CHALMERS

# Human-Robot Collaboration through Augmented Reality

## Visualising the TIAGo robot using Augmented Reality

Systems and Control 2023

Stas Akopau, Raafat Al-Zina, Oskar Andersson,
Philip Fredriksson, Albin Juopperi, Ida Wackerberg

DIVISION OF SYSTEMS AND CONTROL
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Bachelor's thesis 2023

# Human robot collaboration through Augmented Reality

EENX16-23-23


CHALMERS

Department of Electrical Engineering
EENX16-23-23
Chalmers University of Technology
Gothenburg, Sweden 2023

Human-Robot Collaboration through Augmented Reality

STAS AKOPAU
RAAFAT AL-ZINA
OSKAR ANDERSSON
PHILIP FREDRIKSSON
ALBIN JUOPPERI
IDA WACKERBERG

© STAS AKOPAU, RAAFAT AL-ZINA, OSKAR ANDERSSON,
PHILIP FREDRIKSSON, ALBIN JUOPPERI, IDA WACKERBERG, 2023.

Supervisor: Karinne Ramirez-Amaro, Division of Systems and Control
Examiner: Emmanuel Dean, Division of Systems and Control

Bachelor's Thesis 2023
Division of Systems and Control

EENX16-23-23
Chalmers University of Technology
SE-412 96 Gothenburg

Cover: Our picture of the AR application.

Typeset in LaTeX
Gothenburg, Sweden 2023

# Abstract

This thesis describes the design and implementation of controlling an augmented reality, (AR), TIAGo robot. The project's objectives were to explore how an AR application can be developed for controlling an AR TIAGo robot in real-time, and if a potential collision could be avoided before initiating movement of the actual robot. To conduct this research Unity was used to replicate the robot in an AR environment while ROS was used to control it. The robot could be implemented in AR and displayed using a android tablet. The TIAGo robot could move to specific locations and perform pre-defined movements using ROS. No testing was conducted on the real TIAGo robot resulting in the movement differences between the AR robot and the real one being unknown.

Keywords: AR, ROS, Unity, TIAGo, Vuforia

# Acknowledgements

During these last months, we have created a project we are proud to present. It has been a fun and challenging experience. There are however some people this would have not been possible without.

**Karinne Ramirez-Amaro** has been our supervisor during the project and has helped us with anything from how to talk in front of a group to the next step in the project. **Jesper Fagerberg** has created the VR_Chalmers git which our project is based upon. When Karinne was away **Maximilian Diehl** took her place as our supervisor, leading our weekly meetings. Lastly, **Chalmers** made our project finances not stand in the way of our end product giving us the tools needed.

Gothenburg, May 2023

# Contents

# List of Figures

# List of Tables

# List of Acronyms

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

| | |
|---|---|
| APK | Android Package |
| AR | Augmented reality |
| CAD | Computer-aided design |
| Cobot | Collaborative Robot |
| IMU | Inertial measurement unit |
| LAN | Local area network |
| ROS | Robot Operating System |
| TCP | Transmission Control Protocol |
| VM | Virtual machine |
| VR | Virtual reality |

# 1
# Introduction

Augmented reality, AR, is used as a complement to reality. When using AR digital information is projected on the real world through a lens or a camera. Examples of commercial applications on AR are Pokémon Go [1] and Facebook's messenger app which uses AR to modify the user's appearance.

In [2] the use of AR in the industry is examined. AR has proved to be particularly useful in maintenance and repair tasks in the manufacturing industry. By providing step-by-step instructions and visual aids to identify the issues in real-time and alert the worker on defects or deviations. AR is also used for training purposes enabling workers to practice in a virtual environment before doing the task in the real world. AR technology has the potential to increase efficiency, reduce errors and improve training.

During this project the use of AR as a tool for visualisation is developed and in this chapter, the background, purpose and limitations of the project are introduced.

## 1.1 Background

This project was undertaken within the Department of Electrical Engineering and focuses on developing an AR application for the TIAGo robot. The project builds on the institution's previous work on the robot. In the past a VR environment has been developed in Unity and this will serve as a foundation for the development of the AR application.

## 1.2 Purpose and goals

This project will involve an examination of the use of AR technology in collaboration with the TIAGo robot. The main focus will be on utilising AR to display movement sequences.

The technical objectives are to visualize the TIAGo robot at different locations performing different tasks using AR. By visualising the robot in AR the movements of the real robot can be anticipated. The goal for the finished application is that the robot performs movements including both the wheels and arm of the robot.

## 1.3 Related work

When researching similar work to what this project will contain following projects were found:

- *Human-Robot Collaboration through Augmented Reality with the HoloLens2 [3]*
- *Interaktiva Fabriker i Augmented Reality Med hjälp av Microsoft Hololens [4]*
- *Sammanstrålning av världskoordinater hos en augmented reality-redo beräkningsplattform och ett AGV-system [5]*

The aforementioned works, the master thesis *Human-Robot Collaboration through Augmented Reality with the HoloLens2* and the bachelor theses in *Interaktiva Fabriker i Augmented Reality Med hjälp av Microsoft Hololens* and *Sammanstrålning av världskoordinater hos en augmented reality-redo beräkningsplattform och ett AGV-system*, have been selected due to their similarities with the application and work to be carried out in this project [3], [4], [5].

In *Human-Robot Collaboration through Augmented Reality with the HoloLens2* the movement of an industrial robot with a gripping tool in AR was developed [3]. The project used ROS to implement robotics control and create the trajectory planner. Unity was used to simulate the trajectory of the robot. The Vuforia package was then used to create the simulation in AR. The code was then deployed to the HoloLens 2 the simulation worked and the robot was able to be simulated and move around in AR. In the project, the robot was not however able to pick up the AR items. This problem is explained to be caused by a coordinate failure between the Unity environment and the HoloLens 2. The AR interface used a reload button and a publish button. The reload button was able to re-run the simulation helping with debugging while the publish button was used to publish the trajectory. Other problems were that it was only able to simulate the robot using the standard Vuforia image. Vuforia's Model Target is mentioned in the recommendation for future work and this is taken into account in this project.

*Interaktiva Fabriker i Augmented Reality Med hjälp av Microsoft Hololens* by Calle Halvarson is a study where AR was used to create an interactive factory [4]. The commonality between the projects is the development tools used for visualising the factory in AR. The visualisation tool Unity was used due to the fact that it is the only platform supporting the development of an application for Hololens. In the project developed this is taken into account when choosing software.

AR develops with time and by connecting machines to the internet the possibility to get real-time data presents. In [4] the operator wearing the Hololens can see information about the machines but not in real-time and it can not control the different machines. Compared to 2017 the AR technologies have improved and will be considered with a new perspective.

The thesis *Sammanstrålning av världskoordinater hos en augmented reality-redo beräkningsplattform och ett AGV-system* uses Unity to develop an AR application for an AGV system [5]. This application is deployed using an Android device. The

device used for the project was a Yellowstone 7-inch tablet which supported Google Tango. One of the conclusions of the project was that by using Unity the final application can be used on any unit supporting Google Tango. This was seen as one of the strengths in this project. The application being developed in this project will also use a tablet but Google Tango has been replaced with ARCore.

## 1.4 Research questions

Based on the purpose and background of this project the following research questions are answered throughout the project:

- What is the procedure for developing an AR application for the TIAGo robot?
- Can the developed AR application navigate to a target through ROS in real-time?
- When navigating to the target in AR can a potential collision for the real robot be avoided?

## 1.5 Delimitation

To clarify the scope of the project this delimitation section describes what is not included and the boundaries of the final application and research.

The application will be specific to the TIAGo robot and will not apply to other types of robots. There will not be any feedback from the environment or any sensors to the application and the robot will therefore only perform a collection of pre-defined paths. The 3D models for the robot are provided and therefore no creation of new 3D models for the robot will be made.

Finally, it is important to note that this research is limited to the development of the AR application itself, and does not include any evaluations or analyses of the robot's performance or capabilities. Any conclusions drawn from the project are therefore limited to the development and implementation of the AR application only.

# 2
# Technical background

This chapter describes all the software and hardware used for the purpose of the project and gives a brief explanation of why this is used in the project.

## 2.1 The TIAGo robot

The TIAGo is a mobile manipulator cobot that is designed to perform a wide range of tasks in various settings. With its robotic arm and wheels, the robot is capable of moving and handling objects. The robot uses the base and wheels to move around with precision and efficiency, making it highly adaptable and versatile [6].

Equipped with a range of sensors and cameras, the TIAGo is able to navigate autonomously in real-world environments[7]. These sensors include a laser range finder, stereo vision camera, and inertial measurement unit (IMU), providing accurate information on the robot's position and orientation. The cameras on the robot can also capture images and videos of the environment, making it useful for analyzing and monitoring the surroundings.

To control the TIAGo, it uses Robot Operating System (ROS), a flexible and powerful framework of process for robot programming and control. The use of ROS enables easy programming and control, making it possible for developers and researchers to create custom applications and behaviours for the robot [8].

### 2.1.1 The structure of ROS

ROS is a set of software libraries and tools that are used to assist users in building robot applications. ROS is an open source, meta-operating system for robots, providing all the necessary components, tools and interfaces used to control robots. Due to the module structure, the process of communication between different parts of the robot is simplified to the programming of separate modules, called nodes. These nodes, which are specified for an individual part of the robot, send messages to one another [9].

ROS provides users with a workspace building tool, a package called catkin. This building tool allows users to combine created nodes into their own package, and thus create a robot application [10].

Nodes communicate by sending messages over a specific topic. When sending these messages the nodes exchange data. One way to send the data is by using a subscriber and publisher. The main idea of these is presented below:

**Publisher:** A part of a ROS node is a publisher. A publisher is a "talker" that sends a message to a specific topic [11].

**Subscriber:** The subscriber is the "listener" that receives a message from a topic [11].

**Message:** Messages is a description language for describing the data values that ROS nodes publish. The message contains the data that is sent between the nodes. There are different types of messages each having defined structures [12].

