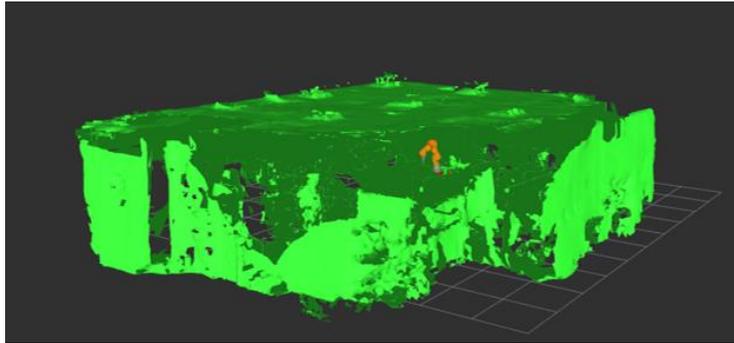


**Figure 11.** Frame mapping

However, when the surrounded environment of the robot is mapped, the actual robot is also mapped and is added to the 3D mesh data (Figure 10). If we do a path planning with this mapping information, the part of the mesh data where the actual robot is mapped will overlap with the robot's planning space and collide with it. Therefore, we removed the mesh corresponding to the real robot.

We sent the coordinate frame changed 3D environment data to the server. We wanted to use this data to acquire the collision information needed for collision free path planning. We used the planning interface of Moveit! running on ROS, which has a right-handed coordinate system unlike Unity3D, so we changed the coordinate system from the left-handed to the right-handed coordinate system.

In the planning space of Moveit!, as shown in Figure 11-(b), there is a different world coordinate system from the world coordinate system in the AR application. Frame O is the world coordinate in the planning scene. Frame B is a base frame of the robot and frame M' is the coordinate frame of the marker in the planning scene. To move the 3D mesh data coordinates to match the coordinate frame of the planning interface, we mapped the coordinate frame of the mesh data from frame M' to frame O. By doing so, we assigned the collision information to the right place. A virtual environment applied to the planning scene of Moveit! is presented in Figure 12. Figure 13 shows a planned path with and without obstacle information. With the surrounding environment information, we could plan the path considering the real-world information in the work cell.

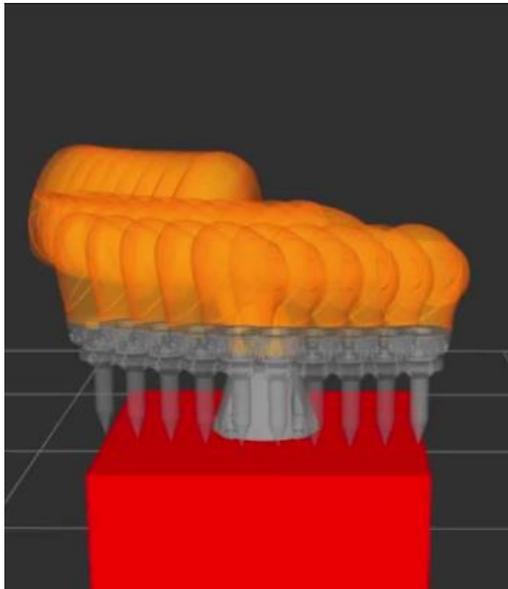


(a) Exterior of the spatial mesh

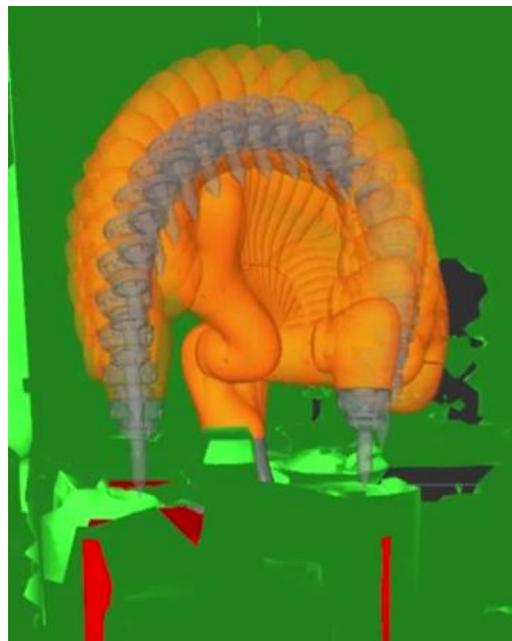


(b) Interior of the spatial mesh

**Figure 12.** Mapped virtual environment



(a) Planned path without an obstacle



(b) Planned path with a box obstacle

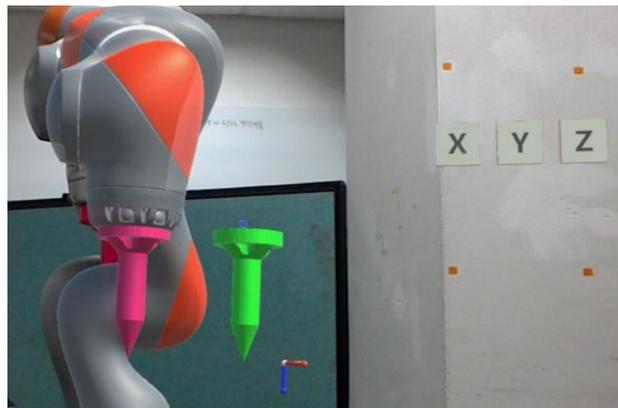
**Figure 13.** Planning in virtual space

