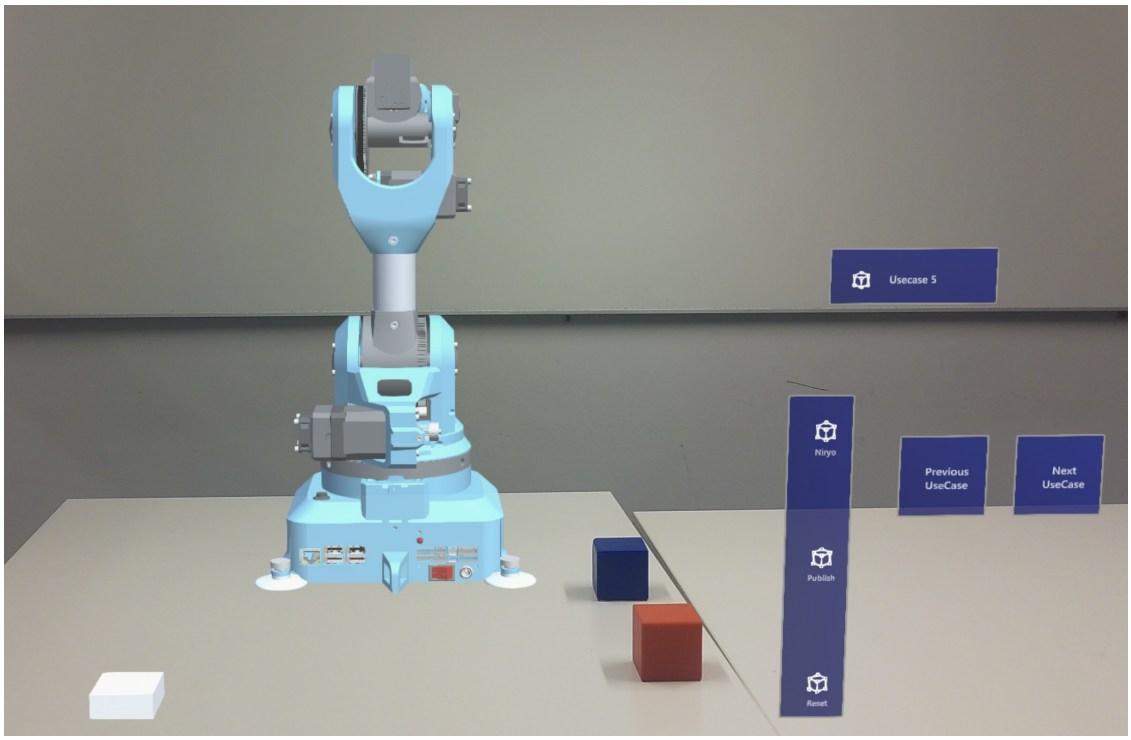

Robot Plan Visualization using Hololens2

Master's thesis in System, Control and Mechatronics

Darshan Gad

Systems and controls

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Robot Plan Visualization using Hololens2

Darshan Gad

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Systems and Control Division
Chalmers University of Technology
Gothenburg, Sweden 2024

Robot Plan Visualization using Hololens2
Darshan Gad

Supervisor: Maximilian Diehl, Doctoral Student, Chalmers
Examiner: Karinne Ramirez-Amaro, Associate Professor, Chalmers

Master's Thesis 2024
Department of Electrical Engineering
Systems and Control Division

Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: AR scene show the AR robot placed on a table along with two real cubes.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Robot Plan Visualization using Hololens2
Darshan Gad
Department of Signals and Systems
Chalmers University of Technology

Abstract

In recent years, the use of robots as the designated solution for simplifying human tasks has seen a notable rise. The incorporation of robots is diverse, and has catered to purely industrial applications like production lines as well as to collaborative environments where humans and robots work together. Research has shown that to increase human reliance on robots, human Trust in robots is essential. However, the integration of robots into human workflows poses a challenge, as it may lead to potential failures arising from software malfunctions, shifts in operating environments, or the misinterpretation of human intentions. Therefore, it is desirable to have a means of conveying both the failures and intentions of robots. One possibility for communication involves the use of Augmented Reality. Recent studies in the augmented reality domain reveal that providing humans with information about the robot's intentions, significantly aids in understanding the robot's behavior and plays a crucial role in building trust.
The objective of this thesis is to investigate the advantages of visualizing robot actions for users and to conduct an assessment of this visualization approach using augmented reality. In the initial phase, we developed a system to enable the visualization of robot's actions in augmented reality, providing users with the capability to replay each action in a robot task. Following this, we conducted a user study to evaluate whether visualizing robot intentions contributes to an enhanced understanding of robot behavior and improves users' ability to identify robot failures caused by errors in plans, incorrect execution of plans by the robot or due to changes in the robot environment .
The results of the user study indicate that users were significantly more successful in understanding the robot's intentions. When failures occurred, users could identify them 97% of the times by visualizing the robot's intentions. Moreover, we found that the ability to replay the robot's actions further enhanced users' ability in identifying failures in robot actions.

Keywords:
Explainable AI Planning
Augmented Reality
Robot failure visualization

Acknowledgements

I would like to extend my sincere gratitude to my supervisor Maximilian Diehl for his invaluable mentorship during the entire course of this thesis project throughout this journey. He has guided me in all aspects starting from problem description and all the way to user study and report writing. His deep knowledge in the field of Augmented reality and Planning has helped me ease through the thesis when at times I was faced by difficult technical challenges.

I am deeply indebted to the continuous feedback and review of my work by my examiner Karinne Ramirez-Amaro and for her diverse perspectives in enriching the quality of this thesis work.

To my friends, and my colleagues for your encouragement and motivations and for your belief in my abilities that kept me going during times that very challenging.

Special thanks to the participants of the user study, which is the most valuable result of this thesis work.

Last but not the least, I am grateful to my family for their unconditional love and support.

Darshan Gad, Gothenburg, November 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AI	Artificial Intelligence
AP	Automated Planning
AR	Augmented Reality
HMD	Head Mounted Device
HRC	Human Robot Collaboration
HRI	Human Robot Interaction
MRTK	Mixed Reality Toolkit
PDDL	Planning Domain Definition Language
ROS	Robot Operating System
UI	User Interface
URDF	Unified Robot Description Format

Contents

List of Figures

1
Introduction

In recent years, the use of robots as the designated solution for simplifying human tasks has seen a notable rise. The incorporation of robots is diverse, and has catered to purely industrial applications like production lines as well as to collaborative environments where humans and robots work together [6]. The integration of robots into human workflows poses a challenge, as it may lead to potential errors arising from software malfunctions, shifts in operating environments, or the misinterpretation of human intentions. [7].

Research has shown that to increase human reliance on robots, human **Trust** in robots is essential [8]. Building this trust requires assurance that tasks assigned to robots will be executed without errors and without posing any risks, whether under supervision or in unsupervised scenarios. The provision of tools that enable humans to detect potential failures in robot performance before execution is crucial. This proactive approach not only aids in understanding robot behaviors but also plays a pivotal role in establishing and reinforcing trust [9].

In this thesis we particularly focus on supporting users understanding the high-level decision making process of the robot. These high-level steps are also referred as **Automated Planning** (AP) [10]. Planning is the process of deducing in advance, the steps needed to achieve a goal [11]. Automated Planning, deals with abstracting several basic or low-level actions, such as picking up an object, to complex or high-level tasks, such as setting the table. The abstraction of low level actions to high level tasks is carried out by a tool called *Planner* [12]. The high level tasks generated by the planners are also referred to as plans. The plans define an optimal sequence of actions to achieve a goal. Before plans can be generated, each action needs to be defined using a set of preconditions, and a set of effects [13]. AP involves the study of methods of creating, analysing, managing, and executing plans [14]. But, the plans generated from AI planning and the action specification are complex to understand and often require domain knowledge in understanding them [13]. According to research, explaining these plans to the users is an effective way to increase human trust in robots [15]. Since most people are not robot experts, it has been suggested that visualization of plans is, an effective way for the users to understand robotic behaviors before evaluating them in the real world [16]. This thesis will investigate visualization of plans as a tool to facilitate the detection of plan execution failures before execution on the physical robots.

Study has shown that **Augmented Reality (AR)** is an effective tool to visualize

robot plans [17]. AR is a system that enhances the real world by superimposing computer-generated information on top of it. In the context of the thesis AR superimposes the robot holograms in the real world which makes the visualizations of robot execution of the plans, more realistic and easier to interpret [18]. There are other traditional visualization tools such as,
Gazebo : Provides a virtual environment equipped with a 3D physics simulator and brings realistic simulations for testing and visualization.
RViz : Here we can visualize the movements of the robot in 3D space. It is mainly used to see the robots' current state and sensor information.

These traditional solutions require modeling of the environment around the robot and lacks the real-world sense offered by AR.

The following are the contributions of this thesis,

- We developed a system that will enable visualization of high-level robot plans in AR aiming to improve the detection of intentions and failures by the users,
- An user interface is developed that allows the users to switch between robots and visualize the execution of tasks by the selected robot,
- The user is able to replay the robot action using the user interface which helps in identifying the robot failures,
- We conducted a user study to analyze the ability of the user to detect robot failures given the robot plans are visualized.

1.1 Research questions

In this thesis we address the following research questions.

1. Does AR visualization of the high-level plan allow users to detect potential failures before the execution on the real robot?
2. How effective is the action visualization in AR, in detecting action mis-specification?

3. Does providing tools to the user to playback the actions aid in robot failure identification?

2
Related Work

The applications involving human interaction and collaboration with robots, where specific tasks are performed jointly, are commonly referred to as Human-Robot Interaction (HRI) and Human-Robot Collaboration (HRC). The success of human-robot collaboration or interaction relies significantly on the establishment of trust among the users toward the robot's operations. Here, we study researches performed within the areas of HRI/HRC, focusing specifically on the recognition of **Intent** and **failure**, both aspects being crucial for building trust. These studies explore techniques and advantages related to recognizing the intentions of robots, where the plans and actions of the robot play a pivotal role.

In a particular study [1], participants were equipped with augmented reality (AR) devices designed to convey the intended flying direction of a drone. This research focuses on assessing how the visualization of robot intentions influences the efficiency of collaboration between humans and robots. The AR hardware utilized in this study includes Hololens and Meta2. The user interface includes spatial mini-maps offering information about the position or planned route of the robots in relation to the user, indicators reflecting robot status (such as battery level, task progress, task queue, etc.), and live video streams from the camera(s) mounted on the robot. Figure 2.1 illustrates various potential AR visualizations depicting the intent of the robot.

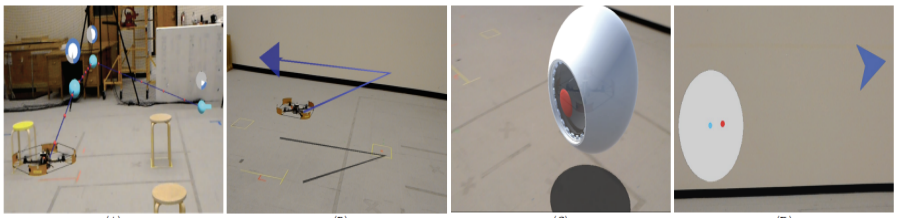

Figure 2.1: Ways of conveying info using AR in HRI by visualizing robot motion intent. Left to Right 4 reference prototypes for cuing aerial robot flight motion: (A) NavPoints, (B) Arrows, (C) Gaze, (D) Utilities [1].

Another research study proposes a system designed to assess the intentions of humans, thereby reducing the workload in a co-located environment [2]. Utilizing the Microsoft HoloLens as the AR hardware, this system is tailored for facilitating collaboration between humans and industrial robots through a head-mounted device (HMD). The system presents the intended goal of the robots, specifically

the end-effector position, through a hologram that incorporates a 3D sound source and a virtual display of intent, as depicted in Figure 2.2. Notably, the system is robot-agnostic and entirely portable, requiring minimal setup at the initiation of interaction. It ensures that a single human worker can seamlessly interface with multiple robots consecutively, eliminating the need for specialized robot cells or sensors around each robot. .

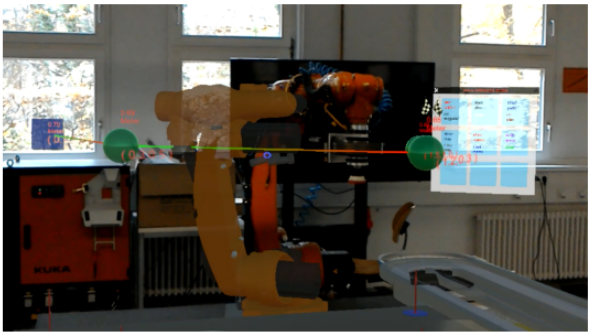

Figure 2.2: Visualisation of the intentions of the robot to the human coworker - virtual execution with a holographic robot and plan visualisation [2].

A study conducted at Örebro University in Sweden [19] presents various solutions aimed at enhancing safety for humans working in areas shared with robots. The robot setup for this study is illustrated in Figure 2.3. Autonomous guided vehicles (AGVs) are increasingly utilized in industries, with one notable application being forklifting. In shared workspaces, these forklifts operate continuously. The project's objective is to project the robot's intent onto the shared workspace using arrows or similar signs easily observable by humans. To achieve this goal, the robots are equipped with cameras to track the eye movements of nearby humans. This eye-tracking data is utilized to understand the direction of human movement, enabling the robots to avoid collisions. The study employs augmented reality (AR) modeling of the robot, tested in a real environment before conducting trials with actual robots.