**Topic:** A topic is a databus for exchanging data between nodes. A topic is necessary for example when a publisher needs to send a message that has two or more destination subscribers [13].

**RQT Graph:** This is a feature provided by ROS that allows the user to get a map of the nodes in configuration, both active and inactive [14]. This graph displays how nodes are connected and how they communicate. The graph shows the allowed input to the node and the name of the topic that provides it.

In a ROS-based robotic system, the robot can be controlled by programming and setting up nodes. The information that is exchanged between the nodes can be used when running a script. The information can for example be a position that the robot should go to.

### 2.1.2 Action client

Action client is a part of ROS actionlib that allows nodes to send requests and receive feedback as seen in Fig. 2.1 [15]. This makes it possible to write a node that constantly gets information regarding a requested goal. When a request is made, the robot will finish this request before taking on any new requests.

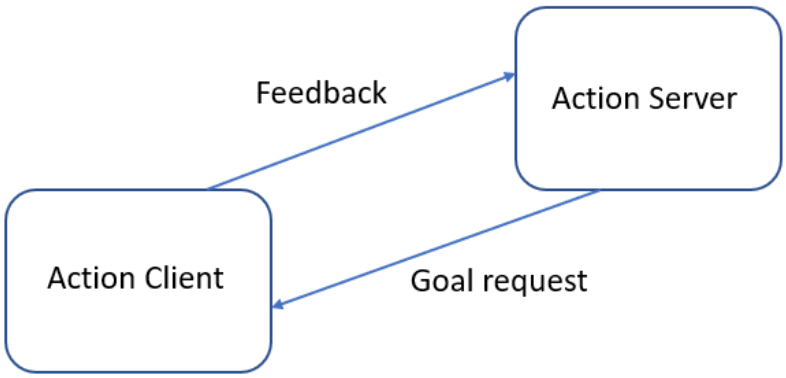



**Figure 2.1:** Communication of two action nodes.

#### 2.1.2.1 Visualisation in Gazebo

Gazebo and ROS are often used together for robot testing and development, it allows developers to simulate robots and test their performance in a simulated environment. Gazebo integrates with ROS by providing plugins that enable ROS nodes to publish and subscribe to data from the simulated robot's sensors and actuators. The Gazebo environment is used to simulate the TIAGo robot and test its movements, while ROS controls the robot's actions and receive sensor data from the simulation.

#### 2.1.2.2 Usage of Docker

Docker enables an application and all its dependencies to be compressed to an image allowing it to easily be deployed on another computer [16]. Docker images can be customized to include specific dependencies, configurations, and software packages, allowing developers to create tailored environments for their applications. Docker can also be used as a lightweight virtual machine, VM. The main purpose in this project is to get access to prebuilt configuration of the TIAGo robot and dependencies in ROS.

### 2.1.3 Introduction to URDF

URDF, Unified Robot Description Format, is a file that describes the physical properties and kinematics of a robot [17]. This enables developers to simulate and visualize the robot's behaviour in a simulated environment. It is standardized way of sharing robot models and configurations among the robotics community. In this project the URDF is provided by PAL robotics on the TIAGo model and is used when simulating the robot in Unity and Gazebo [18].

## 2.2 Hardware and software to visualize AR

In the following section, the software used to create the AR environment will be explained.

### 2.2.1 Unity game engine

Unity is a game engine, which uses scenes for development. These scenes typically involve a single light source and a camera that can be adapted to fit the desired scenario. In the context of augmented reality, the camera can be modified to support AR functionality. To program the behaviour of the elements within Unity's scenes, developers commonly use C# [19].

In Unity it is possible to add a multitude of packages that can expand or simplify the process of environment building, model design and texture [20]. Apps made with Unity are compatible with different operating systems depending on the build settings of the project. If the goal is to create a mobile app for Android, the build settings need to be set for Android, and similarly if it was for iPhone. However, PC, Mac and Linux all fall under the same build [21].

A crucial aspect of AR is determining when the AR model should become visible to the user. The timing of this presentation is dependent on the specific use case for the AR application. A common method of testing AR functionality involves the use of an image target, which is an image that, when observed by the device's camera, triggers the appearance of the AR model [22].

### 2.2.2 Vuforia to use image targets

Vuforia is an AR software development kit that serves primarily as a tool for generating Image Targets, which can then be interpreted within the Unity game engine [23]. Additionally, Vuforia offers the ability to develop what is known as a Model Target. In contrast to an image target, the AR model is triggered by a real-world object, with the Vuforia Model Target Generator playing a key role in this process. By leveraging a CAD model, the generator creates a model target that can be incorporated into the Unity environment [24].

### 2.2.3 Tablet to display AR

To view and interact with AR, specific hardware is required, such as a tablet. However, running an AR simulation on a tablet requires support for ARCore. While many smartphones do support ARCore, their small screen size may not be optimal for an immersive AR experience.

ARCore, developed by Google, is a platform that enables devices to sense and interact with their surrounding environment. Using APIs, ARCore allows devices to estimate lighting conditions, determine the sizes of surfaces and objects, and track their own speed and motion relative to their surroundings using the camera and built-in inertial sensors [25].

In this project, a Samsung Galaxy Tab S6 Lite Wi-Fi is used. The Samsung Galaxy Tab S6 Lite Wi-Fi is a mid-range Android tablet that was released in 2020 using Android version 13. It features a 10.4-inch display with a resolution of 2000 x 1200

pixels, which provides a good viewing experience for AR content. The tablet is powered by an octa-core Exynos 9611 processor and 4GB of RAM, which ensures that it can handle running ARCore apps smoothly [26].

One of the key features of the Galaxy Tab S6 Lite Wi-Fi is the 8 MP main camera, which can be used to capture images and videos. This camera is essential for ARCore to sense and track the surrounding environment accurately.

### 2.2.4 HoloLens 2 to display AR

HoloLens 2 is a headset possessing transparent lenses onto which images and virtual objects can be displayed [27]. Hololens 2 is developed by Microsoft and is supported by Unity.

## 2.3 Establishing communication between software

To be able to communicate between ROS and the AR application a connection needs to be established this is done using TCP Connection via LAN.

### 2.3.1 TCP Connection

Transmission Control Protocol, TCP, is a widely used protocol in computer networking that provides reliable and ordered delivery of data between applications [28]. A TCP connection is a virtual connection that allows two devices to communicate with each other over a network. To connect the two computers a three-way handshake is made between them. Firstly a packet with the bit SYN is set to 1 then asks the second computer to synchronize. The second computer is then responding by sending a packet back with an ACK bit set to 1, meaning acknowledging the synchronization between them. Lastly, the first computer also responds by sending a packet back with an ACK bit set to one. After that packet is received the computers are done with the handshake and the connection is established [28].

TCP breaks the data into smaller parts called packets and assigns them a unique number to keep track of their order. It makes sure that these packets reach the intended destination device in the correct order. TCP uses error detection tools to ensure that the data is delivered without any mistakes. To achieve this level of reliability TCP requires more processing and extra control packets which adds to the overall time and resources required [29].

### 2.3.2 LAN

A Local Area Network, LAN, is a type of computer network that connects devices within a small geographical area, typically within a building or campus [30]. It is designed to allow devices such as computers, printers, servers, and other network-enabled devices to communicate and share resources with each other.

Wireless LANs use radio waves to connect devices without the need for physical cables [31]. This allows for more flexibility in device placement and easier setup, as devices can be added or removed from the network without the need for physical connections [30].

# 3
# Methods and implementation

The project was structured into three distinct phases: preparatory work, a development phase, and an integration phase. As illustrated in Fig. 3.1, each of these phases is accompanied by a specific set of tasks and objectives, which will be elaborated upon in the following chapter.

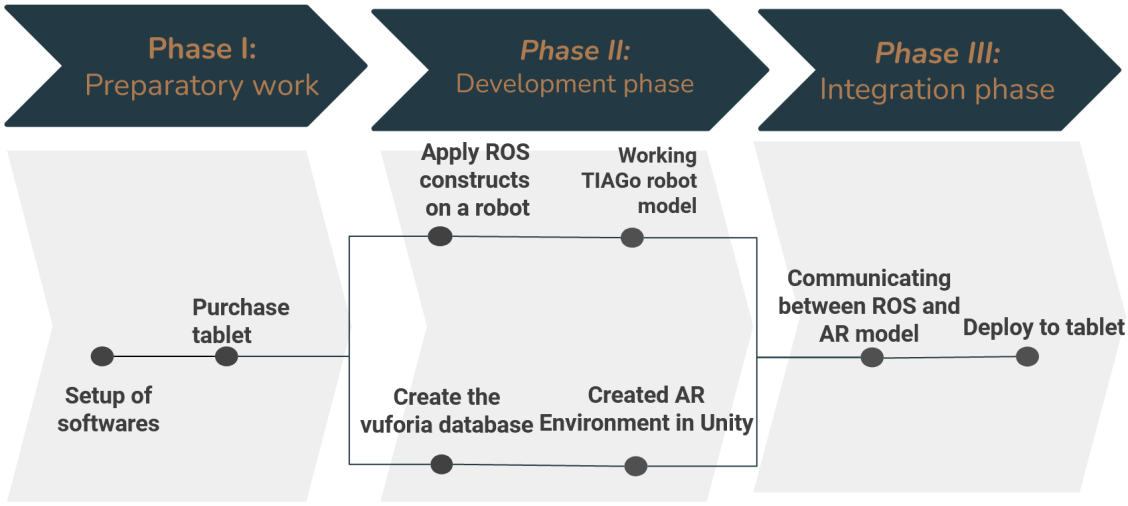



**Figure 3.1:** Visualization of method plan with three phases: Preparatory work, Development phase, Integration phase.

## 3.1 Preparatory work - Phase I

In the project, different software was used. These software programs were installed and initiated to get the right built-in packages. Simultaneously as the software was set up the tablet was purchased.