## 4.2. Input Interfaces in AR

### 4.2.1. Moving the Tool pose

HoloLens uses its head direction as gaze direction, and when the user gazes at the object they can click or select that object using the air tap gesture. In addition, HoloLens recognizes the air tap with the holding and releasing movement of the hand and these gestures are captured by built-in sensors and cameras in HoloLens, just like spatial mapping. This allows the user to move, rotate, and resize the virtual objects. Therefore, we utilized this capability to move the virtual robot in AR.

For an interactive and intuitive simulation interface, we wanted not only to simulate the planned result with the virtual robot, not the real robot, in AR, but also to consider a way of giving a start pose and goal pose of the robot for planning a trajectory in an intuitive way using AR. To assign the start state and goal state, we designed an interface for the controlling position and orientation of the end-effector of the virtual robot (Figure 14).



**Figure 14.** Pose control interface

As shown in Figure 14, a pose control interface consists of a virtual green tool and three buttons near the green tool. A coordinate frame of the tip (Figure 16) of the green tool is with respect to the base coordinate frame of the virtual robot. In addition, a small virtual coordinate object is also attached near the tip of the tool to provide information about the current coordinates of the tip. With each button, the user can rotate the specific axis of the tip. For example, button 'X' is for x-axis rotation of the tip of the tool. The user can rotate the tip of the tool through the horizontal dragging gesture. Dragging to the right is a positive rotation and otherwise is a negative rotation (Figure 15). In the case of controlling the position of the tool tip, as HoloLens can catch the position of the hand in space, the user can move the tool along by making hold and drag gestures with their hand.

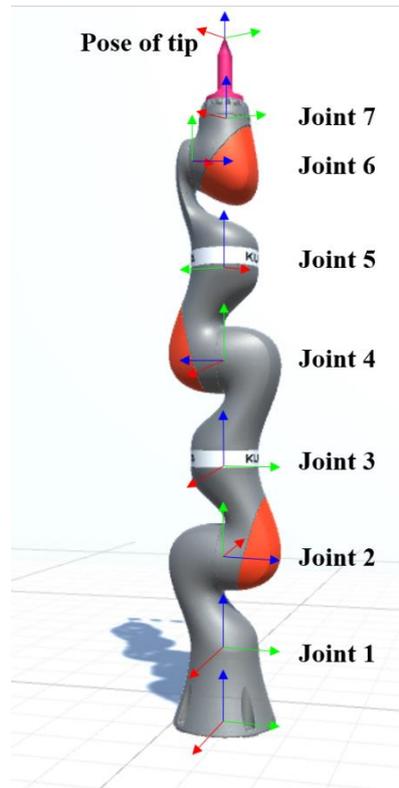


**Figure 15.** Rotation control gesture

#### **4.2.2. Intuitive and Rich Input feedback**

We provided an immediate and rich response every time the user adjusted the pose of the tool tip object to give the desired end-effector's pose. If the user can move only the tool object in the virtual robot, it is not known whether the tool is within the range of the robot's reachability to move or whether the pose of the tool allows the robot to plan the path without collisions. The user has to guess all of these predictions on their own. The only way to confirm the user's guess is to review the results of the simulations and this is a tedious and difficult task for the user. Furthermore, moving the tool separately from the virtual robot is awkward and does not give

the impression of looking at and moving the real robot.



**Figure 16.** Robot 7 joint frames

To deal with the limitations of moving only the tool object, we calculated IK solutions for the pose of the tool tip to figure out how much the robot rotates each of its joints when moving to an assigned pose. In our system, we used seven a degrees of freedom (DoF) robot (Figure 16). Unlike Forward Kinematics (FK):

$$X = f(q),$$
$$q = \{q_i \mid i \in \{1, \dots, n\}\},$$

where  $q$  is the joint space coordinate of the robot and  $n$  is the Degrees of Freedom of the robot and  $X$  is the configuration of the robot's tip, which is the computing pose of end-effector with given joint angles. IK computes the robot's joint angles with the given desired pose of the end-effector:

$$q = f^{-1}(X).$$

HoloLens sends the pose of the tip to the server and the server calculates a collision-free IK solution based on the environment information and then sends calculated seven joint angles back to HoloLens. Then, HoloLens takes joint angles to rotate each joint of the virtual robot so that the user can see the movement of the robot concerning the specific pose of the tip. When the user moves the pose of the tip out of the range of the robot's reachable space, the IK calculation has failed. Furthermore, when the user moves the pose of the tip to the collision space where robot self-collision or collision with the environment occurs, the collision test has also failed. In either case, therefore, HoloLens does not update the movement of the robot and so the user can only see the joint movements that the robot can perform.