In the research study conducted in Japan [20], the focus is on a mixed reality interface system designed to allow users to visualize the current state of robot perception in a retail environment. This addresses the challenge where users lack awareness of the robot's perception, leading to a deficit in trust during the adoption of robots in home or retail settings. The study concludes that employing a mixed reality system facilitates the understanding of human-robot interaction by non-expert users. The mixed reality interface employs a Head-Mounted Display (HMD) worn by users, providing a visual representation of the robot's internal perception of the environment. Figure 2.4 illustrates the visualization of perception presented to users. In a retail context, the study utilizes robots to guide customers with product descriptions and as a store map. Employees in the retail setting, equipped with HMDs, can monitor the robot's status in real-time as it performs its tasks.
In a related project [21], the impact of Industry 4.0 on increasing robot usage in various industries is discussed, emphasizing the importance of enhancing the safety

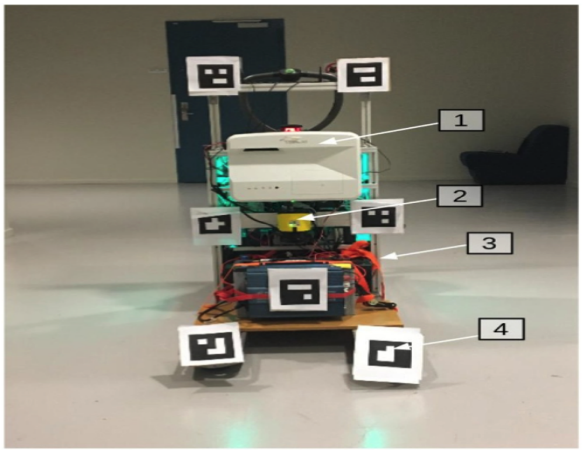

Figure 2.3: A standard short throw projector (1) is mounted on a retrofitted Linde CitiTruck AGV (3). The projector is used to project the intention of the vehicle on the ground plane in front of the truck. Two scanners are mounted in front (2) and back to ensure safety for participants and for tracking the participant motion during the experiment. Label (4) represents the markers placed on the robot for eye-tracking data analysis.

of humans working alongside robots. This project introduces an Augmented Reality (AR) system designed to display robot faults, with Microsoft HoloLens serving as the AR hardware. Virtual symbols are employed to simulate faults in robots, including scenarios such as collisions, software errors, braking system malfunctions, or sensor errors. These symbols are strategically positioned in the AR domain, providing a visual representation close to the location of the identified fault. The results from this approach indicate that users can recognize faults more accurately and with fewer movements. Figure 2.5 illustrates an example of such a symbol, in this case, an arrow, utilized to guide technicians to the point of failure. This adaptive AR system contributes to improving the efficiency of fault recognition in human-robot collaborative environments.
While the presented research examples focus on visualizing the intent of robots in motion, this thesis focuses on visualization of step-wise execution of plans. This thesis will also study the effectiveness of robot plan visualization as a tool to identify the failures of the robot tasks before their execution in real world.

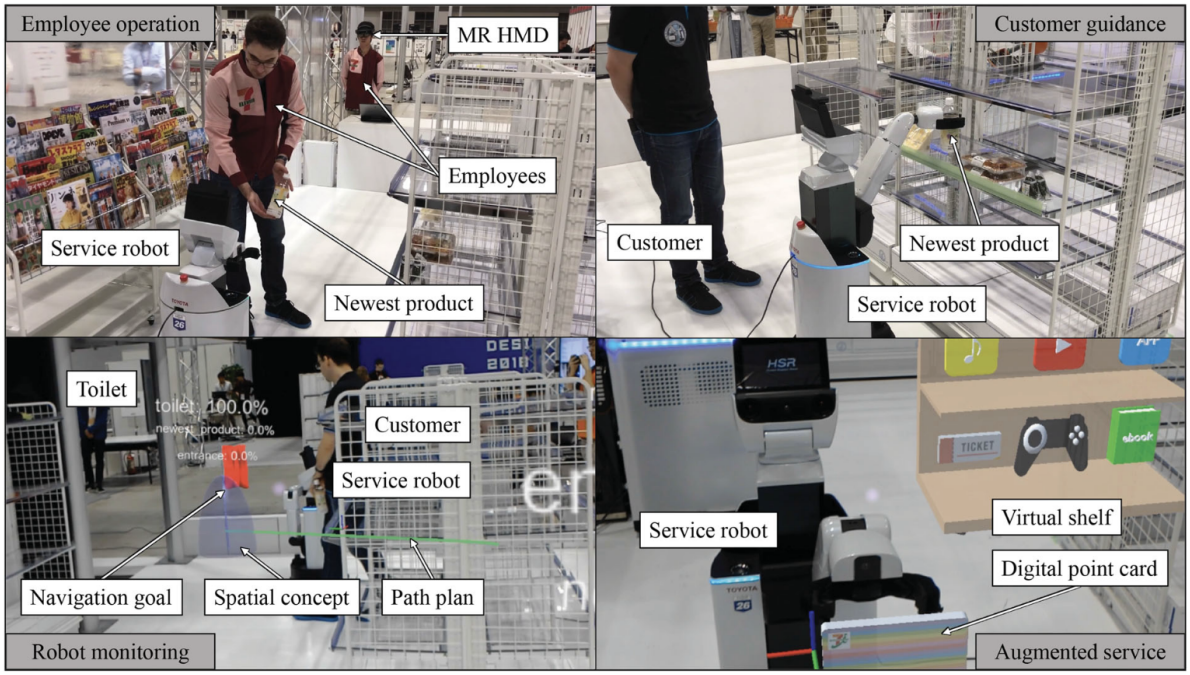

Figure 2.4: Illustration of the service robotics solution. Top left: An employee teaches the robot about a new product using natural language. Top right: The robot serves a customer who asked for the newest product in store. Bottom left: An employee monitors the robot status in MR while it guides a customer to the toilet. Bottom right: A customer interacts with the robot through an MR-enabled device to download a new digital point card.

Figure 2.5: A virtual red arrow marked in the blue circle is used to capture technician's attention by pointing to the direction of the fault

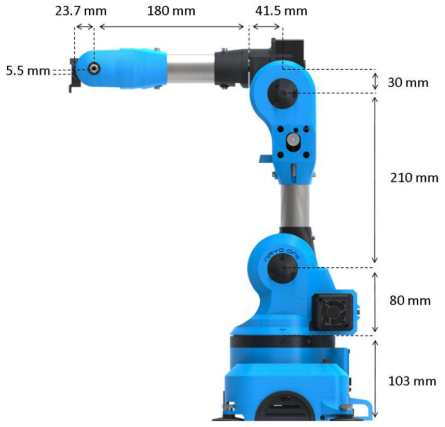

Figure 3.1: Niryo one [3]

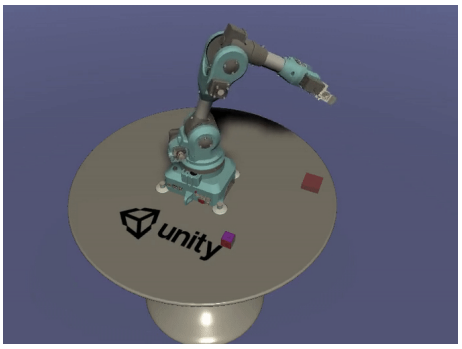

Figure 3.2: Niryo one model

3
Background

3.1 Hardware and software tools

In this thesis we use two robot platforms, the Niryo one and the UR3. Following is the specifications of the robots.

3.1.1 Niryo Robot

The Niryo one robot [22] is a 6 axis desktop robot by Niryo. The robot serves educational purposes and for testing in small assembly lines. Figure 3.1 and 3.2 shows the images of the real and model of Niryo one robot. The manufacturer of the Niryo one robot also provides the Unified Robot Description Format (URDF) model of the robot containing XML specifications for robot models, sensors, etc, which can be imported into 3D tools like Unity.

- The robot weighs about 3Kgs and carry payload upto 0.3Kgs.
- Its robot arms can reach a circle of radius 44cm.
- The robot can connect to 2.4Ghz WiFi in the range 802.11n
- Power consumption is about 60W at 12V.
- The gripper weights 70g and is 8cm long.
- Max opening width of the gripper is 27mm.

In this user study the Niryo one robot model is used in simulations and in user study.

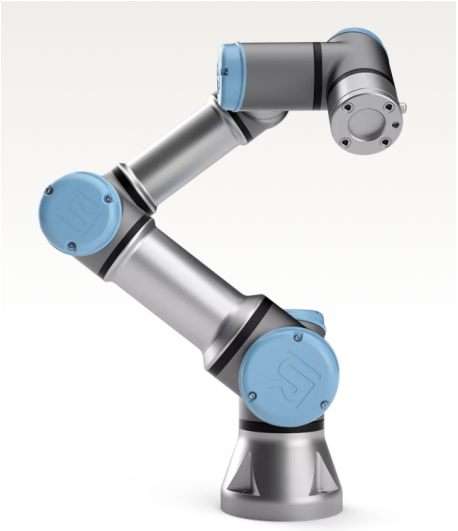

Figure 3.3: UR3 robot [4]

Figure 3.4: UR3 Gripper

3.1.2 UR3 Robot

The Universal Robot 3 seen in figure 3.3 is a more powerful robot used for light assembly and automation. It is developed by Universal robots. UR3 weighs about 11 Kgs and it can carry payload up to 3 Kgs and has a slightly larger reach of 50cm. Due to reasons described in 4.3.1, the user study did not include UR3 robot to maintain the time plan. However UR3 robot model was incorporated in the simulations where the user were able to switch between the robots using a UI. The UR3 robot does not include a built in gripper. However, URDF models of UR3 robot with Robotiq 2-Finger 140 mm gripper[23], shown in figure 3.4, is available as open source, which was used in this thesis work.

3.1.3 Unity

Unity is a 3D game engine from Unity Technologies and is popular for game development, augmented reality (AR) and virtual reality(VR) applications. A majority of AR/VR applications are developed using Unity. Unity ecosystem supports several extended plugins that enable functionalities like, TCP connection, ROS interactions etc,. In this thesis we use Unity to develop AR applications to enable visualization of robot plans.

Unity supports importing of robot models in URDF format provided by the developers of the robot. The URDF files are represents a robot model in XML format. These robot models enable movement of robotic arms in x,y and x axis and emulate physical properties like, gravity, damping, friction etc.

3.1.4 Mixed Reality Toolkit

Mixed Reality Toolkit (MRTK) is used in this thesis to enable deployment of the app in Microsoft Hololens 2. MRTK enables user interaction using spatial buttons

that can be interacted by the users while in AR mode.

Mixed Reality Toolkit is a tool developed by Microsoft that enables development of AR apps in Unity. It provides the building blocks for spatial interactions and user interface. MRTK provides in-editor simulation that enables rapid prototyping. MRTK supports wide range of platforms including, Windows XR, Occulus XR, iOS etc.

3.1.5 Vuforia

Vuforia Engine is a software development kit (SDK) used in AR apps. The SDK provides advanced computer vision functionality enabling images and recognition and providing intuitive options to interact with the real world.

A specifc example of Vuforia engine is to activate a robot in AR mode on detection of a specific pre-configured image. This features enables positioning of the AR robot in a pre-defined position irrespective of the orientation of the AR hardware.

We are using Vuforia to setup activation of a specific robot when the AR headset detects a predefined image using the built in camera.

3.1.6 Robot Operating System

The Robot Operating System (ROS) is an open source collection of robot components, tools and libraries that enables development of robot applications. ROS enables development of a robot using a "Digital Twin" version of the robot instead of using the real robot. This increases the development speed providing a shorter time to market. Using this approach, a robot functionality can be tested in advanced before actually deploying the software to the real robot.

In this thesis we use ROS to communicate with the Unity project and calculate the trajectory of the robot arms to perform a task.

3.1.7 MoveIT library

MoveIt is a robot manipulation platform for ROS, and supports the features like motion planning, manipulation, 3D perception, kinematics, control, and navigation. MoveIT can be used to calculate a robotic joint trajectory to move the robot from an initial position to the goal position. The symbolic AI planning solution generated from solving a PDDL problem can be used to invoke the moveIT library and to feed it with the initial and goal positions of the robot arms to generate a trajectory required to achieve this.

This trajectory is key in executing the planning solution in simulations in Unity or in AR.

3.1.8 Hololens2

Industry 4.0, deals with immersive 3D overlays, wireless/hands-free solutions and mixed reality visualizations that provide more value in manufacturing. Mixed reality consists elements of AR and VR. The user is able to see and interact with digital models and data. The transparency of the device allows the user to continue their work while experiencing realistic model and information.

Microsoft Hololens2 is mixed reality device. It allows the user to visualize the real environment around them while adding 3D holograms to it. The holograms can be interacted by the users through hand gestures or voice commands. The holograms can be animated and interactive.

The AR project developed in Unity can be deployed in Hololens2 device. With this approach the Unity project can be simulated in a real world environment where the components of the Unity project are superimposed in the real work and can virtually interact with the real world.

Figure 3.5: Microsoft Hololens 2 [5]

3.2 Automated Planning

Automated Planning involves the abstraction of various tasks from basic or low-level to high-level. The tool known as a *Planner* is responsible for determining the sequence of actions required to execute these high-level tasks. To ensure the planner generates the optimal set of high-level actions, the task must be defined in a specific format. The Planning Domain Definition Language (PDDL) serves as a standard for artificial intelligence planning languages. PDDL provides a method for defining planning problems, and it originated in the early 2000s for an international Planning Competition, evolving and maturing since then. A planning task comprises two components: a domain description and a problem description, both saved with the extension **.pddl**.