### 3.1.1 Installation and setup of software

To develop the application the main software that was used were Unity and ROS. The software was installed on two different computers as following:

**ROS:** A native version of Ubuntu 20.04 was installed onto a computer that would run the docker container and ROS server. Once the Ubuntu setup was done, Docker was downloaded together with ROS. To know which ROS package would be needed to move the robot in different ways the guides and tutorials in [8] were used.

**Unity:** Unity hub was installed and for the application developed in this project the 2020.3.39f1 version was used. This was installed on a computer with Windows 11.

### 3.1.2 Requirements for tablet

During this period, research was conducted to determine which tablet was suitable for running Unity and AR applications. The selection criteria for the tablet were that it should be an Android tablet, have a high-quality camera, and support ARCore. To ensure that the tablet met all the requirements and the limitations based on the application, a requirement specification was created, see Table 3.1. The different requirements were based on discussions with the project supervisor and Unity's requirements for Android. The requirements also took into account the users comfort, hence a larger display size was favoured.

**Table 3.1:** Requirement specification for tablet where R stands for requirements and D for desirable

| **Requirement specification of tablet** | | **Made:** | | **Modified:** |
|---|---|---|---|---|
| **EENX-16-23-23** | **Chalmers** ***Criteria*** | **2023-01-31** ***Target value*** | ***R/D*** | **2023-02-10** ***Interseted party*** |
| **Functions** | Camera | Yes | R | Application |
| | Android | Yes | R | Application |
| **Performance** | Storage | 32 GB | R | Application |
| | RAM | 4GB | R | Application |
| | Camera | 8 MP | R | Application |
| | ARCore | Yes | R | Application |
| | Depth API | Yes | D1 | Application |
| | CPU | 2GHz | R | Application |
| | Battery | 7000mAh | R | Application |
| **Dimensions** | Screensize | 10" | D2 | User |
| **Purchasing** | Price | max 5000 | D3 | Project group |
| | Supplier | Dustin | R | Project group |

### 3.1.3 Challenges and discussion for Phase I

During the preparatory work different issues came up and some of them will be brought to light in this section.

Initially, the group intended to run ROS and Docker on a Windows computer by using a VM with Docker. Although the group managed to make it work, it was not recommended due to the sensitivity to specific versions of both the Docker image and Python. Therefore, the final product uses a native Ubuntu computer. However, it should be possible to complete this project on a Windows computer with the correct setup.

At the beginning of the project, the plan was to purchase a pair of tactgloves, and the initial budget included this expense [33]. However, these gloves were not available from a European supplier at the time, and thus could not be purchased. If acquired, these gloves would have made the application more interactive by allowing users to interact with the robot in the AR environment.

In this phase it was determined which tablet to purchase and this was based on the different factors mentioned above. The reason that a tablet was purchased was mainly due to the screen size. The final application could be deployed on a smartphone or on a computer with a camera. A computer could be functional if the application were to be displayed in a static environment but since the robot moves around, using a computer would make it hard to see and follow the robot at the same time.

A smartphone could be a good option because of the accessibility but the small screen can become an inconvenience. These were the reasons why a tablet was used but the final application can be deployed on an Android phone or computer.

## 3.2 Development phase - Phase II

The development phase of the project consisted of two different tasks. The first task was to control the robot using ROS, while the second task was the creation of an AR environment in Unity. The implementation method of both tasks is presented in the respective sections below.

### 3.2.1 ROS to control the robot

To control the robot two scripts were created with the help of an rqt-graph and an action client.

#### 3.2.1.1 Analyze Rqt-graph

During the project, the rqt-graph was used to analyze the node responsible for controlling the robot, see Fig. 3.2 for a part of the rqt-graph. For the complete rqt-graph see appendix A. Upon inspection, it was found that the node controlling the movements of the base required information about velocity and angle of rotation to operate. The rqt-graph also provided valuable information about the node's input requirements, indicating that it expected a geometry message of Twist type [34].

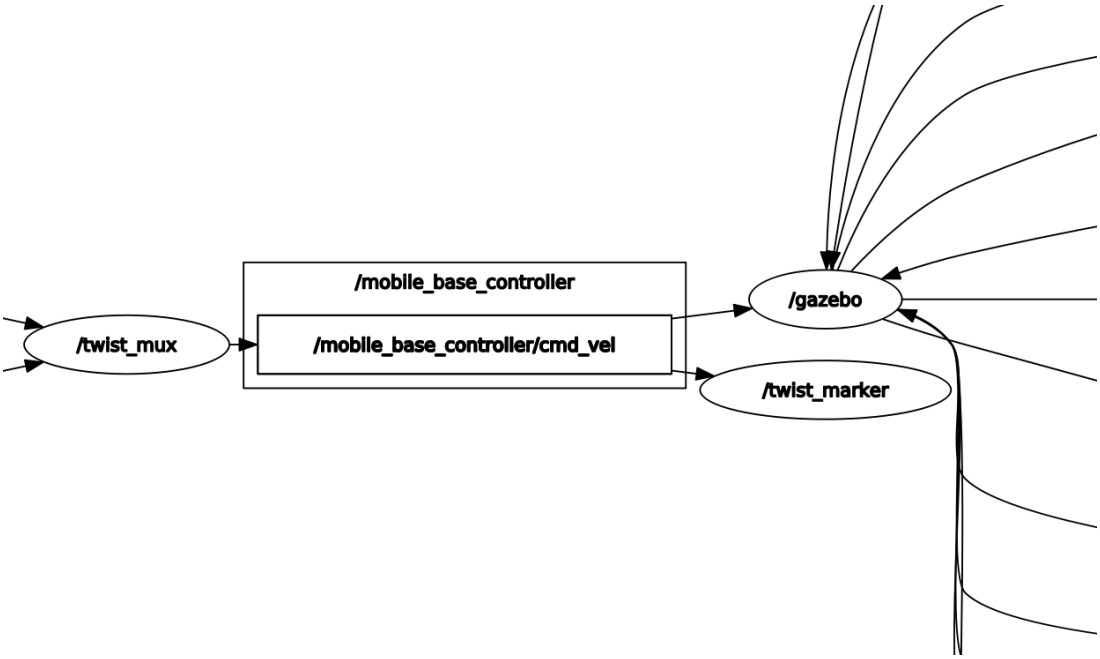



**Figure 3.2:** Nodes presented in the rqt-graph.

The same procedure was followed to control the robot arm, inspecting the relevant node in the rqt-graph, identifying its input requirements, and noting the topic needed to send the necessary message.

#### 3.2.1.2 Developing scripts

The information given by the rqt-graph indicated which nodes communicated and on what topic, when developing the scripts this was used. The scripts communicate with different nodes to control the robot by creating it's own node.

In the script the node was initialized and a client was started. The client was created and a handshake between the nodes was done. To move the TIAGo to the desired position a goal was sent. The goal included what parts of the robot that should be moved, to what position and with what velocity. The goal was sent as a FollowJointTrajectory which is a type of Control message and was formed according to [35].

### 3.2.2 Unity

The following section describes how Unity was used with the support of the Vuforia engine to create an AR environment for the TIAGo robot. The environment included two image targets corresponding to two separate AR objects, the robot and the charging station.

#### 3.2.2.1 Vuforia Setup

To add the Vuforia library to Unity it needed to be downloaded from [36]. On the Vuforia website, a database including the image target was created. The database

was downloaded and added to Unity. The steps can be seen in appendix B.

#### 3.2.2.2 Unity Scene Setup

From the previous VR environment, the provided Unity scene could be modified and by using the preexisting URDF model the TIAGo could be presented. After initializing the scene Vuforia was imported and the previously acquired license was used. With the addition of Vuforia, an AR camera was added to the scene.

Following the addition of the AR camera, an image target object was added to the scene. Afterwards, the previously downloaded Vuforia database was imported into Unity, making the target functional and act as a marker. The properties for the marker were then accessed through the database. To achieve the goal of making the robot appear as a projection on the marker, the robot model object was placed inside the image target, making it the child of the target. In order to set the marker as the origin, the AR camera settings were adjusted.

The second image target was created the same way. But instead of the robot appearing a model representing a charging station appears. To make multiple AR models appear at the same time the max AR simulation was increased under the "vuforia engine configuration". The steps can be seen in appendix C.

### 3.2.3 Challenges and discussion for Phase II

In this section, the challenges that were met during phase II in the development of ROS and Unity are discussed.

#### 3.2.3.1 The challenges with ROS

In the beginning, the plan was to develop a script that communicated with the publishers and the subscribers but a different approach was taken, namely using action client instead. The benefit of this approach is that the client only performed once with all the feedback on performance. This allows to see the exact destination to be reached.

When the action client for the base was developed an error in the communication between the nodes and Gazebo appeared. Due to this, the ROS could not move the base in the AR model. However, this was simulated by ROS requesting and parsing the information from and to Unity. In Unity, the movement of the TIAGo is then controlled by a C# script.

#### 3.2.3.2 Challenges with Unity

The first step in creating the AR environment was the decision on what kind of target to use, Model or Image. A Model Target would give a more immersive experience by not needing a printed image for deploying the AR model. Since we had the URDF model of the robot the CAD model could be exported from Unity using the

"FBX Exporter" package. This CAD model could then be imported into Vuforia's "Model target generator". Creating the model target which was later imported to Unity. Upon testing, the AR model appeared when the camera was panned over the real-life robot. However, an issue arose where the AR model did not appear at the desired location. After evaluating the time required to resolve this issue, the decision was made to switch to an Image Target instead.

Though it was chosen, the Image Target method was not without its own issues. When utilizing this method a number of issues were encountered. Similarly to the Model Target issue, the robot model did not appear at the correct location. Additionally, there was an issue, connected to the first one, where the AR model does not stay at its location and instead moves with the camera, which is undesirable for the end result. Both these issues were solved by clicking the AR camera in the hierarchy and setting the "World Center mode" to "SPECIFIC_TARGET" and dragging the image target to "World Center". Further issues involve the model falling down towards negative infinity, which was fixed by selecting "Tiago" and disabling gravity.