This process is repeated while the user adjusts the pose of the tip so that when the user moves the tool, the virtual robot moves continuously as if the real robot were moving. Throughout this process, the data's coordinate system changes according to the coordinate system of the HoloLens and the server while exchanging the data.

### **4.2.3. Setting the Start and Goal States**

The user can save the start and goal poses of the tip of the robot if the user finds the desired states of the robot to create the robot's trajectory by moving its tip. The user can save the start and goal pose of the tip again if they want to specify the new pose. For the convenience of the user, we provided the ability to turn back the pose of the robot to the original pose. After setting the start and goal pose of the tip, the user could send these poses to the server to plan the path.

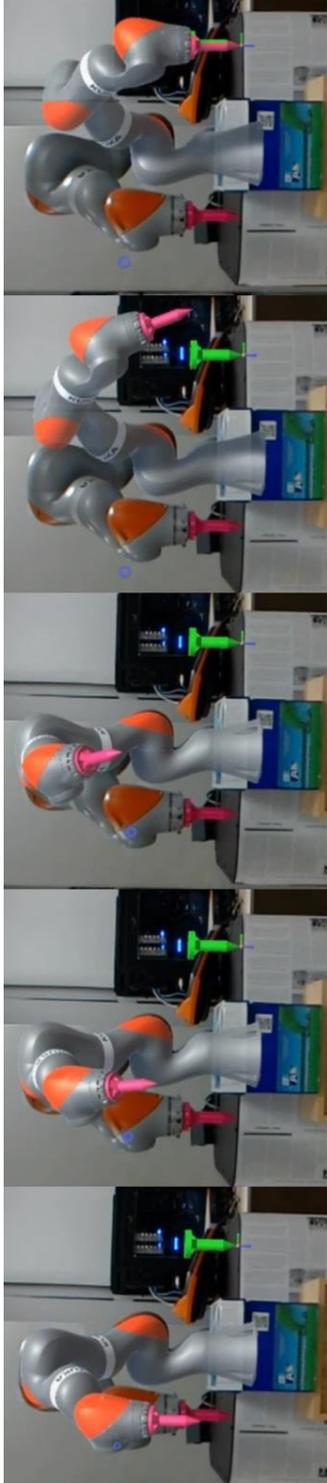
### 4.3. AR simulation

The server plans the path with the received start pose and goal pose of the robot's tip. The resulting trajectory contains several joints states and is sent back to the HoloLens. The user can see the results of the path planning, which is a path between the user-specified robot's start and goal state (Figure 17). As shown in Figure 18, the user can also see the AR simulation overlaid on the actual robot in the work cell. Through AR simulation, the user can intuitively determine whether or not to run the actual robot.

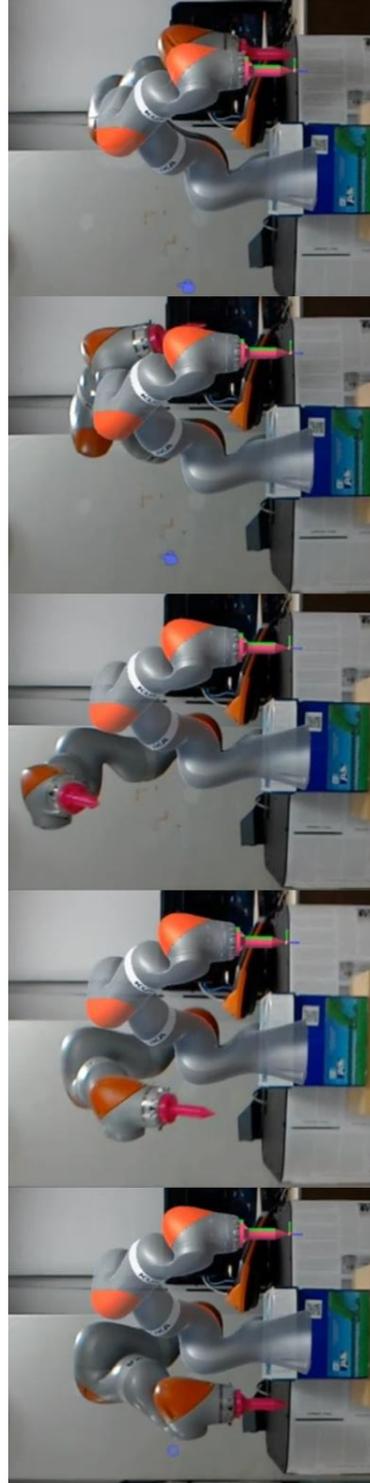
If the user judges that there is no problem, even if they execute the robot and then want to execute the actual robot according to the created path, the user can issue the robot execution command through the interface (Figure 19). If the user wants to create a better path, the user can specify the new start and goal state of the robot and simulate it for checking.