3.2.1 Planning Domain specification

The domain part, consists of the following components,

- **Predicates**: A predicate is a statement that represents a property about the state of the world in a planning problem and they can be either true or false. Predicates are used to describe the various aspects of the planning domain, specifying the possible states and conditions that can exist.
- **Actions**: Actions are the basic building blocks used to model the operations that can be performed in a planning domain. Actions represent the possible changes that can occur in the world state as a result of performing specific tasks. Definition of an action in PDDL involves *parameters* involved in the action and *preconditions* that must be true in the current state for the action to be applicable or executable.

.
Below is an example of a predicate definition.

```
%Example of predicates
    (:predicates
        (on ?b - box ?loc - location)
        (graspable ?b - box ?hand - robot)
    )
```

The above example can be interpreted as

- *True iff box 'b' is on location 'loc'.*
- *True iff box 'b' is graspable by robot 'hand'.*

.

Below is an example of an action called "grasp". Given the precondition that the object is clear, robot hand is empty, object is graspable and object is on a specific location, the actions *Grasp* will change the state to object not being clear, robot hand not being empty, object in robot hand and object not being graspable any longer.

```
%Example of an action
(:action grasp
    :parameters(?hand - robot ?obj - box ?loc - location)
    :precondition(and (clear ?obj) (handEmpty ?hand) (not ...
        (inhand ?obj ?hand)) (graspable ?obj ?hand) (on ?obj ?loc))
    :effect(and (not (clear ?obj)) (not (handEmpty ?hand)) ...
        (inhand ?obj ?hand) (not (graspable ?obj ?hand)))
)
```

This logic can be extended to include logical operators like *AND*, *OR*, *NOT*, and *IMPLIES* etc., which enables expressing complex conditions and effects. The domain in planning defines the set of actions that produces an effect. The effects are expressed as a predicate formula.

3.2.2 Planning Problem specification

The problem part, consists of the following components,

- **Objects**: They are entities that exist in the planning domain. Objects are used to represent specific instances of types within the domain. These types can include things like locations, people, vehicles, or any other relevant entities in the problem being modeled.
- **Initial states**: The initial state of objects is a specification of the initial conditions or state of the world in a planning problem.
- **Goal**: Goals specify the state of the world that the planner is aiming to reach through a sequence of actions. The goal description is part of the problem description and provides the criteria that a plan must satisfy to be considered a valid solution.

.
Below is an example of object definition. The objects in this specific planning problem includes a robot, a container, a location and two boxes.

```
%Example of object definition
(:objects
    robot - robot
    container - container
    table - location
    cube_blue cube_red - box
)
```

The planning problem also specifies the initial state of the world where the planning is executed. The below examples states that the blue cube is on the table, the blue cube is not in the container, the robot hand is empty and the blue cube is not graspable by the robot.

```
%Example of initial conditions
(:init
    (on cube_blue table)
    (not (in cube_blue container))
    (handEmpty robot)
    (not (graspable cube_blue robot))
)
.
```

The goal is the final expected predicate state of the objects involved in the planning. The expected state is achieved by performing the actions defined in the domain. Below is an example of a goal. The goal specifies that the blue cube must be in the container, the red cube must be on the blue cube and the robot hands must be empty.

```
%Example of a goal
(:goal
    (and (in cube_blue container) (on cube_red cube_blue) ...
        (handEmpty robot) ))
```

3.2.3 Planners

While PDDL is used to define a AI Planning problem, an AI Planner helps in solving the planning problem. The AI planner takes the PDDL domain and problem as input in PDDL and provides a solution to achieve the goal, if it exists.

There are a wide variety of planners and serve different purposes based on the domain and problems. Fast Downward planner[24] is used in this thesis. It is a classical heuristic planning system. Fast Downward is a progressive in nature and searches states of a planning task in the forward direction. In this thesis, we use the A*[25] with Landmark-Cut[26] heuristic variant of Fast Downward approach, which is popular due to its efficiency. Given a graph, a source node and a goal node, the A* algorithm finds the shortest path from source to goal.
The A* algorithm considers the exact cost g(n) of the path from the starting point to any vertex n in a graph, and the heuristic cost h(n) from vertex n to the goal. A* balances g(n) and h(n) while moving from the starting point to the goal. Through each loop, the algorithm evaluates the vertex n that has the lowest g(n) + h(n).

The solution of a planning problem is symbolic and defines which actions should be performed on objects, in which order, to achieve the desired goal. This solution is in turn used in executing the actions on real or AR objects. The planner tries to find a minimal sequence of actions that transitions the planning environment from the initial state to the goal state.

Below is an example of a solution obtained from PDDL planner using FastDownward algorithm obtained from the predicates, actions, object definition, initial conditions and the goal described in 3.2.1 and 3.2.2. The task for the robot is to pick up the *cube_1* from the *table* and placing it in a *container*, followed by picking up the *cube_2* and place it on top of *cube_1*.

```
1 %Example of a planning solution
2 (pre_grasp robot cube_1 table)
3 (grasp robot cube_1 table)
4 (put-in robot cube_1 table container)
5 (release robot cube_1)
6 (pre_grasp robot cube_2 table)
7 (grasp robot cube_2 table)
8 (put-on robot cube_2 table cube_1)
```

4

Methods

This chapter will present the implementation details of the thesis work.

4.1 System overview

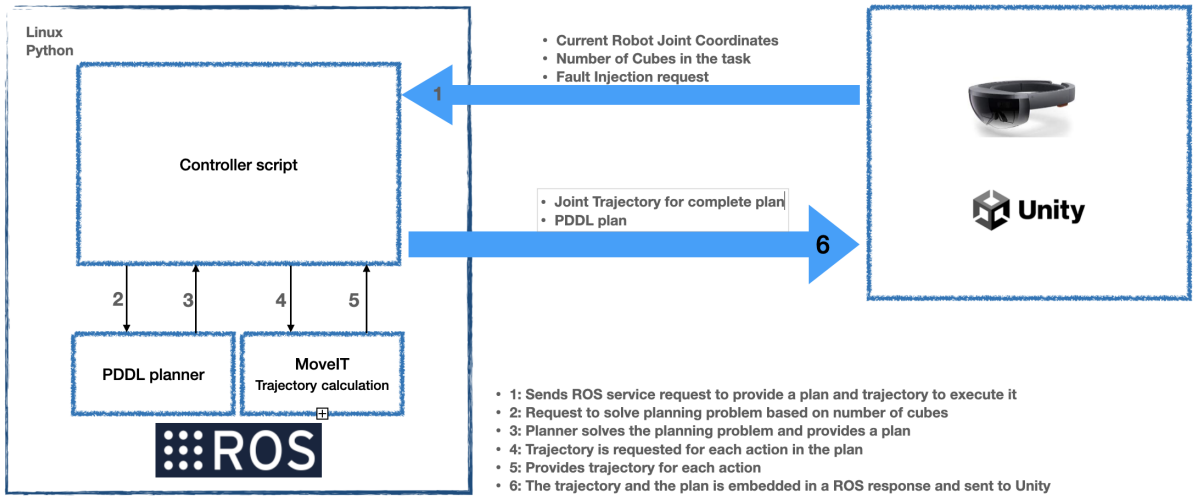

Figure 4.1: Overview of the project system.

Figure 4.1 shows the block diagram of the interconnection between Unity, ROS and Hololens2 and the flow of data between the blocks. The details of the working of each component is described in the later chapters.

The system employed in this thesis can be divided into two main components, as depicted in the figure 4.1. The first component, illustrated on the right side of the figure, consists of a 3D modeling and simulation environment. In this environment, various elements such as robot models, the surrounding environment, the user interface, robot trajectory execution, and mixed reality scenarios are modeled. A URDF model of the robot is utilized to facilitate control over the robot arms. A software solution is developed to command the robot arms along a specified trajectory. The Mixed Reality Toolkit (MRTK) is integrated into the project, configuring it for deployment to an Augmented Reality (AR) headset. A wireless communication channel is established to connect with an external computer responsible for services like solving PDDL problems and calculating robot trajectories. The user interface

is designed to visualize the execution of robot tasks, the PDDL plans, and specific use cases designed to evaluate the system's effectiveness in aiding users to identify the intention and failures in robot tasks.

The second component involves PDDL problem-solving and robot trajectory calculation, represented by the block on the left side of the figure 4.1. A ROS (Robot Operating System) server is set up on a computer, providing services to calculate trajectories based on the robot's current joint position and the final goal position. The robot requests a trajectory for a specific task. Upon receiving the trajectory request, the controller script running in the ROS service initiates a planning request to the PDDL planner. The planner responds with a PDDL plan. For each action in the plan, the controller script requests the MoveIT library for a trajectory. Once trajectories are received for all actions, the controller script packages them into a response message for the requester.

This system is designed to address the research questions outlined in section 1.1. The methodology for visualizing high-level plans in AR is detailed in section 4.2. Ultimately, the system serves as the foundation for conducting a user study essential for addressing research questions addressed in this thesis. The user study is elaborated upon in section 5.

4.2 Method to visualize of high-level plans in AR

The method uses the approach to visualize and playback the actions of the PDDL plan as shown in 4.4, before the actual execution of the robot action in AR.

- The user interface presents the option for the user to choose between different robots.
- After selecting a robot, the user initializes the task using a suitable UI interaction.
- The current joint configuration of the robot, along with the present positions of the objects involved in the task and the final goal positions, is supplied to the trajectory calculation algorithm.
- A PDDL solver is employed to solve the predefined problem, determining a sequence of actions to achieve the specified goal.
- The trajectory calculation algorithm utilizes the PDDL plan to compute trajectories, guiding the movement of objects from their initial positions to the goal positions.
- The PDDL plan is visualized to the user, providing insight into the planned sequence of actions, followed by the execution of the robot's trajectory to accomplish the task.
- Users are offered an interface to playback each action of the PDDL plan, enabling a detailed review of the executed sequence.

The high level method described above is implemented as described in the following sections.

4.2.1 Unity setup

AR applications are first developed in Unity in the computer and tested using the computer display. In the Unity environment, the modeling and simulation of the robot, UI, and trajectory execution are implemented. A view is set up within Unity, representing the environment of the thesis. Throughout the course of the thesis, various experiments were conducted, resulting in different environments surrounding the virtual robot. For instance, initially, a user study was planned using virtual cubes, and a corresponding environment was created using virtual cubes. However, recognizing the potential for increased realism, the decision was made to use real cubes, leading to the creation of a new environment accommodating these real-world objects.

URDF models of the robots, including Niryo and UR3 models, are integrated into the Unity project. Additional objects such as red and blue cubes are added to the environment, along with a goal position marked by a virtual white rectangular slab, as depicted in Figure 4.4.

The Unity project incorporates the Mixed Reality Toolkit (MRTK) and Vuforia packages. MRTK facilitates rendering the project in augmented reality (AR), enhancing the user's experience. Vuforia, on the other hand, enables marker-triggered placement of virtual objects relative to the spatial configuration of the Hololens2 device. In this setup, an image serves as the marker, causing the virtual robot and the user interface to appear when the Hololens2 camera detects the pre-configured image.

To enrich the user experience and interaction with the environment, suitable user interface elements such as buttons and message boxes are added from the MRTK library. These assets enable the users to interact with the environment and control the robot in AR mode.

Both Niryo and UR3 robot models are setup in the simulation phase of the thesis. Figure 4.2 presents the view when Niryo robot is selected and figure 4.3 presents the view with UR3 robot.