For the creation of the moving AR robots goal, two options were considered. The first uses an AruCo to transmit its coordinates to ROS, while the second use a second image target. Given the previously mentioned "World Center mode" and "SPECIFIC_TARGET," the implementation of the second image target proved to be a more feasible and straightforward solution. Since the first Image Target always would be deployed at the origin. The second one will have coordinates relative to this since all desired is movement from the first to the second. Only the coordinates from the second image target would need to be sent to the ROS.

## 3.3 Integration phase - Phase III

In the integration phase, the connection between the ROS program and the AR environment was established. The robot in AR was controlled by ROS using serialized messages. Unity provides tutorials connecting Unity and ROS on Git [37]. To understand how the connection should be set up these were followed and later on the methods were applied to this project.

### 3.3.1 Establishing connection between Unity and ROS

After the unity project and the ROS script have been initialized, the package ROS TCP Endpoint was used to connect the software. Using the package ROS TCP Endpoint the ROS environment can start a TCP server which can connect to Unity. When setting up the server the ROS-host defines which port is available for the Unity device to connect to. The host also connects to the IP-address of the Unity device. For this to work the device running Unity and the host needs to be on the same LAN. For the Unity environment the package ROS TCP Connector was used and connected to the ROS-host computers IP-address and specified port[38]. In Fig. 3.3 a visualization of the connection between the software can be seen.

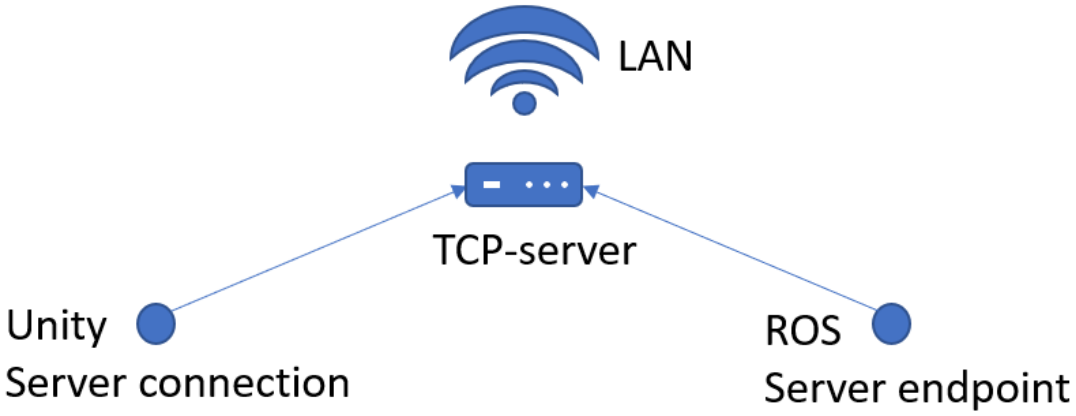



**Figure 3.3:** TCP-server between Unity and ROS.

### 3.3.2 Communicating between Unity and ROS

After establishing a connection between Unity and ROS, data can be exchanged between the two systems through ROS topics, messages, and services. By identifying the messages sent by the ROS nodes to control the robot, the messages can be translated into C# code to enable the AR application to interpret the data from ROS. Subsequently, the information received by Unity from ROS can be parsed onto scripts, updating the state of objects in the Unity scene.

#### 3.3.2.1 How Unity-ROS connection works

To move the robot in AR, C# and Python scripts were written and a ROS service script was created. This service script enabled the ROS-side to request information about the TIAGo robot and target position. By leveraging this information, ROS moves the robot towards the target. The information was relayed back to Unity, where the Unity engine executed the corresponding actions. In Fig. 3.4 communication between the computer running ROS and Unity scene is seen.

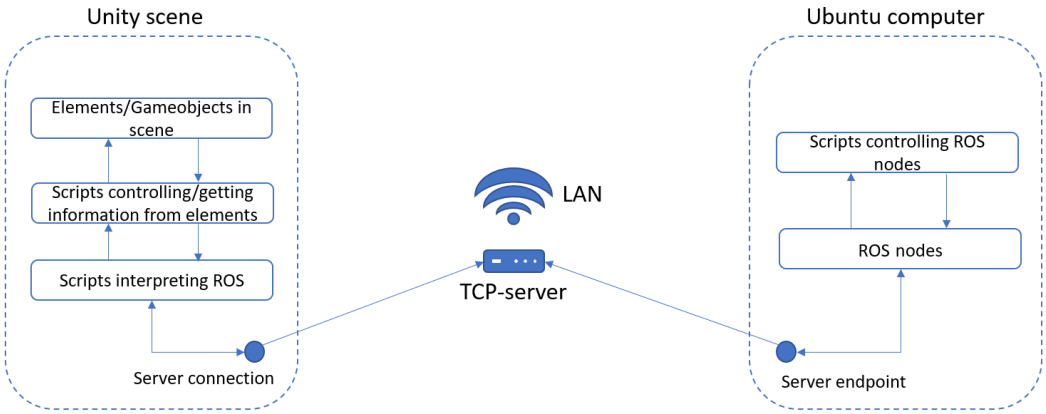



**Figure 3.4:** Visualization of communication between Unity-ROS.

#### 3.3.2.2 How Unity-ROS-Gazebo connection works

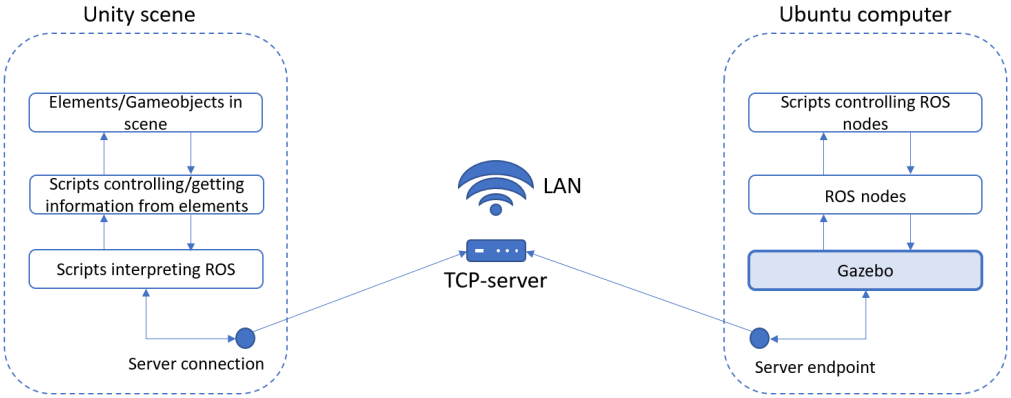



**Figure 3.5:** Visualization of communication between Unity-ROS-Gazebo.

When controlling a robot using Gazebo, an additional layer of communication is added to the process as seen in Fig. 3.5, which is not present in the ROS-Unity communication flow. In this case, instead of using a ROS service to obtain the robot's location from Unity, ROS can directly fetch the position from Gazebo since both Gazebo and ROS are interconnected by nodes. A script is then sent to Gazebo and the robot will perform the instruction.

To visualize the robot's movement in AR, Unity listens to the data published by Gazebo and simulates the same actions as the virtual TIAGo robot within Gazebo. This allows for a smooth and accurate representation of the robot's movement in the AR environment.

### 3.3.3 Deploying on tablet

To deploy the developed project to the target tablet, the APK was exported from Unity and downloaded onto the device. This was achieved by downloading the "Android Build Support" module and configuring the build settings to Android. Following this, the APK file was created, enabling it to be transferred to the tablet and executed, successfully launching the application [39].

### 3.3.4 Challenges and discussion for Phase III

During phase III of the project, several problems were encountered. One of the most pressing issues was the presence of an encrypted LAN, which obstructed communication through the WiFi on Chalmers. As a result, the group had to set up their own LAN through 4G instead. In addition, there were multiple ways to specify the IP address that the Unity device should listen to. This required the IP address to be defined not only in the tab for the robotics system but also in a Unity script that superseded the manually set IP address. To enable the computer running ROS to listen to all devices connected to the server, it was best to set the IP address to listen to 0.0.0.0 instead of specifying the address of the Unity device.

To control the robot's movements via ROS, the group needed to initiate nodes through Gazebo, which also facilitated communication. The Unity environment then listened to the nodes initiated by Gazebo, allowing the robot to run in both Unity and Gazebo simultaneously. Although there are alternative methods based on the package of the initiated nodes, the group ultimately chose to continue with the Unity-Gazebo approach due to complications in creating new packages, such as the risk of missing dependencies.

# 4
# Results

In this section, the results of the project will be presented. First showing the created ROS scripts, the robot's deployment in AR and lastly, the final product where the connection is established and the robot moves in AR.

## 4.1 Phase I Results

The result of this phase involved the purchase of an Android tablet. Using ARCore and Unity's demands for Android, along with the requirement specification in Table 3.1, three possible tablets were presented [25], [32]. The presented tablets were: Lenovo Yoga Tab 11, Lenovo Tab P11 Plus and the Samsung Galaxy S6 Lite Wi-Fi [40], [41], [26]. The Samsung tablet was identified as the most suitable device for the project's needs since all requirements were met.