**Figure 17.** AR simulation of the path between the start pose and goal pose of the tool



**Figure 18.** AR simulation



**Figure 19.** Robot execution

## V. Implementation

### 5.1. Implementation Platform

#### 5.1.1. AR Platform

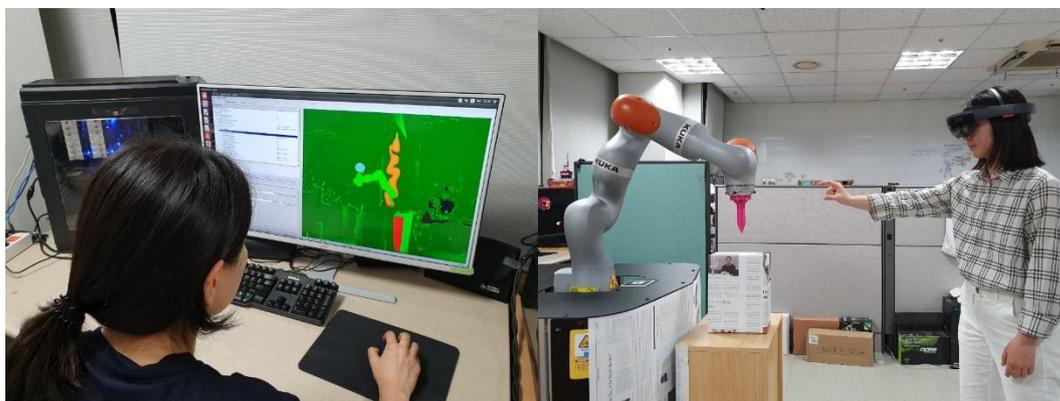
We used Microsoft HoloLens to support the user's interaction with the robot. It is an AR head-mounted display (HMD), which is the first untethered type and see-through holographic computer that supports wireless communication, spatial mapping, and hand gesture recognition. We used the HoloToolkit [27], HoloLensARToolKit [28], the 5.6.1f1 version of Unity3D and Visual Studio 2015, and Windows SDK 10.0.14393.0 to program the application of HoloLens under a Windows 10 Pro 64-bit operating system equipped with an Intel® Core™ i7-4790 CPU @ 3.60GHz, NVIDIA GeForce GTX 750 Ti GPU, and 16GB RAM. The application of HoloLens is written in the C# programming language.

#### 5.1.2. Robot Platform

We used the KUKA LBR IIWA 7 R800 robot. This 7-DoF manipulator has torque sensors integrated into all seven joints. In the controller connected to the robot, the Sunrise OS is used to control this manipulator and using the SunriseWorkbench we could write the robot application in Java programming language. We used the ROS Indigo framework to manage the message server under an Ubuntu 14.04 LTS 64-bit operating system equipped with a 4.0-GHz Intel Core i7 CPU, GeForce GTX 970 GPU, and 24-GB RAM. The ROS is used as a server that assists the user to plan, simulate, and perform the task and the ROS nodes are coded in C++ programming language.

## VI. Experiment

We conducted a  $2 \times 1$  within-participants experiment with repeated measurements to evaluate how our simulated interface helps user interactions with a robot. The independent variable in this study was the type of OLP interface, one is our simulated interface and the other is RViz [29], which is a conventional visualization tool for a robot. In both conditions, the same environment mapping data was used. In the RViz condition, participants interacted with the RViz interface with a 2D monitor and mouse (Figure 20-(a)). In our interface condition, participants wore a Microsoft HoloLens to interact with the robot (Figure 20-(b)). Dependent variables included the objective measure of task performance and subjective measure of the participants' perceptions and preferences.



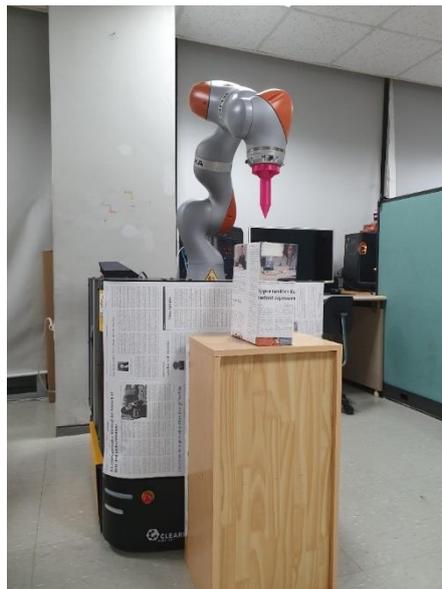
(a) RViz

(b) Our interface

**Figure 20.** Two interface conditions

## 6.1. Task and Environment

Our experimental setup was motivated by contexts where a user nails an object to assemble using a collaborative robot in the manufacturing site. In our study, participants programmed a robot to teach them to nail an object in a work cell. We used a 3D-printed tool (Figure 21), which is attached to the end effector of IIWA to mimic a drill machine. In our experimental environment, there was an object that needed to be assembled on the table (Figure 21). Participants were tasked with teaching a robot to nail the object using the tip of the tool. To assemble the object, participants were instructed to give poses of the tool through the OLP interface to put nails in the middle of the two opposite sides (left and right) of the object, respectively.