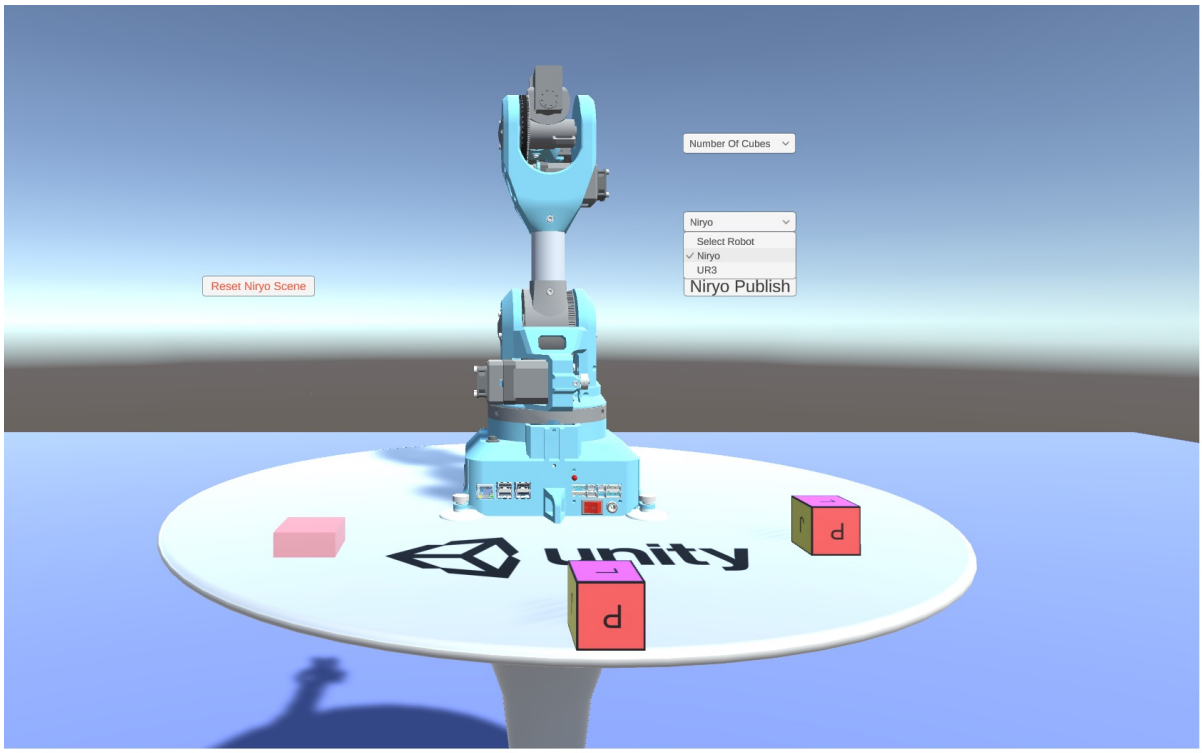

Figure 4.2: View of the Niryo robot in Unity simulation

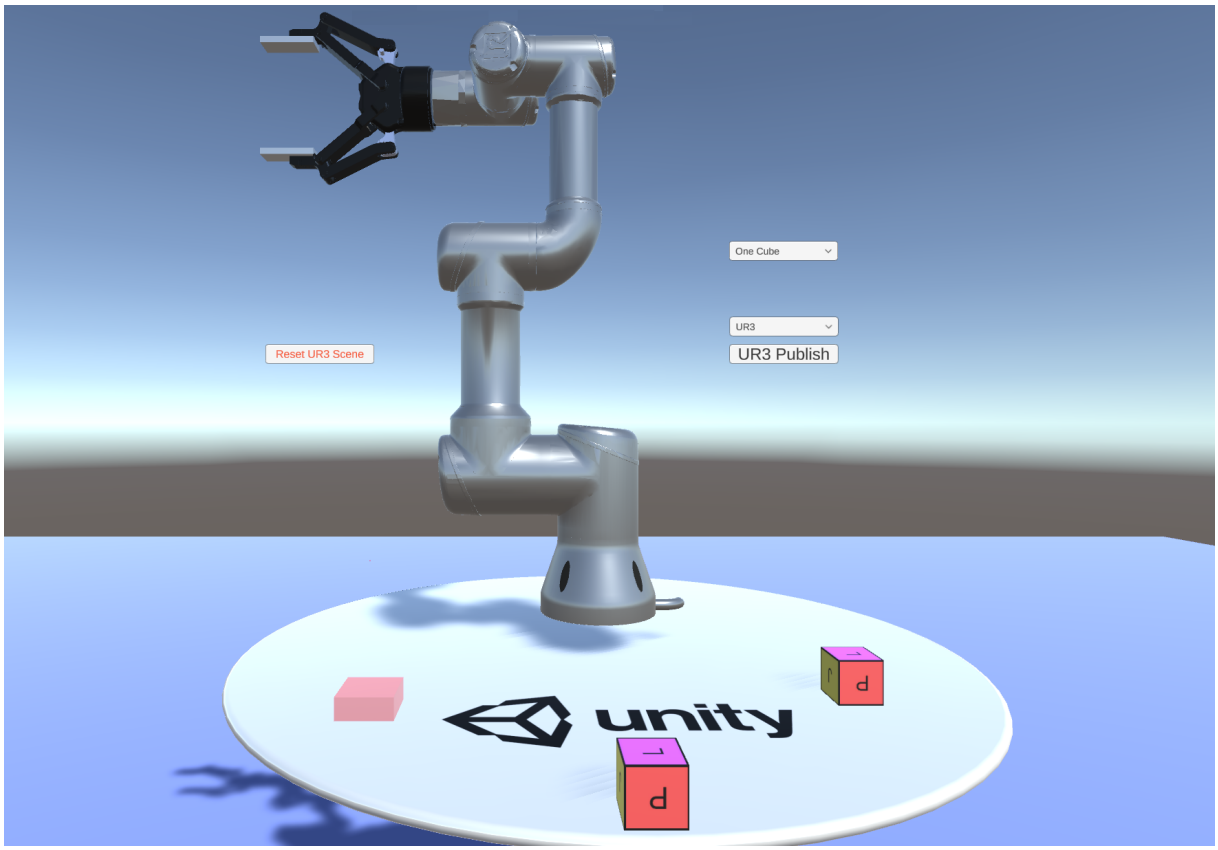

Figure 4.3: View of the UR3 robot in Unity simulation

4.2.2 Pick and Place Task

The primary task employed in the thesis revolves around the picking and placing of two cubes seen in the figure 4.4. The virtual robot engages in the following sequence of actions:

- The robot initiates by picking up the blue cube from its initial position.
- Subsequently, the robot transports the blue cube to the target location.
- The robot then proceeds to pick up the red cube from its initial position.
- The red cube is then stacked on top of the first cube.

This task serves as a fundamental scenario for testing and evaluating the capabilities of the system, involving both PDDL problem-solving and robot trajectory execution.

4.2.3 Environment description

With reference to figure 5.1 the main view for the thesis used in the user study consists of the below,

- Niryo Robot model,
- A real red and a real blue cube,
- Virtual slabs that takes the color of the cube being placed in the target position. The slab is white at the start of the environment as seen in 4.4 and is programmed to change color when a cube is successfully placed in its goal position.
- Niryo button: Pops-up the Niryo robot,
- Publish button: To initiate the task,
- Reset button: To reset the view containing the AR robot and its environment,
- Action Notifications: These boxes provide real-time notifications about the ongoing action being executed by the robot. Each block is pressable, allowing the user to replay the specific action. This feature is valuable for users to gain a detailed understanding of the events that occurred during a particular action. In figure 5.1 the blue boxes containing text like, *PreGrasp Cube Blue* etc are action notifications,
- Next use case button: To switch to the next use case,
- Previous use case button: To switch to the previous use case,
- Current use case Block: This block is used to present the current use case number. This block is not pressable and is for notification only.

Together, these elements contribute to an interactive and informative user interface, enabling users to actively engage with and comprehend the robot's actions and the ongoing use case.

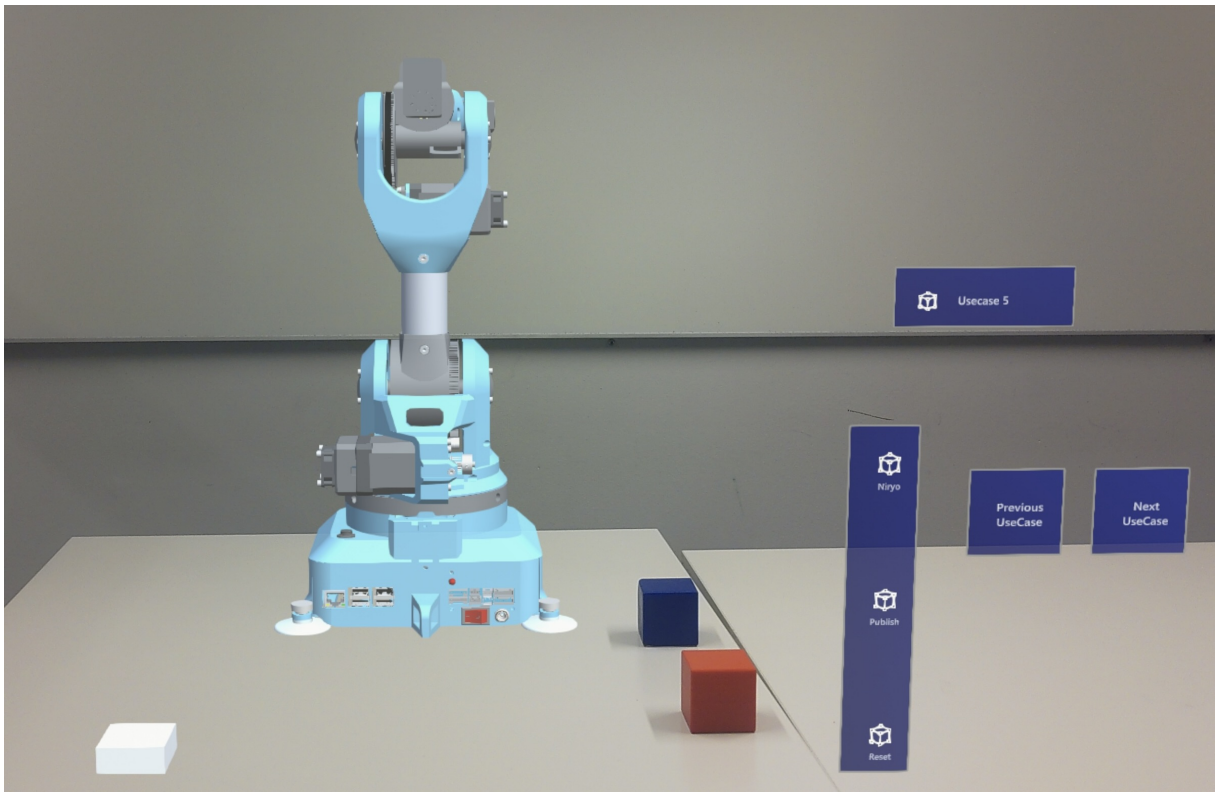

Figure 4.4: View of the robot and its environment in user study

4.3 Trajectory Execution methods

During the course of the thesis two types of trajectory execution was implemented.

4.3.1 Joint position based trajectory execution

This solution uses the trajectory received from ROS and moved the robot joints by rotating each joint according to the received trajectory. This solution was implemented to display robot joint position movements on the unity side and instead to handle the physical controllers part of it on the ROS side. This would enable experimentation of different physical controllers and not rely on the ones provided by Unity ecosystem.
In this thesis, the trajectory execution based on joint position control is implemented as shown in the below code snippet.

```
1 var result = trajectory_point.positions.Select(r => (float)r * ...
      Mathf.Rad2Deg).ToArray();
2 shoulder_link.localRotation = Quaternion.Euler(0, ...
      -(float)result[0], 0);
3 upper_arm_link.localRotation = ...
      Quaternion.Euler(((float)result[1]), 0, 0);
4 forearm_link.localRotation = Quaternion.Euler((float)result[2], ...
      0, 0);
5 wrist_1_link.localRotation = Quaternion.Euler(90 + ...
      (float)result[3], 0, 0);
6 wrist_2_link.localRotation = Quaternion.Euler(0, ...
      -(float)result[4], 0);
7 wrist_3_link.localRotation = Quaternion.Euler((float)result[5], ...
      0, 0);
```

In order to achieve this, the component called articulation body [27] which is a part of the robot URDF model had to be removed from the robot model in unity. Articulations are objects comprised of links, each of which behaves like a rigid body, connected together with joints. Articulation bodies enable in building physics articulations such as robotic arms. They provide a realistic physics-conforming behavior to robot applications. In this case if the articulation body component is not removed, it would oppose the movement of the robot arms by changing local rotation components of the joints. The articulation body uses properties of physics and has inertia. It waits for a physical drive command to enable movement of robot arms and without this drive, it tries to keep the robot arms in its current state.

4.3.2 Physical articulation based trajectory execution

This approach uses the xDrive to move the robot joints. A drive, applies forces and torques to the connected bodies likes robot joints. xDrive corresponds to Drive in X axis.
In this approach, an articulation body is used and the robot arms are moved by by assigning a target to the xDrive component of the robot joints.The xDrive component

moves the robot joints to the desired target using properties of physics. The below code block shows the robot trajectory control using xDrive.

```
var result = jointPositions.Select(r => (float)r * ...
    Mathf.Rad2Deg).ToArray();
for (var joint = 0; joint < m_JointArticulationBodies.Length; ...
    joint++)
{
    var joint1XDrive = m_JointArticulationBodies[joint].xDrive;
    joint1XDrive.target = result[joint];
    m_JointArticulationBodies[joint].xDrive = joint1XDrive;
}
```

In this implementation, the gripper captures the object by pressing against the object and creating enough friction to grab the object. The code block below descries a gripper closing action.

```
public void CloseGripper()
{
    var leftDrive = m_LeftGripper.xDrive;
    var rightDrive = m_RightGripper.xDrive;

    leftDrive.target = -0.01f;
    rightDrive.target = 0.01f;

    m_LeftGripper.xDrive = leftDrive;
    m_RightGripper.xDrive = rightDrive;
}
```

This method is implemented on Niryo one robot and the results looked realistic and accurate.

4.3.3 Issues with position based control of robot joints

Movement of robot arms based on local rotation of the joints worked well after removing the articulation body component. However this produced a side-effect that made gripping of the object look unrealistic. Since articulation body component was removed, it was not possible to grip the object using friction. Instead the gripper pad position had to be moved closer to the object as if it looked to be gripping the object and then the gripper pad was made the parent of the object so that the object followed the gripper pad. Below is the code block showing the function that closed the gripper pads in this method.

```
void CloseGripper() {
    target_rigidbody.useGravity = false;
    m_Target.transform.parent = right_inner_finger_pad;
    right_inner_finger.localRotation = Quaternion.Euler(0, 0, 70);
    left_inner_finger.localRotation = Quaternion.Euler(0, 0, 70);
```

```
6       right_inner_finger_pad.localRotation = Quaternion.Euler(0, ...
            0, 40);
7       left_inner_finger_pad.localRotation = Quaternion.Euler(0, 0, ...
            40);
8   }
```

As seen in 4.5, although the object now moved along with the gripper giving a sense of it being gripped, parts of the gripper went through the object making it look unrealistic. This result was obtained using the UR3 robot which was also in the scope of the thesis. Due to time constraints this approach was not used for the user study and so was experimentation with UR3 robot.