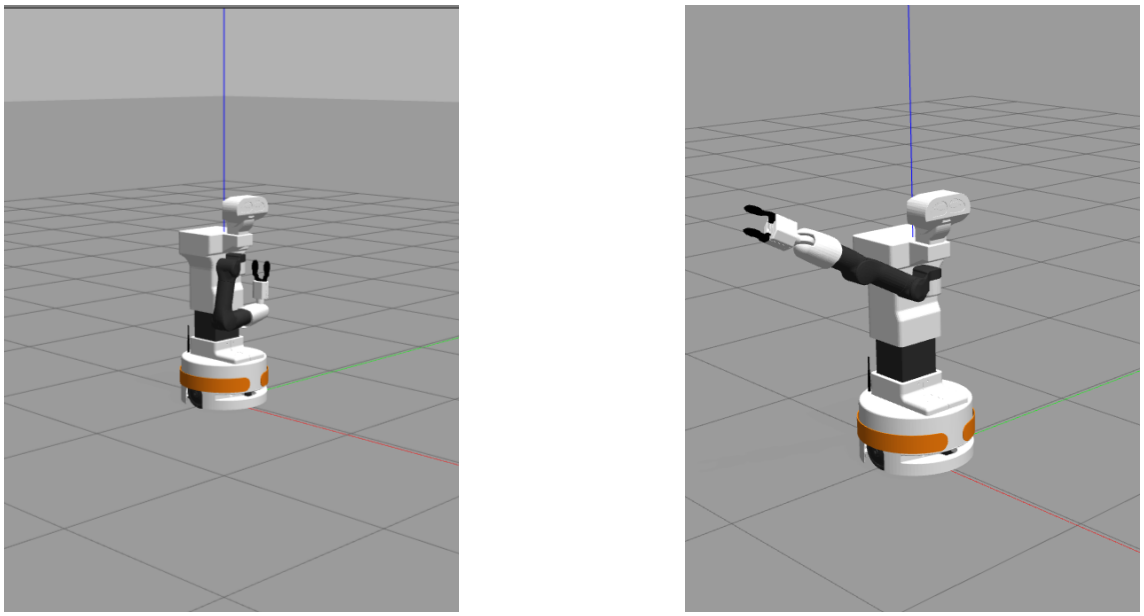
## 4.2 Phase II Results

During phase II the last step before integration was taken. These results can be seen in the following sections.

### 4.2.1 The created ROS scripts

During phase II ROS implemented two scripts: action client for the arm and unity object detection service.

The action client for arm control allows the user to send a goal to the TIAGo robot in Gazebo. The joints of the TIAGo is moved to the coordinates of the specified goal. When using Gazebo the robot will refuse impossible movements. In Fig. 4.1b the script controlling the arm is showed.



**(a)** Tiago in start position. **(b)** Movement performed.

**Figure 4.1:** Arm client script performed in a Gazebo Simulation.

The second script that was implemented fetches the position of an object in Unity and then sends the information back. A Unity script then uses this position to move the robot to the target position.

### 4.2.2 The finished AR product

Two image targets were created. The first one deploys the AR Robot while the second one was the charging station. Both can be observed in Fig. 4.2 deployed at the same time. The image target utilized for the AR robot will always establish its coordinates to origo.

**Figure 4.2:** The robot and charging station visualised in AR at Chalmers.

## 4.3 Phase III Results

In phase III the TCP Server and Gazebo simulation were set up to communicate to the Unity device, the Unity computer converted the scene to an Android application and the application was transferred to the Android tablet. The tablet then starts the application and connects to the ROS-computer's IP and port. Once this was done the AR environment could send information to ROS and the TIAGo robot could then be controlled.

### 4.3.1 The TIAGo moving in AR

The AR model of the robot can move by using manually controlled inputs and scripts when using Gazebo as an additional layer of communication. When the TIAGo is deployed at the image target it can either run a script controlling the arm or move by pressing the arrow keys. See Fig. 4.3 where the AR robot moves simultaneously with the Gazebo simulation.

**Figure 4.3:** The TIAGo robot moving in the Gazebo environment and in AR simultaneously. The robot is moving forward and rotating.

### 4.3.2 The TIAGo moving towards other objects in AR

When multiple objects are presented, the Unity scene can on request from ROS, send information about the position of an object. In Fig. 4.4, the robot moves towards the charging station, represented by the black cylinder. This is done by using the requested and received coordinates from the ROS script. A C# script will then move the robot towards the coordinates where the charging station is located.

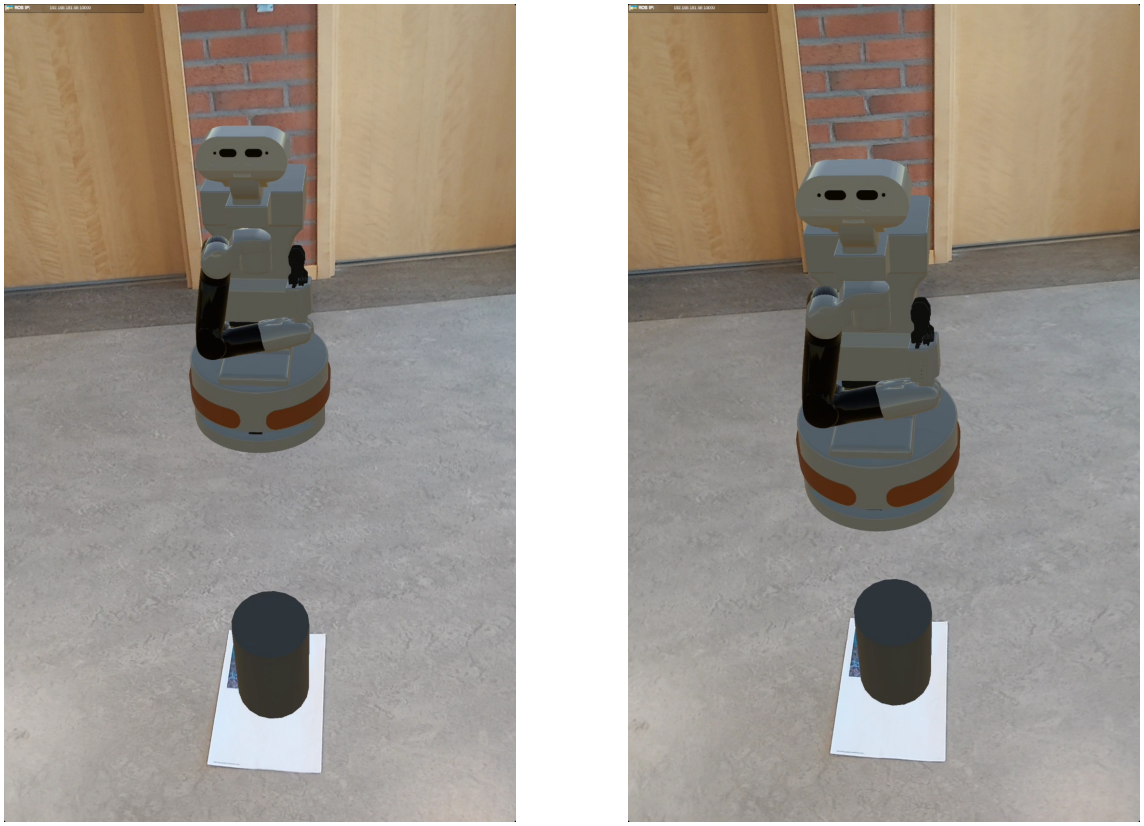


**Figure 4.4:** The TIAGo robot moving towards the charging station, i.e. the black cylinder.

## 4.4 Result discussion

In the following section, the problems and opportunities of the results will be discussed.

### 4.4.1 The AR application on different devices

The final application could be deployed on any Android device supporting ARCore but in this project, it has mainly been used on the purchased tablet. See appendix D when it is used on a OnePlus 8. The problem that could appear is if the ratio of the screen does not match present ratio of the Unity scene. If the application would have been desired to run on an IOS device this would have been possible if the steps in section 3.3.3 were changed from "Android Build Support" to the corresponding IOS package. Using an IOS device compared to an Android device would not give an advantage, due to the similarity in tablets screen size and design. But by changing the application to support HoloLens 2 the need for a handheld tablet could be removed and that could improve the user experience.

### 4.4.2 Differences between AR robot and real TIAGo robot

The TIAGo robot moves in the AR application however there has not been any comparison between how the real robot moves when sending the same instructions to both robots. This could potentially be a problem since no calibration or testing has been done according to section 1.5. The script controlling the arm could be tested on the TIAGo robot without modification. The possibility to test the same script on the real robot as the robot in AR was taken into account when choosing ROS to control the robot in AR. However since there was an error in moving the base of the robot, see section 3.2.3.1, the base movements were controlled by both Unity and ROS. Unlike the plan, ROS only parses the information to Unity instead of controlling the robot. This should be changed to get a more accurate result but in this project, the use of a C# script gave the possibility to visualise the robot's movements.

### 4.4.3 Evaluating the AR application

As seen in the results, visualizing the TIAGo robot and its' movements were successful. The main reason for developing this application was to determine if it was possible to avoid a potential collision when using the real robot. Judging by the current state of the application, the goal seems to have been reached. However, there are complications and limitations to the AR environment. The AR models within the environment stay in the correct position, this is true as long as the device using the AR application is nearly stationary otherwise the model will skew and move with the camera.

Nevertheless, since the current use requires the user to follow the virtual robot with the device, it will be difficult to keep the device stable enough for the afore-

mentioned issue not to occur. The current suggested solution to this predicament could be the use of a stabilizer, to keep the camera as steady as possible. Though it may not completely remedy the issue at hand it will prove rather effective in minimizing the effect and have a smaller error margin. This however is not the only solution as a more reliable one might be available. Even so, due to the constraint of the currently available assets, it falls beyond the scope and limitations of this project.

In section 1.5 it was stated that the environment would not provide any feedback to the robot or other objects in AR this could be a potential for further development. If the application for example could use image recognition to identify the surroundings Unity could send this information back to ROS when planning a trajectory.

### 4.4.4 Practicallity of the AR simulation

Compared to a traditional visualization tool like Gazebo, the main advantage of AR visualization is the ability to see the robot in the actual environment where it will be used. In Gazebo, the environment is static and changes in the real world can lead to discrepancies between the simulation and reality. This can cause the developer to assume that the robot's movements will work in the real world when in fact it may crash into objects that were not accounted for in the simulation. With AR, the environment is accurately represented, making it a useful option for controlling the robot's movements.