**Figure 21.** Experimental environment

## 6.2. Participants and Procedure

We conducted a user study with 12 female participants. The average age of the participants was 27.58 (SD = 3.64), with a range of 23–38. On a seven-point scale, the participants reported a moderate prior familiarity with a robot (M = 3.5, SD = 1.32) and a relatively low familiarity with ARHMD (M = 3.08, SD = 1.80).

The study consisted of performing one task with two interface conditions, RViz and our simulated interface. The study took approximately 40 minutes and consisted of four phases: (1) performing a task with one condition and a survey, (2) break, (3) performing a task with another condition and a survey, and (4) additional comments. We counterbalanced the order of interface conditions in performing the task and gave participants a 5 minutes break after performing the task with one interface. After the participants completed performing the task with two interfaces and the survey on each interface, they could give additional comprehensive comments.

When performing a task with RViz, it has the following steps: (1) introduction, (2) task, and (3) survey. Performing a task with our simulated interface consisted of 5 steps: (1) introduction, (2) AR gesture training, (3) calibration, (4) task, and (3) survey. In the introduction phase, the participant was led into the work cell where they read instruction sheets detailing the task and description of the corresponding interface. Only when using our interface, we trained the participant in gestures for HoloLens using ‘Learn Gestures’ which is a built-in gesture tutorial application of HoloLens. Then our ARHMD application for simulation was started and calibration was performed. In both interface conditions, the participants performed the task through the given interface and the time taken to complete was measured. The participant was then given a post-survey about their experience.

### 6.3. Measures and Analysis

For quantitative analysis, we used an objective measurement and three subjective measurements to evaluate the RViz and our proposed interface (Table 1). Objective task efficiency was measured by the total time the participant spent in teaching a robot through the given OLP interface. Lower task completion times indicate better performance. We used 7-point Likert scale questionnaire items to measure subjective participant perceptions and preferences. The scale rated how easy it was to learn the given interface, how easy it was to use the given interface, and how easy it was to understand the spatial cues through the given interface.

In addition, we obtained qualitative feedback from the participants as part of the concluding questionnaire to describe their experiences of the given interface. We asked the participants to write down a detailed explanation of why they gave the rating they did concerning their subjective experiences and we also asked them to express any additional suggestions or personal feelings.

**Table 1.** Assessment item

Measurement
Task completion time
Learnability
Easy to give an input
Situation awareness

## 6.4. Results

We have summarized our objective and subjective results in Figure 22. We used the average value of the data from each assessment item, which was obtained when the participant performed experiments with RViz and our interface for analysis (Table 2). We analyzed our task performance metric to confirm that our simulated interface was easy to learn and helped participants to quickly teach a robot. Comparing the participant's task completion time using our interface to the RViz, we found that the total time spent on completing a task significantly decreased by two thirds when using our simulated interface.

In subjective results, participants rated their experience of learning the given interface to teach a robot. We found that our simulation interface was rated significantly higher than the RViz more than twice. Participants commonly pointed out that the pose control interface of RViz, which consists of seven graphic markers to controls the translation and rotation of a robot, is difficult to control. Because all seven of its markers are attached to the end-effector of a robot, participants said it took time to distinguish them without confusion. Furthermore, in RViz, the user not only adjusts the pose with the mouse drag, but they also have to give the change of viewpoint using the mouse buttons.

Participant p03: *"I had difficulty distinguishing the translation/rotation mark from the beginning. Once I learnt it, it becomes simple, but there were many tasks to be learned before that."*

Participant p06: *"Those who are familiar with existing 3D tools find it easy to learn, but those who are not will find it difficult to learn because of the difficulty of delicate manipulation and memorization of rotation, translation, and scale."*

However, regarding our interface, many participants said that our pose control UI is simple and intuitive because of its simple design and hand gestures. They also said it was convenient because there was no need to manipulate the viewpoint.

Participant p04: *“It was certainly intuitive by holding and moving the control interface by hand rather than mouse.”*

Participant p06: *“It was very intuitive in the case of translation.”*

Participant p02: *“Interface design for pose rotation was clean and easy to recognize.”*

Participant p07: *“Interaction with the system through the gesture input method and panel button input makes it easy to learn.”*

Furthermore, we analyzed the participants’ responses to the interface in terms of how easy or difficult they found it to use. We found that our interface significantly improved the usability for participants providing input to the interface over the RViz almost by more than three times. Participants responded that interacting with the RViz using a mouse is difficult and inconvenient. Because the three-dimensional information within the interface is not handled by drag alone, which is the two-dimensional input of the mouse, the additional moving viewpoint of a mouse is needed, and such tasks are very cumbersome. In addition, some participants said that there was some discomfort involved in using the graphic interfaces of RViz.