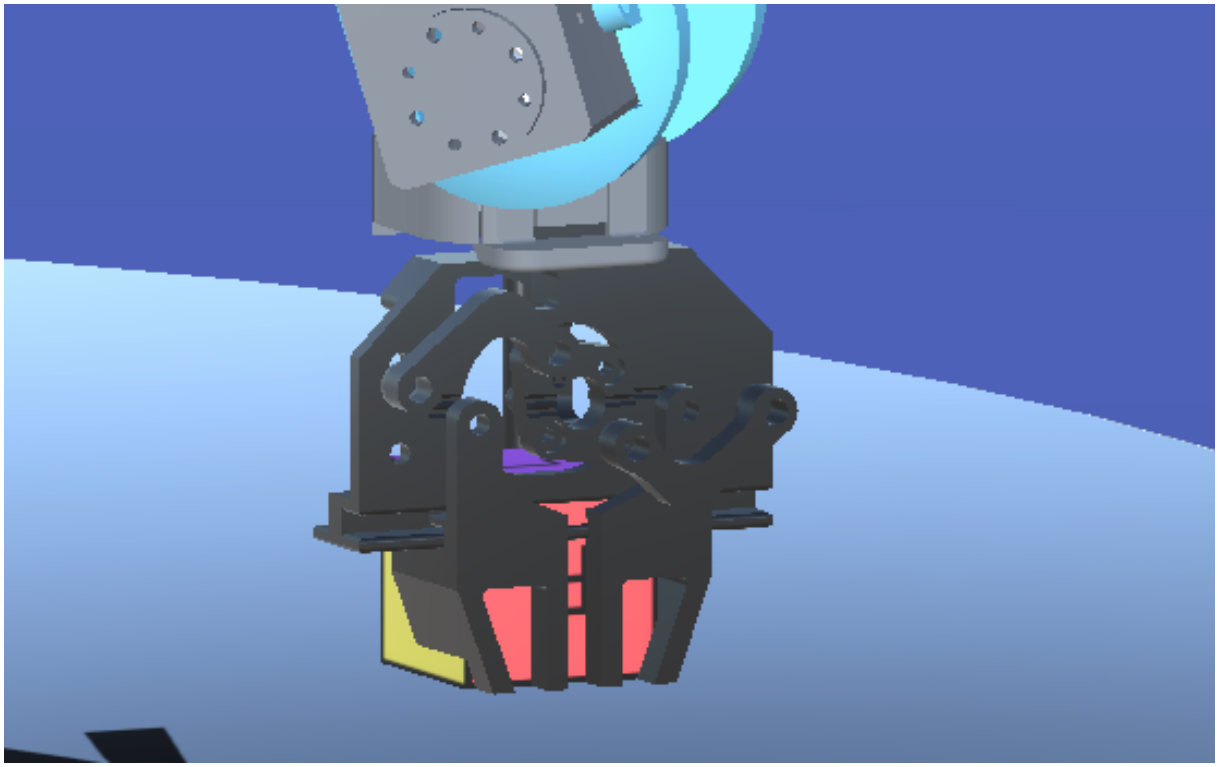

Figure 4.5: Gripper and object colliding is seen clearly in the image. The gripper seems to be inside the boundaries of the cube.

4.4 Robot trajectory calculation method : ROS Setup

In the ROS side of the project system, the PDDL solution and trajectory calculation are carried out. The PDDL planner is integrated as a new ROS package [28]. This package includes the predefined PDDL domain and problem, along with a Python script that invokes the PDDL solver.

When the ROS server is initiated, the PDDL service is also started, making the ROS setup ready to receive requests from the Unity project and respond with the

trajectory.
The subsequent section details the data exchange process between Unity/Hololens2 and ROS during the execution of a task.

4.4.1 ROS service

A request and a response together form a ROS service [29]. The below code block shows the new service created in the thesis. The first part of the service file describes the request message published from Unity/Hololens2 and the second part contains the response sent from ROS.

As can be seen, tasks with up to 2 cubes are supported by this service message. The service request includes the position of the cubes, the current robot joint configuration, the number of cubes in the task and information about the type of fault to be induced when needed.

The response contains the PDDL plan that will be visualized to the user, along with the joint trajectory that will be used to control the robot.

```
%Request
NiryoMoveitJoints joints_input
geometry_msgs/Pose pick_cube_blue_pose
geometry_msgs/Pose pick_cube_red_pose

geometry_msgs/Pose place_cube_blue_pose
geometry_msgs/Pose place_cube_red_pose
int8 number_of_cubes
int8 fault_injection

%Response
---
string[] plan
moveit_msgs/RobotTrajectory[] trajectories
```

Figure 4.1 shows the data exchanged between Unity/Hololens2 and ROS. While in simulation mode (Unity <–> ROS) the data is exchanged by TCP and while in deployed mode(Hololens2 <–> ROS), the data exchange happens via WiFi and TCP.
The order of data flow,

- Unity/Hololens2 sends a service request to ROS server
- The ROS server calls the PDDL planner
- PDDL planner runs the fast downward algorithm [30] and if a solution is found, it returns the PDDL plan similar to the description in section 3.2.3
- ROS server then calls the MoveIT[31] library according to the received PDDL plan.
- MoveIT responds with the trajectory.
- ROS server packs the complete trajectory in an array and sends the service response back to Unity/Hololens2 along with the PDDL plan

4.4.2 Setup connection to Hololens2

Hololens2 communicates with the ROS project through WiFi. While building the software in Unity, it is possible to specify the ROS IP address in the ROS prefab. Unity will communicate to this IP address. During experimentation it was observed that Hololens2 communication is stable on 2.4GHz channels compared to the 5GHz channel.

The ROS IP address could change if a new router is used to connect the Hololens2 and the computer running ROS project. In this case, the IP address has to be updated in ROS prefab on the Unity side and the software built again and deployed to Hololens2.

ROS was next implemented both on a docker container and in a VirtualBox instance on Windows.

4.4.2.1 Docker vs Virtualbox for ROS

Initial Implementation in Docker Container: In the initial stages of the thesis, the ROS project was implemented within a Docker container. This approach proved successful for simulation mode, allowing seamless communication between the ROS project and the Unity project when both were running on the same computer. However, challenges arose when deploying the Unity project to Hololens2, leading to communication failures between ROS and Unity.

Transition to VirtualBox: To address the communication issues, the approach was shifted to using VirtualBox. This transition successfully resolved the problem, enabling effective communication between ROS and Hololens2. It's important to note that only the BridgedAdapter type of network setup resulted in a successful connection.

4.5 Deployment and test execution

The Unity project is build and deployed according to the instructions in [32]. The deployment is done using Visual Studio tool. Deployment can be don using a wired connection to Hololens2 or by wireless means using WiFi. In order to connect to Hololens2 using WiFi the IP address of Hololens2 must known in advance.
Once deployed on Hololesn2, the Unity project or the App starts automatically presenting the robot and its environment to the user. When Vuforia is used the robot model pops-up when the Hololens2 camera detects a predefined image. For example, the image is printed and placed on a table. The robot will then appear on the table. The robot model can also be activated through a AR button presented to the user as seen in figure 4.4. For the user study the virtual button is used to activate the robot instead of Vuforia. On pressing this button the robot pops-up at a pre-defined position relative to the position of Hololens2.

5

User Study

The user study is the major part of the thesis that will enable understanding if visualizing the PDDL plan to the users helps the users to comprehend the intention of the robot and detect robot faults when occurred, thus answering the research questions section in 1.1. This user study included 4 participants in the age group 25-35, with 50% of the participants had not prior experience with AR applications.

In preparation for the user study, 10 use cases were identified. These use cases included both successful and failure cases. Out of the 10 identified use cases, seven are designed to simulate failure scenarios represented by figures 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8, while the remaining three consist of repeated instances of the same successful cases. Below are the brief description of the successful and failure cases.

Successful case: In the context of the user study, the successful case involves the successful stacking of the red cube upon the blue cube. The visualization for the successful case is illustrated in Figure 5.1, where the virtual red slab is stacked on the virtual blue slab, indicating the desired outcome of the robot task. The two rows of visualized plan represent each action performed by the robot, as derived from the solution to the PDDL problem.

Failure cases are categorized into 4 areas as described below,

- Plan-Execution mismatch
  - 5.2.1 : Blue cube stacked on red cube
  - 5.2.2 : Blue cube stacked on red cube's original position
- Trajectory execution failures
  - 5.3.1 : Early release of blue cube
  - 5.3.2 : Gripper does not close while grasping cubes
- Incorrect data in service request
  - 5.4.1 : Incorrect goal position for blue cube
  - 5.4.2 : Incorrect pick position for red cube
- Incorrect Plan
  - 5.5.1 : Missing grasp action for blue cube

For the user study the three successful cases and the 7 failure cases were randomly presented to the users aiming to obtain more accurate feedback from the user study, providing insights into the user's ability to comprehend and respond to different scenarios, based on the plan visualization.

5.1 User study procedure

The user study is initiated by presenting participants with a detailed description of the AR scene, including visualized objects and the user interface. Participants wear the Hololens2 headset with the thesis app pre-started. Users are introduced to the main view and provided with an explanation of the study procedure. They are given the opportunity to familiarize themselves with a default scene and successful case, and any necessary clarifications are provided.
The user study progresses as follows:

- **Use case Initialization**: Participants begin with use case 1 and initiate the task. The robot starts executing actions/trajectory in response to the user's input. Before the start of each action, an information block with a brief description of the action is presented to the user.
- **Task Execution**: The robot proceeds to execute the action in alignment with the PDDL plan.
- **Questionnaire**: After the completion of each use case, participants fill out a questionnaire (refer section 5.5.2) indicating whether they consider it a success or failure. Optionally, participants can provide additional comments explaining their assessment.
- **Repetition**: The process is repeated for all identified use cases, with participants progressing through each one. Participants have the flexibility to express their opinions on the success or failure of each use case, and the optional comments allow for qualitative insights into their reasoning.

To maintain a controlled environment, only one participant was allowed in the project room during each session.

The user study was conducted at Chalmers University of Technology campus in Johanneberg, Gothenburg, Sweden, with the participation of four users.

5.1.1 Use case 3,4 and 7 : Successful case

These are the successful cases where the red cube is stacked on the blue cube as expected. To enhance the comprehension of the robot actions, the success case is visualized by the virtual red slab being stacked on the virtual blue slab. The information block describing the actions corresponds to the actions being executed. A successful use case is shown to the users prior to the user study to help them to identify the task the robot is expected to perform.

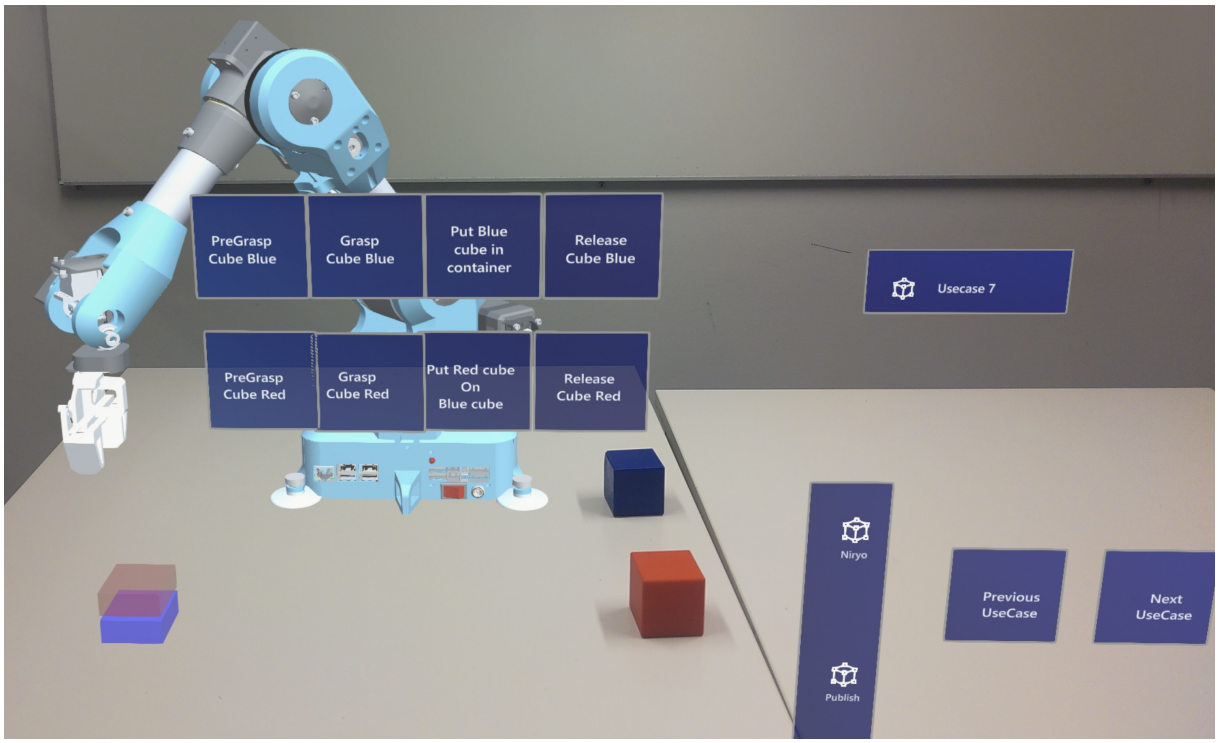

Figure 5.1: Successful case. This use case represents the scenario when the task has been completed successfully. Since real cubes are used, the virtual robot cannot interact with them. Therefore the successful completion of the task is indicated by the stacked virtual slabs. In the image the red slab is stacked upon the blue slab indicting that the virtual robot has performed the task successfully.

5.2 Plan-Execution mismatch

These types of failures occur when the plan communicated from ROS is incorrect or if the execution is incorrect.

5.2.1 Use case 1 : Blue cube stacked on red cube

In this use case, the plan visualised differs from the executed actions. The information block describes that the blue cube is being grasped, but in reality the red cube is picked. Therefore, at the end of the scene the blue cube is stacked on top of red cube instead of stacking red cube on top of the blue cube. This can be understood by reading through the sequence of information block from left to right as seen in the image 5.2 and also observing the AR placement position of the cube marked by AR slabs bearing the same color as the cubes in the scene.