# 5
# Conclusion

The AR application effectively guides the TIAGo robot from its initial location to the desired destination. Potential collisions with objects or walls are prevented, thanks to the assistance of AR provided to the human operator. Nevertheless, the application exhibited instability in displaying the robot's position, despite having image targets with their corresponding AR models in the Unity environment.

In summary, this project accomplishes the set goals stated in section 1.4. Successful visualization of two AR objects and the avoidance of collisions with the aid of a human operator. The AR robot can by using ROS and Unity navigate to the other AR object. These results can be seen in chapter 4. However, further optimization and adjustments are required to make the application suitable for real-world use as discussed in section 4.4.

# Bibliography


[1] Niantic Inc, "Pokémon Go". [Online]. Available: https://pokemongolive.com/ Accessed on: 2023-01-28.

[2] Å. Fast-Berglund,L. Gong and D. Li, "Testing and validating Extended Reality (xR) technologies in manufacturing", Procedia Manufacturing, Vol #25, nr 8, ss 31-38, May 2018, doi:10.1016/j.promfg.2018.06.054

[3] Vamsi Krishna N, Gustav Silverstam, Ernesto Lozano Calvo, Darshan Gad, "Human-Robot Collaboration through Augmented Reality with the HoloLens2" (n.d). [Online]. Available: https://git.chalmers.se/phric/devels/bachelors/2023/eenx16-23_augmented_tiago/eenx16_23_23_augmented_tiago/-/blob/main/Documentation/Resources/Literature/projectgroup8_11015_2570830_Final_report__SSY226_Group_08.pdf, Accessed on: 2023-01-29.

[4] C.Halvarsson, "Interaktiva Fabriker i Augmented Reality Med hjälp av Microsoft Hololens", 2017. [Online]. Available: https://odr.chalmers.se/items/2764ffec-0cd3-4a9c-b038-4b6a98f0d634, Accessed on: 2023-01-29.

[5] J. Spång and I. Brinkenberg, "Sammanstrålning av världskoordinater hos en augmented reality-redo beräkningsplattform och ett AGV-system", 2017. [Online]. Available: https://odr.chalmers.se/items/429f6b7a-a387-43ab-b4f7-3693a12aa81c, Accessed on: 2023-01-29.

[6] PAL Robotics, "TIAGo", (n.d). [Online]. Available: https://pal-robotics.com/robots/tiago/ (Accessed on: 2023-01-29).

[7] PAL Robotics, "TIAGo TECHNICAL SPECIFICATIONS", (n.d). [Online]. Available: https://pal-robotics.com/wp-content/uploads/2022/04/Datasheet_TIAGo.pdf Accessed on: 2023-05-04.

[8] ROS, "TIAGo", 2020. [Online]. Available: http://wiki.ros.org/Robots/TIAGo (Accessed on: 2023-03-10).

[9] Open Robotics, "ROS", (n.d). [Online]. Available: https://www.ros.org/ (Accessed on: 2023-05-03).

[10] Open Robotics, "catkin", 2017. [Online]. Available: http://wiki.ros.org/catkin (Accessed on: 2023-05-03).

[11] ROS, "Writing a Simple Publisher and Subscriber (Python)" 2022. [Online]. Available: http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29 (Accessed on: 2023-05-05).

[12] ROS, "msg", 2023. [Online]. Available: http://wiki.ros.org/msg (Accessed on: 2023-05-05).

[13] ROS, "Topics", 2019. [Online]. Available: http://wiki.ros.org/Topics (Accessed on: 2023-05-05).

[14] ROS, "rqt_graph", 2018. [Online]. Available: http://wiki.ros.org/rqt_graph (Accessed on: 2023-05-05).

[15] ROS, "actionlib" 2015. [Online]. Available: http://wiki.ros.org/actionlib (Accessed on: 2023-05-03).

[16] Docker, "What is a Container?" (n.d). [Online]. Available: https://www.docker.com/resources/what-container/ (Accessed on: 2023-05-04).

[17] Open Robotics, "urdf", 2023. [Online]. Available: http://wiki.ros.org/urdf (Accessed on: 2023-05-03).

[18] ROS, "tiago_description" 2015. [Online]. Available: http://wiki.ros.org/tiago_description (Accessed on: 2023-05-03).

[19] Unity, "Unity Manuel: Scenes", 2023. [Online]. Available: https://docs.unity3d.com/Manual/CreatingScenes.html (Accessed on: 2023-04-11).

[20] Unity, "Unity Manuel: Packages", 2023. [Online]. Available: https://docs.unity3d.com/Manual/PackagesList.html (Accessed on: 2023-04-11).

[21] Unity, "Unity Manuel: Build Settings", 2023. [Online]. Available: https://docs.unity3d.com/Manual/Buildsettings-linux.html (Accessed on: 2023-04-11).

[22] Vuforia, "Vuforia Engine Overview", (n.d). [Online]. Available https://library.vuforia.com/getting-started/vuforia-features (Accessed on: 2023-04-11).

[23] Vuforia, "Getting started", (n.d). [Online]. Available https://library.vuforia.com/ (Accessed on: 2023-04-11).

[24] Vuforia, "Model Targets", (n.d). [Online]. Available https://library.vuforia.com/objects/model-targets (Accessed on: 2023-04-11).

[25] ARCore, "Overview of ARCore and supported development environments", 2022. [Online]. Available: https://developers.google.com/ar/develop (Accessed on: 2023-01-27).

[26] Samsung, "Galaxy Tab S6 Lite Wi-Fi", (n.d). [Online]. Available: https://www.samsung.com/se/tablets/galaxy-tab-s/galaxy-tab-s6-lite-10-4-inch-gray-64gb-wi-fi-sm-p613nzaanee/ (Accessed on: 2023-02-10).

[27] Microsoft, "HoloLens 2", 2023. [Online]. Available: https://www.microsoft.com/en-us/hololens (Accessed on: 2023-01-18).

[28] FORTINET, "What is Transmission Control Protocol TCP/IP?" [Online]. Available: https://www.fortinet.com/resources/cyberglossary/tcp-ip (Accessed on: 2023-04-10).

[29] Khan Academy, "Transmission Control Protocol (TCP)", (n.d). [Online]. Available: https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:

transporting-packets/a/transmission-control-protocol--tcp (Accessed on: 2023-04-10).

[30] CISCO, "What is a LAN?", (n.d). [Online]. Available: https://www.cisco.com/c/en/us/products/switches/what-is-a-lan-local-area-network.html (Accessed on: 2023-04-10).

[31] Verizon, "Wi-Fi", 2023. [Online]. Available: https://www.verizon.com/articles/internet-essentials/wifi-definiton/ (Accessed on: 2023-04-10).

[32] Unity, "About AR Foundation", 2022. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/manual/ (Accessed on: 2023-02-16).

[33] BHaptic, "TACTGLOVE", (n.d). [Online]. Available: https://www.bhaptics.com/tactsuit/tactglove (Accessed on: 2023-01-28).

[34] ROS, "geometry_msgs/Twist Message", 2022. [Online]. Available: https://docs.ros.org/en/noetic/api/geometry_msgs/html/msg/Twist.html (Accessed on: 2023-03-15).

[35] ROS, "control_msgs", 2022. [Online]. Available: https://docs.ros.org/en/noetic/api/control_msgs/html/index-msg.html (Accessed on: 2023-03-15).

[36] Vuforia, "Vuforia Engine", 2023. [Online]. Available: https://developer.vuforia.com/downloads/sdk (Accessed on: 2023-03-08).

[37] Unity, "Unity-Robotics-Hub", 2022. [Online]. Available: https://github.com/Unity-Technologies/Unity-Robotics-Hub (Accessed on: 2023-03-22).

[38] Unity, "ROS-TCP-Connector", 2022. [Online]. Available: https://github.com/Unity-Technologies/ROS-TCP-Connector?path=/com.unity.robotics.ros-tcp-connector (Accessed on: 2023-04-14).

[39] Unity, "Android environment setup", 2023. [Online]. Available: https://docs.unity3d.com/Manual/android-sdksetup.html (Accessed on: 2023-04-20).

[40] Lenovo, "Lenovo Yoga Tab 11", (n.d). [Online]. Available: https://www.lenovo.com/se/sv/tablets/android-tablets/lenovo-tab-series/Lenovo-Yoga-Tab-11/p/WMD00000472 (Accessed on: 2023-02-10).

[41] Lenovo, "Lenovo Tab P11 plus", (n.d). [Online]. Available: https://www.lenovo.com/se/sv/tablets/android-tablets/lenovo-tab-series/Lenovo-Tab-P11-Plus/p/WMD00000476 (Accessed on: 2023-02-10).