Participant p02: *“When I gave the input, I felt uncomfortable because I had to change the viewpoint with precision.”*

Participant p03: *“When I saw the planned results and specify the start and goal pose, it was necessary to turn the viewpoint continuously to check through the mouse. And in some viewpoints, the movement for controlling a pose was inconvenient with the mouse alone. It was difficult to get out if I got stuck at a viewpoint. It was also difficult to repeat rotate, translate, and rotate again to get out of a robot’s immobile pose.”*

However, the participants’ comments on our interface were positive. As mentioned in the previous assessment of learnability, they said that the design of our pose control interface was easy and convenient for inputting and reduced the number of controls.

Participant p08: *“Translation and rotation input methods were clearly distinguished, so there were fewer errors when inputting them.”*

Participant p02: *“The UI of the pose control interface was easy to operate without complexity.”*

*It was intuitive and easy to understand as it was possible to input by directly moving a virtual tool. And with simple gestures, I could give the translation and rotation of the pose.”*

Participant p03: *“It was convenient to move the tool anywhere I want, contrast to RViz, and I could see exactly where it is located in the space.”*

Finally, we compared our simulated interface to the RViz along a situation awareness metric of how the given interfaces affected participants’ understanding of spatial communication with a robot. We found that our interface was ranked as almost three times more helpful than RViz. Many participants answered that it was difficult to verify three-dimensional spatial information in RViz. The reasons for this were the difficulty of manipulating the viewpoint and the hassle of having to shift the viewpoint several times to give accurate input. In addition, some participants said that there was a lack of understanding about the environment due to the lack of texture mapping.

Participant p02: *“The spatial understanding of the virtual view was not as intuitive as the view of HoloLens.”*

Participant p03: *“It looks like the additional texture is needed. Because there was no knowing what the surrounding mapped environment looked like.”*

Participant p06: *“It was not easy to control where obstacles need to be nailed by moving the camera view.”*

Participant p07: *“Because it was difficult to check the 3D spatial information, it was difficult to confirm whether the input was properly made.”*

In contrast, however, participants said that our interface provided an intuitive cue for spatial communication, which helped them a lot in performing the task.

Participant p02: *“It was very intuitive and easy to understand by providing simulation views with real robots.”*

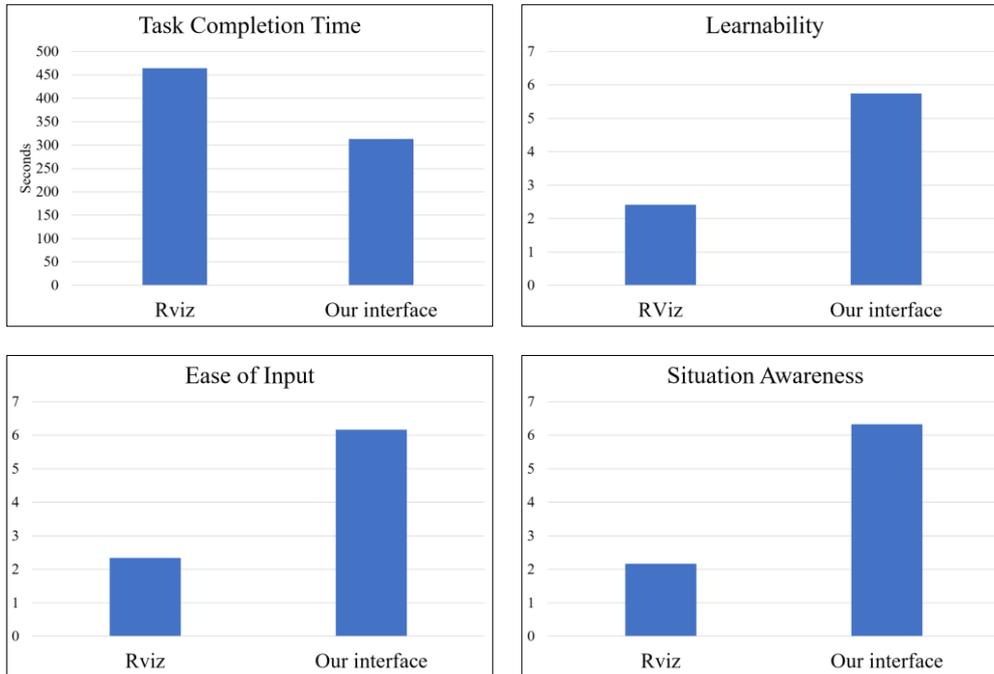
Participant p03: *“Compared to the RViz, repetitive tasks were reduced, and the reason seems to be that the sense of realism and immersion felt in the space where the robot is certainly located. As a result, the rotate to translate and to rotate iterative correction were reduced without pulling*

*to an excessive position that the robot cannot reach.”*

Participant p05: *“It was able to move intuitively because the virtual robot had to be manipulated and the actual robot seemed to match.”*

Participant p06: *“It was good to see the virtual robot's pose in the real world in any direction of the field where the real robot is located.”*

Therefore, in this study, we have shown that our simulation interface is much easier to learn and use and improves situation awareness more than RViz. We have also shown that all of these have affected the users' task completion time.