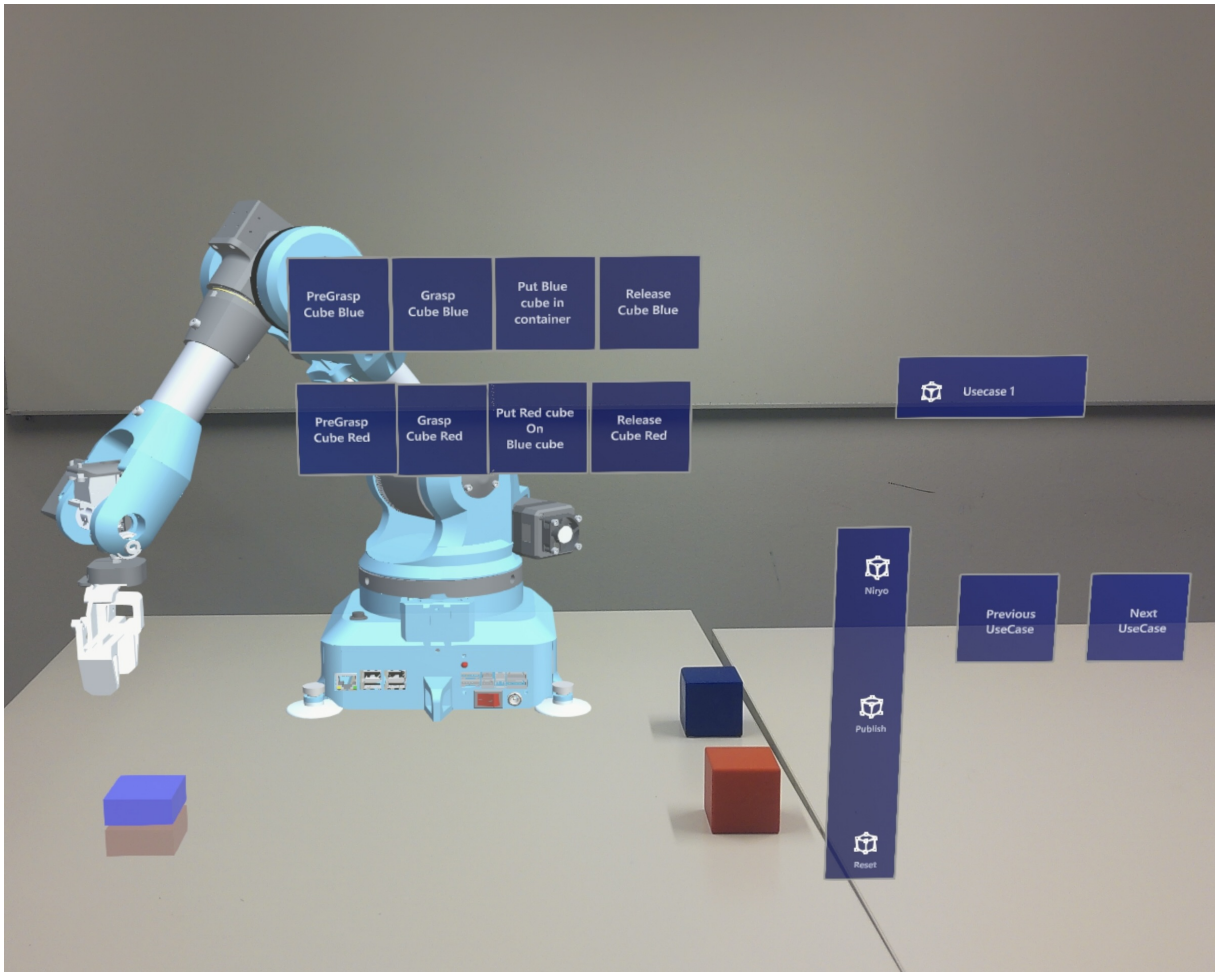

Figure 5.2: use case 1. The use case looks like a successfully scenario. However, the stacking order of the cubes are incorrect. The blue slab on the red slab indicates that the virtual robot picked the red cube first instead of the blue cube.

5.2.2 Use case 2 : Blue cube stacked on red cubes original position

In this failure use case, the blue cube is stacked on the red cube, in the original position of the red cube, instead of placing the blue cube in the position marked by the AR blue slab. As seen in the image 5.3, the third information block reads "Put blue cube in container" but the action being performed corresponds to "Put blue cube on red cube" which clearly is a failure.

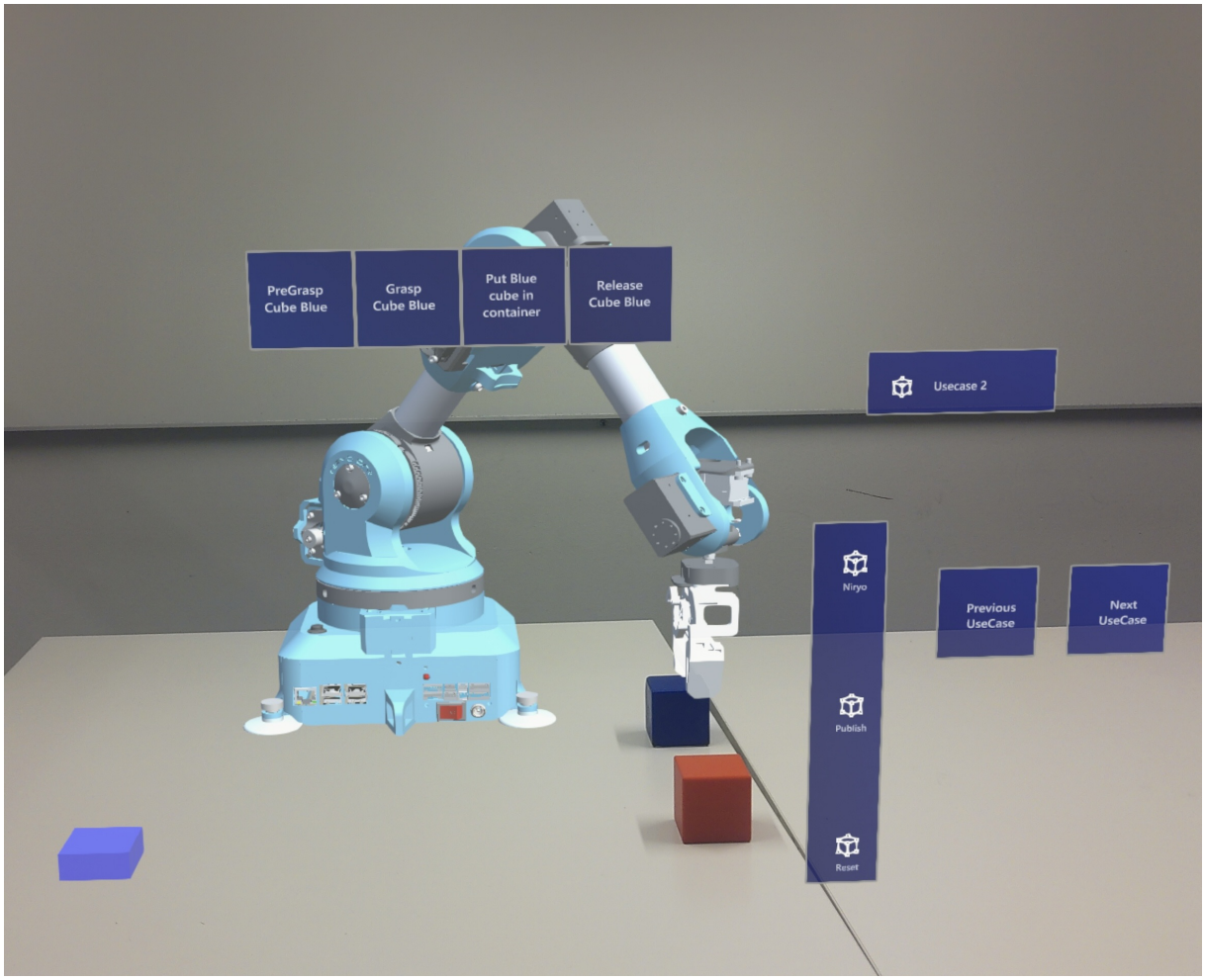

Figure 5.3: Use case 2. This figure shows the end of use case 2. The robot picked the blue cube only and placed it in the position of the red cube instead of placing it in the container.

5.3 Execution failures

These failures might occur when the robot actions are incorrect. The plan and its actions are correct and the visualized action block informs the user the upcoming action of the robot but the robot executes the action incorrectly.

5.3.1 Use case 5 : Early release of blue cube

In this use case, the robot opens the gripper ahead of time. This results in the blue cube being dropped in an incorrect goal position shown by the blue slab in figure 5.4. The position of the red slab is the expected goal position. The result is that the cubes not stacked upon one another.

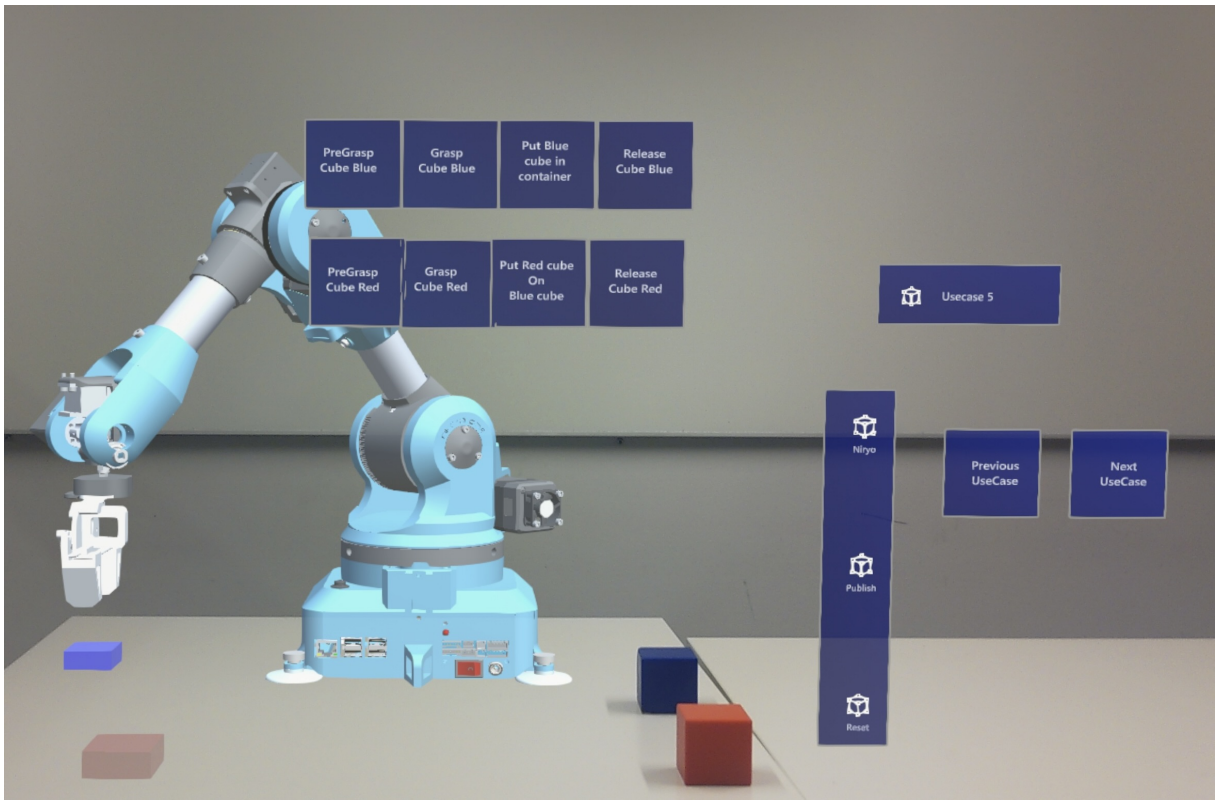

Figure 5.4: use case 5. The robot picks up the blue cube and drops the cube before reaching the goal position marked by the red slab.

5.3.2 Use case 6 : Gripper does not close while grasping cubes

In this failure use case, the robot fails to grasp the cubes. It places the grippers around the cubes but is unable to close the grippers. This results in the cubes being in its original position at the end of the scene execution. Figure 5.5 shows this use case during its execution. As can be seen in the figure, the robot has finished placing the blue cube but the AR slab is still white indicating that no cube wad dropped in the goal position.

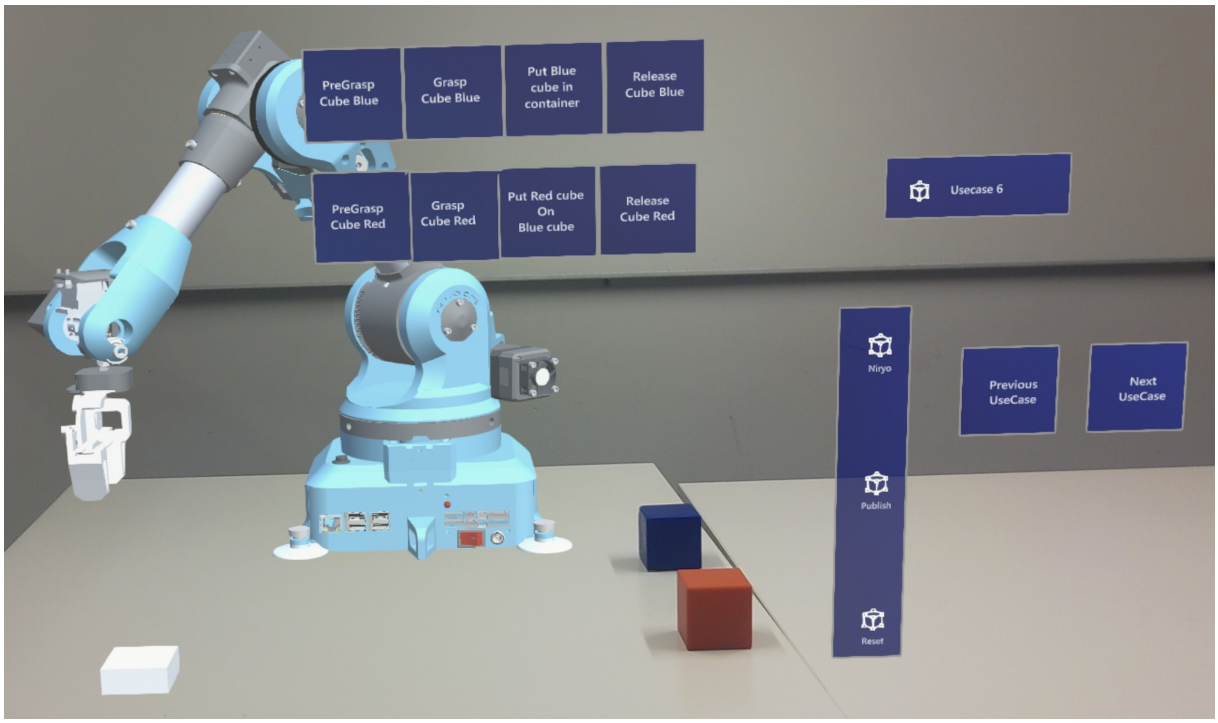

Figure 5.5: use case 6. In this use case the robot does not close the grippers which grasping the cubes. Therefore although the robot performs the actions, the cubes are not moved indicated by the white slab representing the goal position.