# A

# Appendix

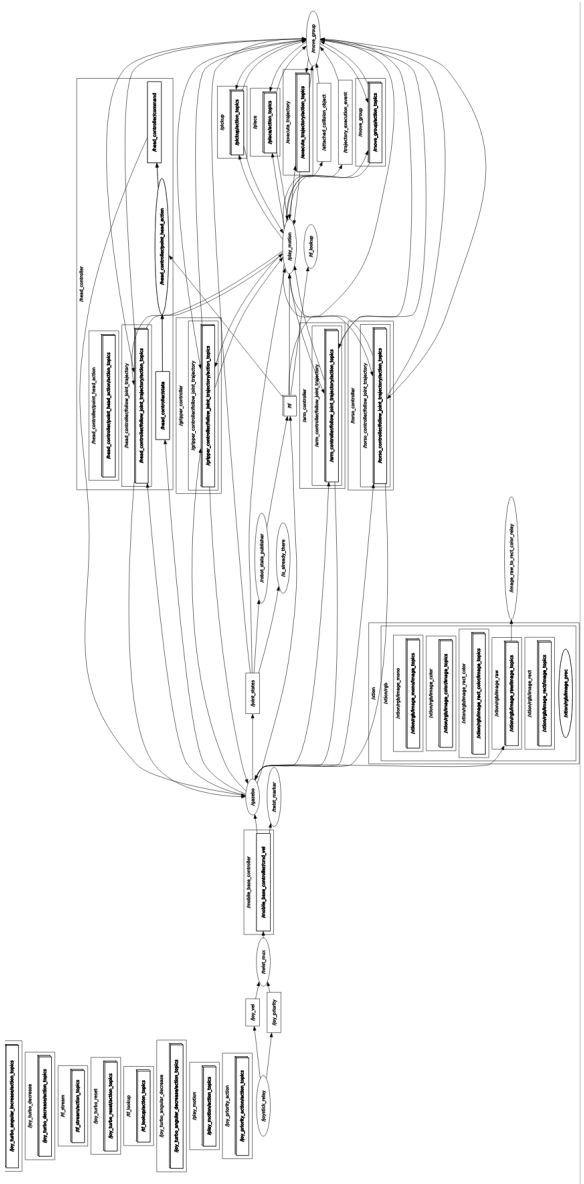


**Figure A.1:** Showing the network of nodes.

# B

# Appendix

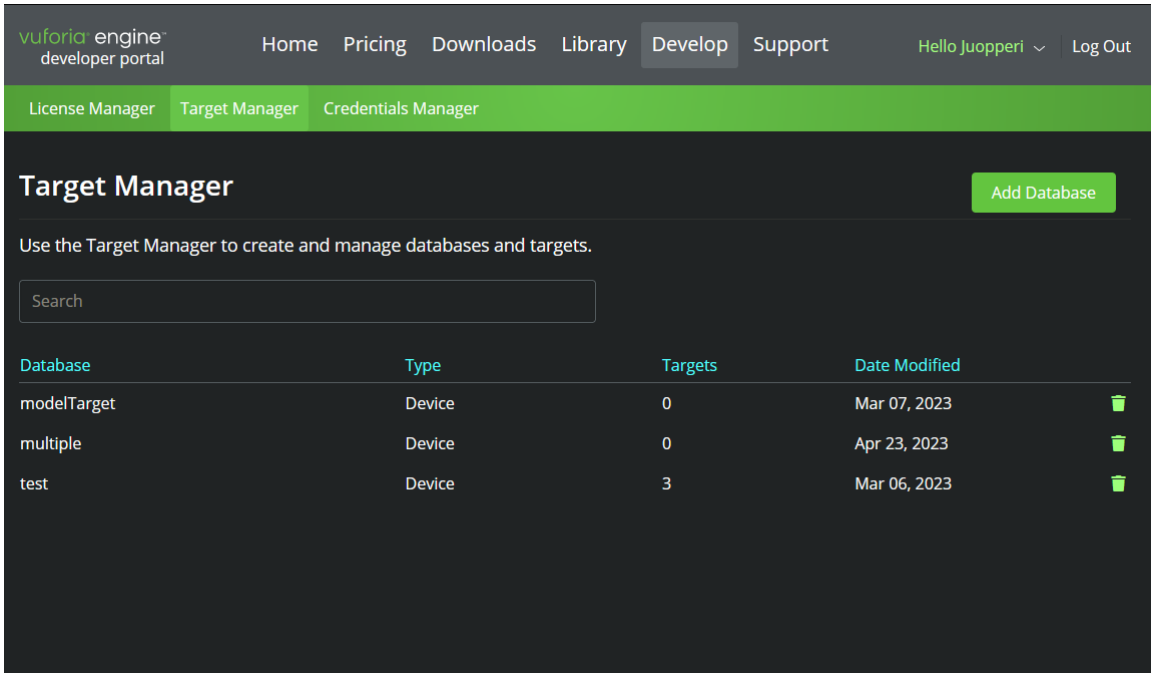



**Figure B.1:** Vuforia add database.

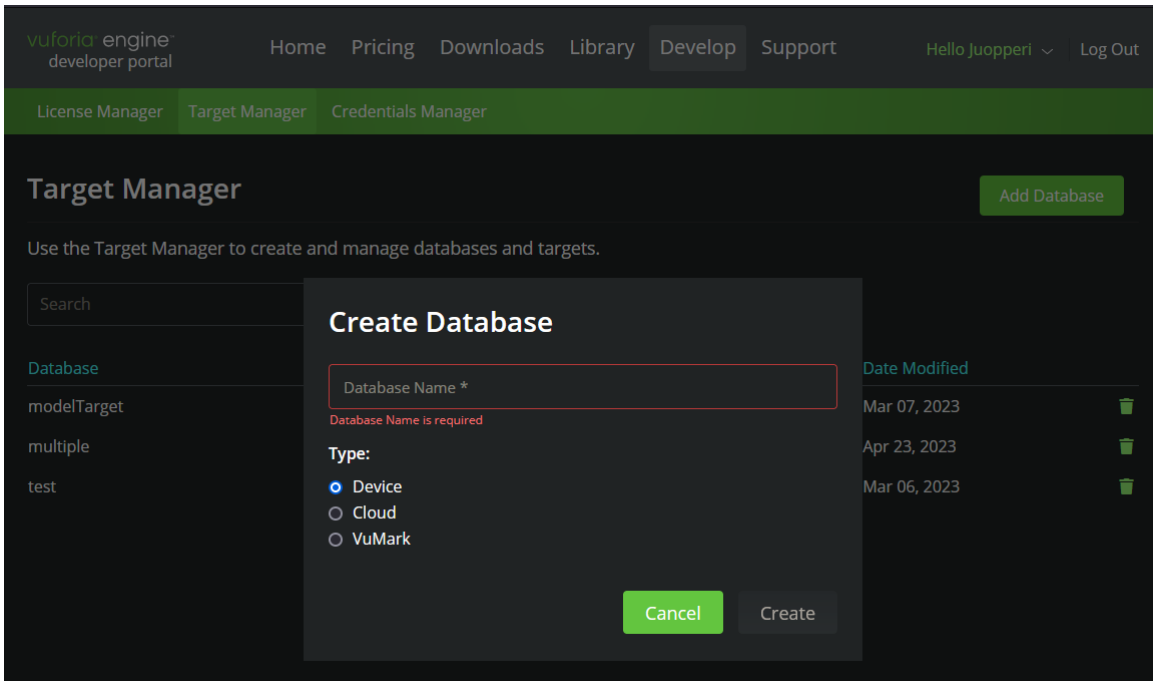



**Figure B.2:** Vuforia check device.

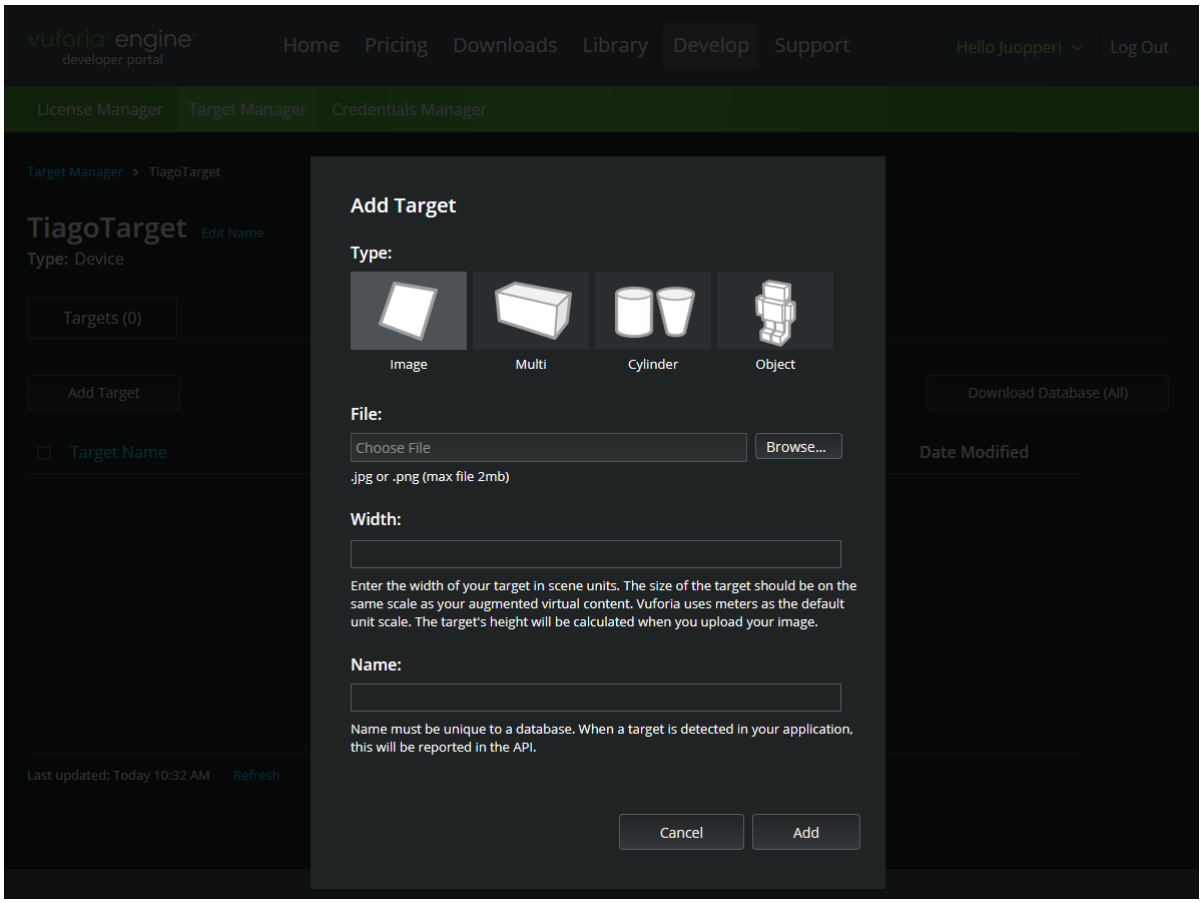



**Figure B.3:** Vuforia add target in database.

# C

# Appendix

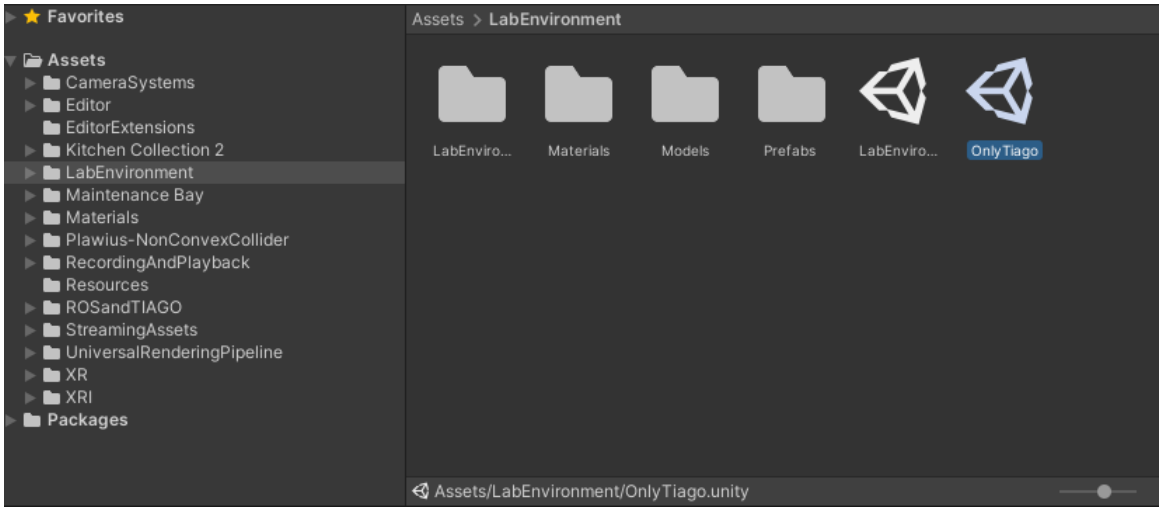



**Figure C.1:** Unity accessing the robot model.

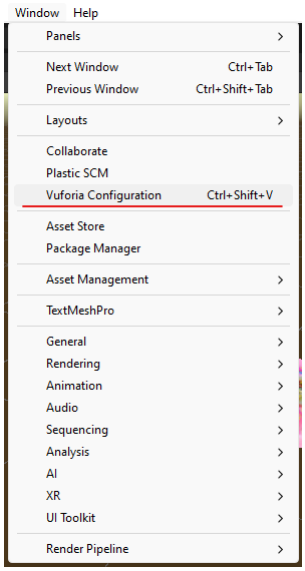



**Figure C.2:** Path to open Vuforia's Configuration window in Unity.

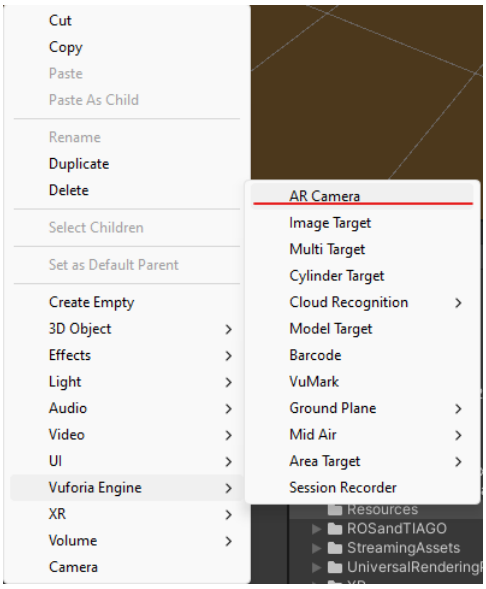



**Figure C.3:** Unity adding an AR Camera as an object.

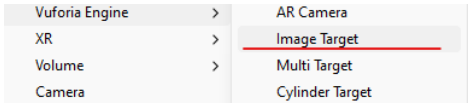



**Figure C.4:** Adding an image target object into the scene in Unity.

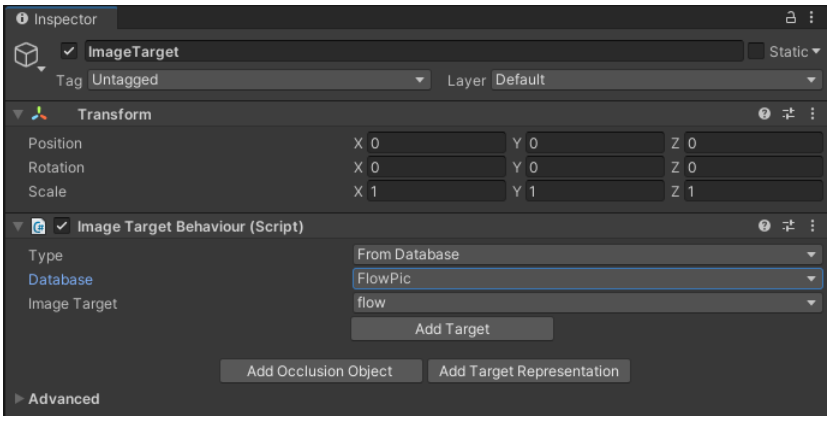



**Figure C.5:** Unity configuring image target behaviour, in this instance the database is called "FlowPic" and the image target is "Flow".

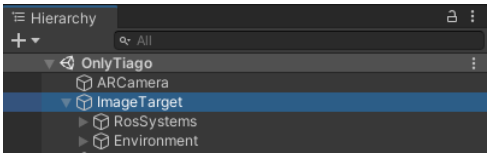



**Figure C.6:** Setting RosSystems and Environment as children of the image target in Unity.

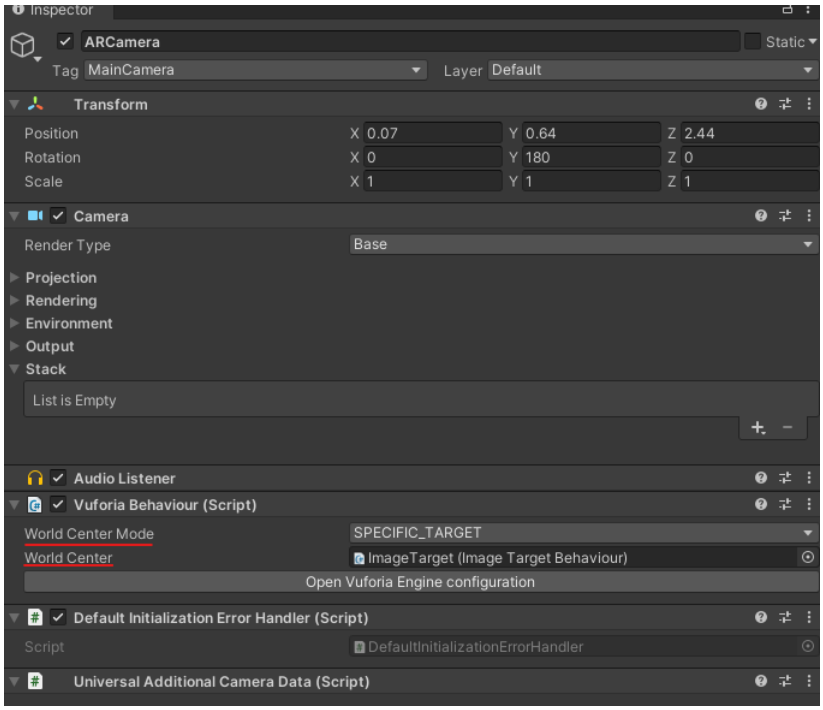



**Figure C.7:** Setting the "World Center" in Unity.

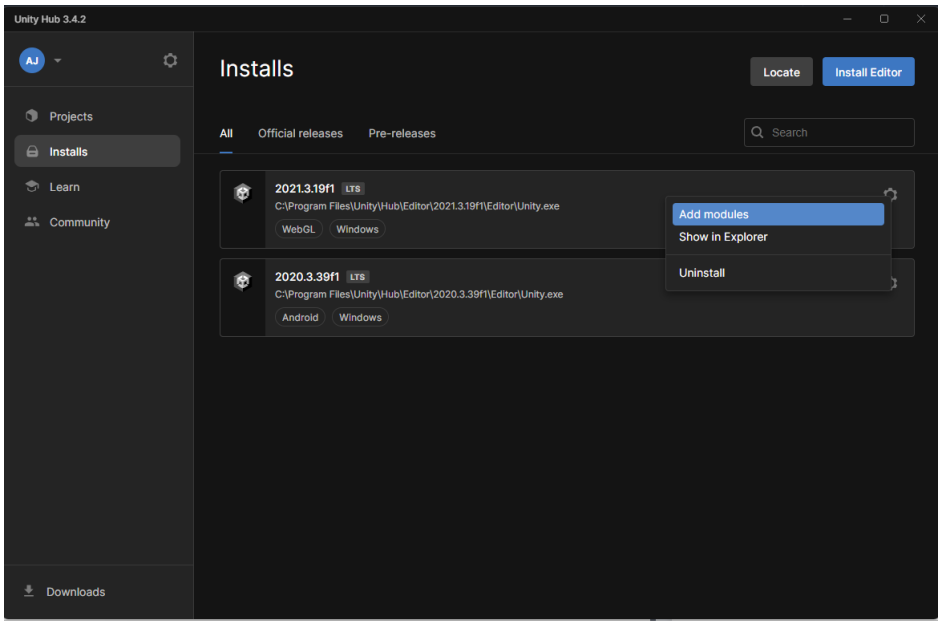



**Figure C.8:** Adding a module in the Unity hub.

# D

# Appendix


**Figure D.1:** TIAGo robot deployed on a OnePlus 8.

**DEPARTMENT OF ELECTRICAL ENGINEERING**
**CHALMERS UNIVERSITY OF TECHNOLOGY**
Gothenburg, Sweden
www.chalmers.se


CHALMERS