**Figure 22.** Comparison of RViz and our interface based on user data

**Table 2.** User study data

The data is collected from a group of 12 students

(a): Task completion time (b): Learnability (c): Ease of input (d): Situation awareness

Participant	RViz				Our interface			
	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)
p01	1601	1	1	1	602	5	6	4
p02	335	3	4	3	279	7	6	6
p03	285	2	3	3	311	6	7	7
p04	600	1	2	2	490	6	5	7
p05	339	2	4	2	280	6	7	7
p06	294	4	5	5	442	6	4	6
p07	338	3	2	1	177	7	7	7
p08	483	2	1	1	181	5	6	7
p09	381	4	1	1	199	5	7	7
p10	327	3	1	2	298	5	6	7
p11	302	2	2	2	235	5	7	6
p12	289	2	2	3	262	6	6	5
<b>Average</b>	464.5	2.4166667	2.3333333	2.1666667	313	5.75	6.1666667	6.3333333

## VII. Conclusion

In this dissertation, we have presented an intuitive robot simulation interface to improve the HRI of non-expert users and support safe and effective robot programming in an unstructured environment using untethered see-through AR. We have proposed a new concept of teaching robots: teaching by AR, which is teaching the robot using AR with the user's demonstration through a virtual robot without contact with the robot. We used robot programming through the AR approach and provided intuitive visual feedback informing the validity of the user input in real-time in AR. Through ARHMD, we supported the user to see a view that matches the virtual simulation interface on the user's physical workspace, and we provided mapping of the work cell environment for collision-free path planning. We also conducted a user study in which participants were tasked with teaching the robot a program through our simulation interface and RViz. The results of the study demonstrate that our interface is intuitive, time efficient, and helps the user to teach the robot. As for future work, we would like to increase the accuracy of the calibration by adding more markers to increase the accuracy of the input position in the AR. We would also like to recognize both hands so that a user can more easily and intuitively give the input with both hands rather than the basic gestures possible with HoloLens, which currently recognizes one hand only. Our system can be used for robot education. In this system, a virtual robot acts, moves, and looks like a real robot existing in the real world. Our prototype system shows the potential for new techniques for teaching robots using AR and will be a guide to future studies on robot simulation using AR.

## Bibliography

- [1] P. Milgram, S. Zhai, D. Drascic and J. Grodski, "Applications of augmented reality for human-robot communication," *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, pp. 1467-1472 vol.3, 1993.
- [2] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators & Virtual Environments*, 6.4: 355-385, 1997.
- [3] U. Neumann and A. Majoros, "Cognitive, performance, and systems issues for augmented reality applications in manufacturing and maintenance," *Proceedings. IEEE 1998 Virtual Reality Annual International Symposium (Cat. No. 98CB36180)*, pp. 4-11, 1998.
- [4] S. A. Green, M. Billingham, X. Chen, and J. G. Chase, "Human-robot collaboration: A literature review and augmented reality approach in design", *International journal of advanced robotic systems*, 5.1: 1, 2008.
- [5] J. A. Frank, M. Moorhead, and V. Kapila, "Realizing mixed-reality environments with tablets for intuitive human-robot collaboration for object manipulation tasks," *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 302-307, 2016.
- [6] A. Gaschler, M. Springer, M. Rickert, and A. Knoll, "Intuitive robot tasks with augmented reality and virtual obstacles," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6026-6031, 2014.
- [7] M. Stilman, P. Michel, J. Chestnutt, K. Nishiwaki, S. Kagami, and J. Kuffner, "Augmented reality for robot development and experimentation," *Robotics*

*Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-55, 2.3, 2005.*

- [8] R. D. Schraft and C. Meyer, "The need for an intuitive teaching method for small and medium enterprises," *VDI BERICHTE*, 1956: 95, 2006.
- [9] Homepage of the Occupational Health and Safety Administration, U.S. Department of Labor. [Online]  
[http://www.osha.gov/dts/osta/otm/otm\\_iv/otm\\_iv\\_4.html](http://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html).
- [10] KUKA Robotics. 2018. Products, Robot controllers, SmartPAD [Online]  
<https://www.kuka.com/en-de/products/robot-systems/robot-controllers/smartpad>.
- [11] G. Reinhart, U. Munzert, and W. Vogl, "A programming system for robot-based remote-laser-welding with conventional optics", *CIRP annals*, 57.1: 37-40, 2008.
- [12] S. Wrede, C. Emmerich, R. Grünberg, A. Nordmann, A. Swadzba, and J. Steil, "A user study on kinesthetic teaching of redundant robots in task and configuration space," *Journal of Human-Robot Interaction*, 2.1: 56-81, 2013.
- [13] C. Schou, J. S. Damgaard, S. Bøgh, and O. Madsen, "Human-robot interface for instructing industrial tasks using kinesthetic teaching," *IEEE ISR 2013*, pp. 1-6, 2013.
- [14] K. Kronander and A. Billard, "Learning compliant manipulation through kinesthetic and tactile human-robot interaction," *IEEE transactions on haptics*, 7.3: 367-380, 2013.
- [15] B. Akgun, M. Cakmak, J.W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective", *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, p. 391-398, 2012.