5.4 Incorrect data in service request

In these types of failures, the request contains incorrect data like incorrect initial position of the robot arms, or incorrect cube positions or incorrect goal positions. This will clearly result in the task being executed incorrectly even if the plan is correct and the robot executes the plan in a correct way.

5.4.1 Use case 8 : Incorrect goal position for blue cube

In this failure use case the goal position for the blue cube is specified incorrectly in the service request sent to ROS, resulting in the blue cube being place in the wrong destination. The correct destination is marked by the red slab in the figure 5.6

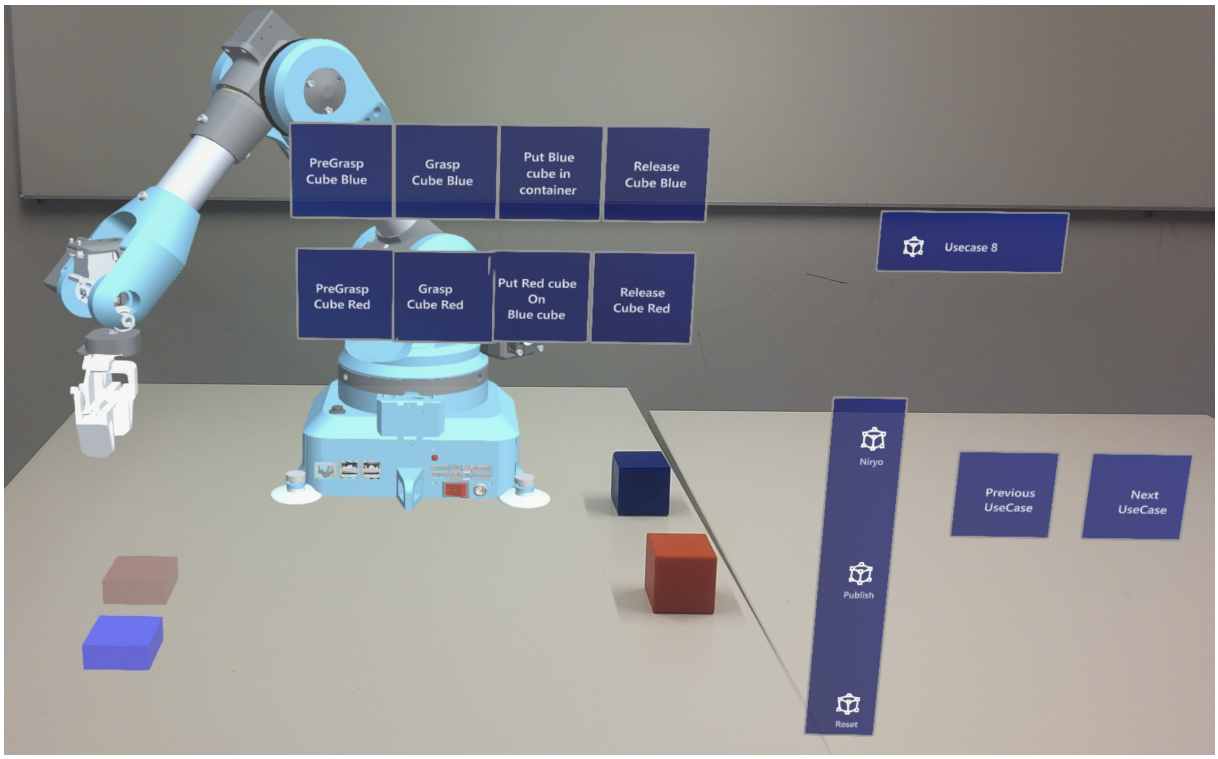

Figure 5.6: use case 8. The blue cube is not stacked on red cube but is dropped at the position farther away from the goal position marked by the red slab.

5.4.2 Use case 9: Incorrect pick position for red cube

As seen in figure 5.7 the robot tries to pick up the red cube from a position away from the actual location of the red cube. This can occur when the initial position of the red cube is transmitted incorrectly in the service request.

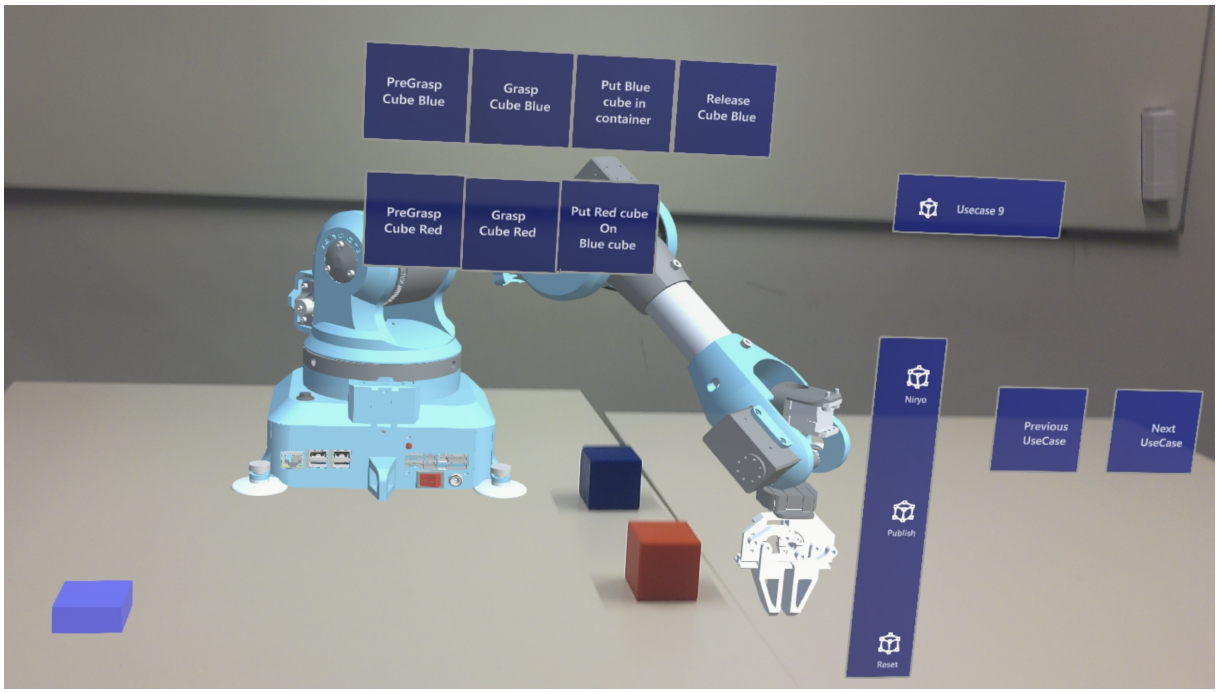

Figure 5.7: use case 9 : Incorrect pick position for red cube.

5.5 Incorrect plan

When the ROS server calls the PDDL planner, the solution can be incorrect if either the PDDL Domain or the Problem is specified incorrectly resulting in eventual failure in the task execution.

5.5.1 Use case 10 : Missing grasp action for blue cube

As seen in 5.8, at the end of this failure use case, only red AR slab is shown. This indicates that the blue cubes was not picked up from its position. That the failure is in the PDDL plan can also be observed in the presented plan. The presented action blocks is missing the action "Grasp" and this action is clearly missed by the robot resulting in the blue cube not moved from its initial position.

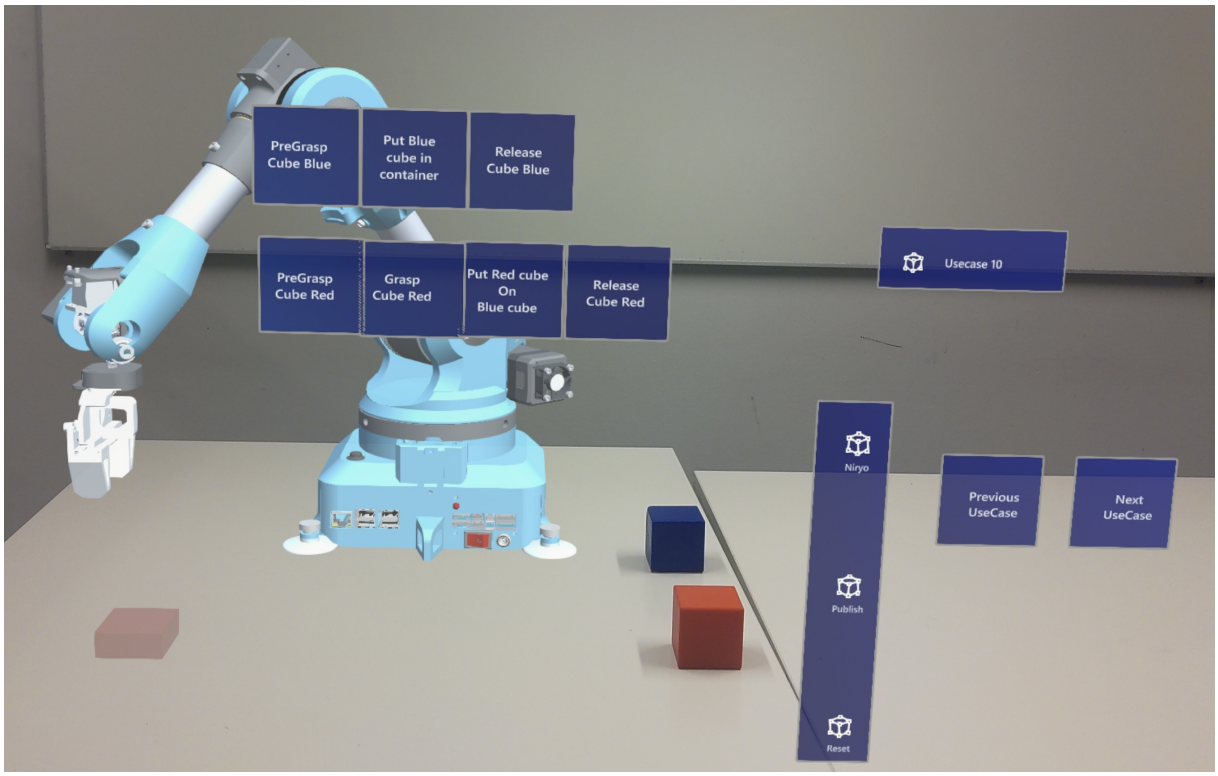

Figure 5.8: use case 10. The plan from the planner, does not include the action to grasp the blue robot. The robot places the gripper around the blue cube but does not pick it up.

5.5.2 Questionnaire

Figure 5.9 presents the questionnaire presented to the users participating in the user study. The use cases are numbered from 1 to 10 and each use case can be either market as "Successful" or as "Failure". The users can also write their observation for each use case explaining their reasoning behind identifying each use case as a success or a failure.

User Study Questionnaire

Master Thesis : Visualization of plans using Hololens 2

Use cases	Successful	Failure
1. UseCase1 Observations:	☐	☐
2. UseCase2 Observations:	☐	☐
3. UseCase3 Observations:	☐	☐
4. UseCase4 Observations:	☐	☐
5. UseCase5 Observations:	☐	☐
6. UseCase6 Observations:	☐	☐
7. UseCase7 Observations:	☐	☐
8. UseCase8 Observations:	☐	☐
9. UseCase9 Observations:	☐	☐
10. UseCase10 Observations:	☐	☐

Figure 5.9: Questionnaire used in the user study performed in this thesis

6
Results

6.1 User study outcome

Following the conclusion of the user study, questionnaires were collected from each participant, and the results were subsequently analyzed. The findings from the analysis of the answered questionnaires provide valuable insights into the participants' experiences and assessments of the use cases.

The analysis of the answered questionnaires yielded the following key findings:

Identification of Failure Use Cases: in 97% of the cases, the participants in the user study demonstrated the ability to identify the failure use cases presented to them. This indicates a successful understanding of scenarios where faults or failures were intentionally introduced.

Recognition of Successful Cases: Participants were also successful in identifying the successful cases among the use cases presented.

Precise Identification of Failures: Notably, in addition to identifying if a use case is a success or a failure case, the participants exhibited a high level of precision(97%) in identifying the exact failure that occurred in each use case presenting a failure. Identifying the success or failure of Use Case 5.2.1 proved challenging as the robot appeared to execute the task correctly. Only upon closer inspection by playing back the actions and comparing it to the visualised plans, did it become apparent that the stacking order was incorrect.