- [16] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, "Upper-body kinesthetic teaching of a free-standing humanoid robot", *2011 IEEE International Conference on Robotics and Automation*, pp. 3970-3975, 2011.
- [17] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, 31.2-3, 115-131, 2011.
- [18] L. Žlajpah, "Simulation in robotics." *Mathematics and Computers in Simulation*, 79.4: 879-897, 2008.
- [19] Z. Pan, J. Polden, N. Larkin, S. V. Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pp. 1-8. VDE, 2010.
- [20] J. J. Craig, "Introduction to Robotics: Mechanics and Control, 3<sup>rd</sup> Edition"
- [21] P. Neto and N. Mendes, "Direct off-line robot programming via a common CAD package", *Robotics and Autonomous Systems*, 61.8: 896-910, 2013.
- [22] R. Azuma, Y. Baillet, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality", *IEEE computer graphics and applications*, 21.6: 34-47, 2001.
- [23] D. Drascic, J. J. Grodski, P. Milgram, K. Ruffo, P. Wong, and S. Zhai, "ARGOS: A display system for augmenting reality," *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems*, p. 521, 1993.
- [24] G. Michalos, P. Karagiannis, S. Makris, Ö. Tokçalar, and G. Chryssolouris, "Augmented reality (AR) applications for supporting human-robot interactive cooperation," *Procedia CIRP*, 41: 370-375, 2016.

- [25] J. W. S. Chong, S. Ong, A. Y. Nee, and K. B. Youcef-Youmi, "Robot programming using augmented reality: An interactive method for planning collision-free paths," *Robotics and Computer-Integrated Manufacturing*, 25.3: 689-701, 2009.
- [26] H. C. Fang, S. K. Ong, and A. Y. C. Nee, "Interactive robot trajectory planning and simulation using augmented reality," *Robotics and Computer-Integrated Manufacturing*, 28.2: 227-237, 2012.
- [27] HoloToolkit [Online] <https://github.com/microsoft/MixedRealityToolkit-Unity/releases/tag/v1.5.6>
- [28] HoloLensARToolKit [Online] <https://github.com/qian256/HoloLensARToolKit/releases/tag/v0.1>
- [29] RViz [Online] <https://github.com/ros-visualization/rviz>

## 국문초록

시스루 증강 현실을 이용한 비 구조화된 환경을 위한 직관적인 로봇

계적 시뮬레이션 인터페이스

김현정

컴퓨터공학과

이화여자대학교 대학원

최근 로봇 기술의 발전으로 등장한 협업 로봇은 크기가 작고 로봇 전문가 없이 로봇 프로그램을 자주 그리고 쉽게 프로그램 할 수 있다는 점에서 중소 기업에서 유용하다. 그러나 로봇이 사용자에게 충돌할 위험이 있으며 이를 극복하기 위해서는 작업자가 안전하고 쉽고 직관적으로 협업 로봇으로 작업을 할 수 있는 인터페이스 시스템이 필요하다. 본 논문에서는 시스루 증강 현실 HMD 디바이스를 이용하여 비 구조화된 환경에서 전문가가 아닌 사용자의 인간-로봇 상호 작용을 개선하며 안전하고 효과적인 로봇 프로그래밍을 지원하는 직관적인 로봇 시뮬레이션 인터페이스를 제시한다. 이를 위해 본 논문에서는 로봇과 실제로 접촉하지 않고 증강현실에서 가상의 로봇으로 사용자가 로봇에게 시연하여 가르치는 증강 현실에 의한 로봇 교습이라는 새로운 개념을 제안한다. 제안한 시스템에서는 증강 현실을 활용한 로봇 프로그래밍을 사용하고 증강 현실에서 실시간으로 사용자 입력의 유효성을 알리는 직관적인 시각적 피드백을 제공한다. 증강 현실 HMD를 통해 사용자는 가상 시뮬레이션 인터페이스를 사용자의 실제 작업 공간에서 겹쳐 볼 수 있으며, 작업 공간의 환경 매핑을 제공받아 충돌이 없는 경로 계획을 할 수 있다. 실험을 통해 본 논문에서 제안한 로봇 프로그래밍을 위한 시뮬레이션 인터페이스가 기존의 시뮬레이션 인터페이스보다 직관적이며 사용하기 쉽고 효율적인

것을 보였다.