These findings collectively highlight the effectiveness of visualizing the PDDL plan to users in enhancing their ability to identify both successful and failure scenarios. Participants stated that by utilizing the ability to replay the robot's actions, enabled them to pinpoint specific failures and accurately assess each use case.

Following is the summary of the thesis outcomes,

- **Does AR visualization of the high-level plan allow users to detect potential failures before the execution on the robot?**: The user study data provided valuable insights into the effectiveness of robot plan visualization in aiding users to identify robot failures. Participants confidently categorised the successful and failure use cases by comparing the robots execution of the

task and comparing to the visualised plans. The visualization of the high-level plan allowed users to anticipate and detect potential failures, enhancing their overall understanding of the robot's intended actions.
These findings highlight the value of robot plan visualization in enhancing the user's ability to identify and understand potential failures and mis-specifications in the robot's high-level plan and actions.

- **How effective is the action [33] visualization in AR, in detecting action mis-specification?**: Use case 10 described in section 5.5.1 illustrates the case when the plan derived from the planner is incorrect. In this case the *Grasp Cube Blue* action is missing and therefore when the robot performs the actions "*Put Blue cube in container*" (see figure 5.8), the effect is not as expected because the precondition, which is the gripping of the robot, is not satisfied. The initial idea was to implement the action visualization, which should be popped-up when the user hovers over a specific action, but due to complexity and time constraint it was not implemented. However it can be seen in figure 5.8 that among the visualized actions, the *Grasp Cube Blue* action is missing. This helped the users identify that a failure had occurred and that the action "*Put Blue cube in container*" did not execute as expected due to a lacking pre-condition. Thus visualization of the list of actions helped the users identify failure cases.

- **Does providing tools to the user to playback the action aid in robot failure identification?** : The participants informed that having a system that enables them to playback each action of the robot aided in better interpretation of the intended actions of the robot and identify the deviations. The participants also replayed actions from a successful use case for comparisons between actions in different use cases. This helped them in better identifying the robot failures. With this enabled, the users were able to over the difficulty in identifying failures in corner cases like use case 1 (section 5.2.1).

6.2 Real cube vs virtual cube

Virtual Cubes in Simulations: In the simulation environment within Unity, the use of virtual cubes proved effective. This approach allowed users to visualize the cubes during the simulation, providing a clear representation of the robot's interaction with the environment. Without virtual cubes, the robot arms would move from the initial position to the final destination, and the gripper would open and close during the process. However, the absence of visual cubes made it challenging to determine if the robot actually grasped the objects. Figure 6.1 illustrates the simulation in Unity with virtual cubes.

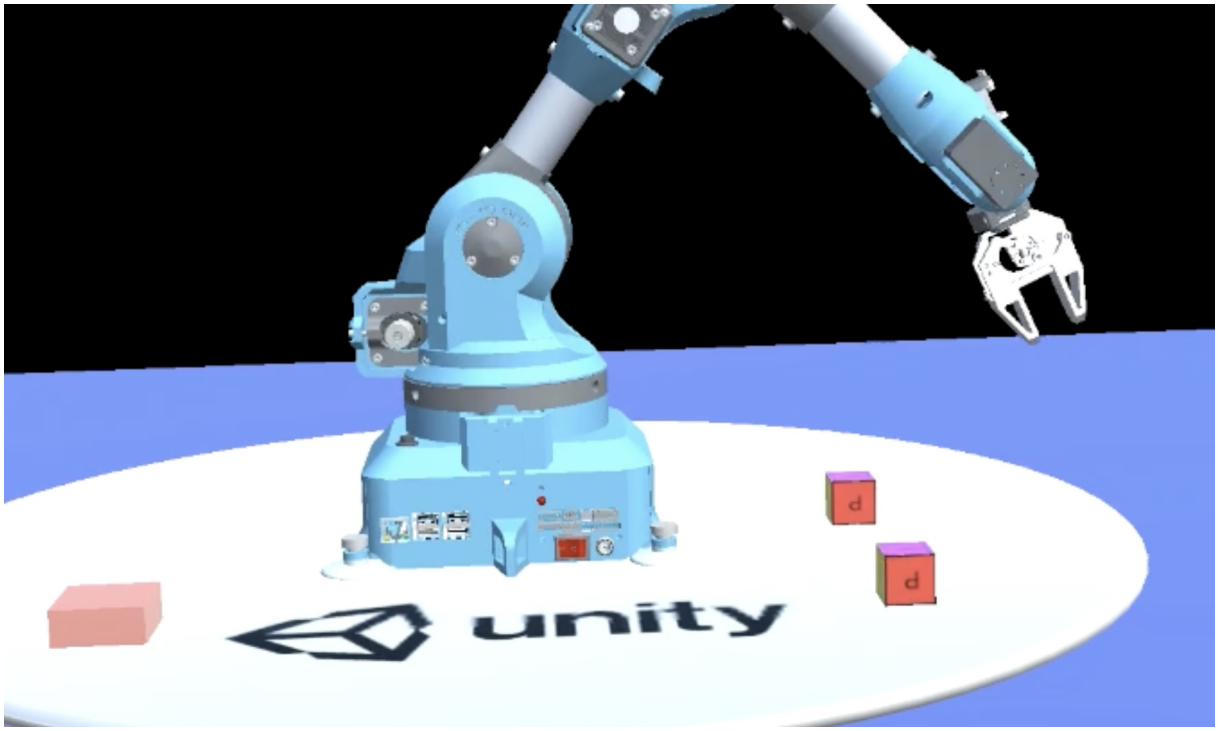

Figure 6.1: Scenario with virtual cubes

Real Cubes in User Study: For the user study conducted using Hololens2, real cubes were employed instead of virtual cubes since the intention was to avoid modeling of the environment as much as feasible. Figure 5.3 displays the virtual robot attempting to grasp a real cube. The visual cue using the virtual slabs aids users in understanding the progress and outcome of the robot's actions.
Each scenario presents its own set of advantages and considerations. The choice between virtual and real cubes depends on factors such as the desired level of realism, the intended user experience, and the practical constraints of the physical environment in which the AR robot is deployed. Ideally, modeling only the robot is preferable as it enables the application of the simulation in the most general case. The more extensive the modeling of the environment, the greater the effort required for each new task. In the user study, the goal was to primarily indicate the goal positions using slabs without modeling the cubes.

7
Conclusion and future outlook

The results from the user study it is evident that visualization of PDDL plan to the user helps the user understand the intention of the AR robots and also the identify the failures in task execution.

The users were able to reply the specific actions to understand better how the AR robot executed the task. This helped them in identifying the failures in execution. The participants of the user study also indicated that had there not been any visualization of the plan and the ability to replay the actions, it was likely that they would not have identified the failures and intentions of the AR robot.

Visualizing the robot and its actions in AR domain is also an effective way to educating the user about the capabilities of the robot before using the real robot. Gaining knowledge of the robot capabilities before using the robots leads to safer and cost effective way of using the real robots.

The system developed in this thesis can be a platform to deploy and test new robot plan visualization solutions. More robot models can be Incorporated into the system that will enable the behavioral study of different robots using the same AR application.

Another prospect is to move away from usage of predefined planning problem and instead to generate a planning problem based on robot perception of its environment. Methods such as the one described in [34] presents ways to generate robot plans given a domain specification from observations. This will enable further studies of effects of robot plan visualization of complex tasks, in building trust of robots in humans.

Furthermore, the thesis uses MoveIT for trajectory calculation. However MoveIT is a generic trajectory calculation tool which could be replaced with the robots own trajectory calculation algorithm or other libraries specific to a robot model.

8

Ethical and sustainability aspects

Industry 4.0 is affluent with autonomous robots or Automated Guided Vehicle (AGVs). Though robots can operate autonomously, they still need to be monitored and managed by humans. Therefore, the collocation of humans and robots are prominent state in industry 4.0. In light of this, providing enough safety measures to humans is highly important. Currently, robotic errors are one of the factors of accidents in industrial environments. For the robots to collocate with humans, the robots need to be trusted and relevant and useful information needs to be provided to the humans near them. This information builds trust and in case of errors provides information to humans to be able to avert harmful accidents.

Efforts in building human trust in robots, such as by visualizing the robot intents(plan visualization) and failures, play an important part in increase adoptions of robot in human workflow.

Bibliography

[1] M. Walker, H. Hedayati, J. Lee, and D. Szafir, "Communicating robot motion intent with augmented reality," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 316–324. [Online]. Available: https://doi.org/10.1145/3171221.3171253

[2] P. David, Z. Bowen, M. Ke, and H. Björn, "Hair: Head-mounted ar intention recognition," *arXiv*, 2020. [Online]. Available: https://arxiv.org/pdf/2102.11162.pdf

[3] "Niryo robot." [Online]. Available: https://www.gotronic.fr/pj2-36480-mechanical-specifications-11-09-2018-2063.pdf

[4] "Ur3 robot." [Online]. Available: https://www.universal-robots.com/products/ur3-robot/

[5] "Hololens2." [Online]. Available: https://www.microsoft.com/en-us/d/hololens-2/91pnzzznzwcp?activetab=pivot:overviewtab

[6] Kuka, "Human-robot collaboration: 3 case studies," 2022. [Online]. Available: https://www.wevolver.com/article/humanrobot.collaboration.3.case.studies

[7] S. K. Kim, E. A. Kirchner, L. Schloßmüller, and F. Kirchner, "Errors in human-robot interactions and their effects on robot learning," *Frontiers in Robotics and AI*, vol. 7, 2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2020.558531

[8] Z. R. Khavas, "A review on trust in human-robot interaction," *arXiv*, 2021. [Online]. Available: https://arxiv.org/abs/2105.10045

[9] "Explainability + trust," 2021. [Online]. Available: https://pair.withgoogle.com/chapter/explainability-trust/

[10] "Task planning in robotics." [Online]. Available: https://roboticseabass.com/2022/07/19/task-planning-in-robotics/

[11] "Ai planning." [Online]. Available: https://researcher.watson.ibm.com/researcher/view_group.php?id=8432

[12] D. S. Nau, "Current trends in automated planning," *AI magazine*, vol. 28, no. 4, pp. 43–43, 2007.

[13] M. Diehl, C. Paxton, and K. Ramirez-Amaro, "Automated generation of robotic planning domains from observations," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6732–6738.

[14] P. Haslum, N. Lipovetzky, and D. Magazzeni, *An Introduction to the Planning Domain Definition Language*. Morgan Claypool Publishers, 2019.

[15] "Explainable ai – how humans can trust ai." [Online]. Available: https://www.ericsson.com/en/reports-and-papers/white-papers/explainable-ai--how-humans-can-trust-ai

[16] G. D. Cantareira, G. Canal, and R. Borgo, "Actor-focused interactive visualization for ai planning," *Proceedings of the International Conference on Automated Planning and Scheduling*, Jun. 2022. [Online]. Available: https://doi.org/10.1609/icaps.v32i1.19857

[17] H. Liu, Y. Zhang, W. Si, X. Xie, Y. Zhu, and S. Zhu, "Interactive robot knowledge patching using augmented reality," 05 2018.

[18] T. Williams, L. Hirshfield, N. Tran, T. Grant, and N. Woodward, "Using augmented reality to better study human-robot interaction," in *Virtual, Augmented and Mixed Reality. Design and Interaction: 12th International Conference, VAMR 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part I 22*. Springer, 2020.

[19] R. T. Chadalavada, H. Andreasson, M. Schindler, R. Palm, and A. J. Lilienthal, "Bi-directional navigation intent communication using spatial augmented reality and eye-tracking glasses for improved safety in human–robot interaction," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101830, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584518303351

[20] Y. T. Y. K. H. N. T. F. T. M. G. A. G. R. M. Y. A. T. Y. H. L. El Hafi, S. Isobe and T. Taniguchi, "System for augmented human–robot interaction through mixed reality and robot training by non-experts in customer service environments," *Advanced Robotics*, vol. 34, no. 3-4, pp. 157–172, 2020. [Online]. Available: https://doi.org/10.1080/01691864.2019.1694068

[21] G. Avalle, F. De Pace, C. Fornaro, F. Manuri, and A. Sanna, "An augmented reality system to support fault visualization in industrial robotic tasks," *Ieee Access*, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2940887

[22] "Niryo robot." [Online]. Available: https://niryo.com/docs/niryo-one/user-manual/complete-user-manual/

[23] "Ur3 gripper."

[24] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence*. [Online]. Available: https://www.jair.org/index.php/jair/article/view/10457

[25] "A*." [Online]. Available: https://en.wikipedia.org/wiki/A*_search_algorithm

[26] "Lmcut." [Online]. Available: https://ai.dmi.unibas.ch/papers/helmert-domshlak-ipc2011.pdf

[27] "articulation." [Online]. Available: https://docs.unity3d.com/Manual/articulations-section.html

[28] "Ros packages." [Online]. Available: http://wiki.ros.org/Packages

[29] "Ros service." [Online]. Available: http://wiki.ros.org/Services/

[30] "Fast downward repository." [Online]. Available: https://github.com/aibasel/downward/blob/main/README.mdl

[31] "moveit." [Online]. Available: https://github.com/ros-planning/moveit
[32] "Deploy." [Online]. Available: https://learn.microsoft.com/en-us/training/paths/beginner-hololens-2-tutorials/
[33] E. De Pellegrin, "Pdsim: Planning domain simulation with the unity game engine," in *Proceedings of the ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 2020.
[34] M. Diehl, C. Paxton, and K. Ramirez-Amaro, "Automated generation of robotic planning domains from observations," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6732–6738. [Online]. Available: https://doi.org/10.1109/IROS51168.2021.9636781

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY