Augmented Reality Interface for Human-Robot Interaction

Eliisabet Hein

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2018

Abstract

Robots are becoming increasingly ubiquitous in many environments, from warehouses and factories to schools and museums. Many tasks are best accomplished when robots work alongside humans in an environment that facilitates collaboration. However, a major roadblock for collaborative systems is the fact that the robot's planning processes remain invisible to the human participants. They are thus unable to anticipate the robot's actions and respond accordingly; the resulting uncertainty is a threat to both the usability of collaborative systems and the safety of the human. This project is concerned with exploring the viability of using visual head-mounted augmented reality to bridge this gap and create a system that enables the human to better anticipate and respond to the robot's actions. We created an augmented reality interface for a handover task with the Baxter robot and conducted a user study to evaluate performance (time per handover) and subjective user experience when using the interface, compared against a control case with no visualisations. Time per handover was measured both for a simple handover case and a complex handover case. For the simple case, we found no significant decrease in average time per handover when using the interface with visualisations; however, we found that using the interface with visualisations led to a statistically significant decrease in the average time per handover for the complex case. The participants on average rated the interface with visualisations as more mentally demanding and complicated, as well as requiring of more effort, but also less frustrating and more supportive. We conclude that there is evidence that head-mounted augmented reality interfaces can facilitate human-robot interaction; however, there are still technical and design aspects that remain to be tackled.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Eliisabet Hein*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation

As safety cage-less robots are becoming increasingly available and less expensive to install in many environment, especially in small-scale manufacturing and assembly, so too are close-quarters human-robot collaborative teams becoming more common in many industrial settings. The benefits of humans and robots working together are embodied in the distinct sets of skills that each brings to the interaction; the human provides creative thinking and flexibility, while the robot provides precision and consistency [33, 32]. However, the safety and comfort of the human in these environments remains an unsolved problem, in large part because the details of behind-the-scenes processes and motion plans of the robot can seem obscure and opaque even to trained operators. As such, working in close proximity with robots can be daunting and even dangerous. This project is concerned with implementing and evaluating a proposed method of facilitating safe and transparent close quarters interaction through head-mounted augmented reality display.

1.2 Background

Along with other technological advances such as Internet of Things and artificial intelligence, *cyber-physical systems*, and in particular collaborative human-robot systems, are an important part of the *Fourth Industrial Revolution*, better known as *Industry 4.0*, which is the process of digitalisation and technological innovation that currently fuels the manufacturing industry [2]. A crucial difference between Industry 4.0 and past approaches to industrial robotics is that instead of *replacing* human workers with robots, robots are introduced to *assist* the worker by performing tasks matched with their skillset, such as those requiring heavy physical labour or extreme precision, leaving humans to handle conceptual tasks and guide the interaction. However, to accomplish this goal, robots must work in close quarters with humans without the presence of safety cages while also ensuring the safety and comfort of the human worker.

To achieve the goal of safer interaction and eliminate the need for safety cages, significant advances have been made by robot manufacturers such as KUKA [23], FRANKA EMIKA [15] and Rethink Robotics [36] to create robots with built-in safety features such as collision prevention. However, this fails to solve the problem of how to facilitate better understanding of the robot's behind-the-scenes processes and decision-making. A prominent method that has been proposed to provide cues to the human about the robot's processes is through *augmented reality interfaces*. **Augmented reality** (**AR**) is defined as a view of the real environment, which has been altered by adding computer-generated perceptual information. It is used to enhance natural environments and offer perceptually enriched experiences [8].

While augmented reality can also include other senses, such as touch, smell or hearing, this project will concentrate solely on *visual* augmented reality. The reason for this is twofold: firstly, vision has been shown to be the most important sense for perceiving information [54]. Secondly, visual augmented reality is also the most prominent approach to augmented reality in research.

A variety of technologies are available for visual augmented reality interfaces. *Spatial augmented reality* involves using light projection to beam information directly onto objects. This approach has been used to allow humans to anticipate a robot's planned movement by projecting navigation information on the floor in front of a mobile robot in a shared environment [3, 10, 30], and to indicate the robot's next action in collaborative tasks in an industrial environment, such as car assembly [1]. Spatial augmented reality benefits from the fact that the projector can be mounted on the robot, which eliminates the need for humans to wear specialised devices. However, it also requires the presence of a flat surface such as a wall or floor, and the flexibility of what can be visualised is limited.

This, as well as significant improvements in the availability and quality of augmented reality headsets such as Google Glass, Microsoft HoloLens and Meta 2, have led to a focus on *see-through augmented reality* in most recent research into human-robot interaction. See-through augmented reality, which involves a head-mounted display to augment the human's field of vision, has been shown to significantly improve worker confidence and facilitate more effective human-robot collaboration in collaborative industrial tasks [28, 29]. It has also shown to be an intuitive interface for programming collision-free trajectories [4].

Convincing recent evidence for the use of augmented reality in human-robot interaction comes from Rosen *et al*. [37], who showed that when predicting robot arm motion, use of an augmented reality headset lead to a significant increase in accuracy and decrease in time taken to predict the robot's arm motion over both a control case with no visualisations, and over visualisations on a 2D screen. Additionally, participants rated the head-mounted display to be the most enjoyable, as well as the best for understanding the robot's arm motion. This indicates that head-mounted augmented reality displays have significant potential for applications where predicting the robot's movements would facilitate better collaboration by providing the human with information about the robot's processes via a visual interface.

1.3 Project description

This project is concerned with the design, creation and evaluation of a head-mounted visual augmented reality interface to enhance human-robot collaboration. Two separate completion criteria were defined in the project description:

1. Completed and tested integration between ROS and Unity for AR display.
2. Developed and tested use-cases where augmented information improves interaction between a human and real robot manipulator.

As such, the project could be divided into two main parts. Firstly, the *technical aspect*, which involved the development of a channel of communication between a workstation connected to the robot manipulator – in this project, the **Baxter** robot developed by Rethink Robotics – running the **Robot Operating System**, and a separate device running a **Unity** application with visualisations, which were displayed using the **Meta 2** augmented reality headset. These tools will be introduced in more detail in Chapter 2. The implementation will be described in Chapter 3.

Secondly, the project involved a *design aspect*, which was concerned with creating visualisations for a specific use case. We had the opportunity to use a collaborative object handover task developed by EPFL LASA [see Section 1.4]. Using this task, we then conducted an experimental user study to determine the effect of using an augmented reality interface to enhance interaction. The process of designing and implementing the interface will be described in Chapter 3, and evaluation will be described in Chapter 4.

1.4 Collaboration task

This section will introduce the collaborative object handover task that was used to develop and evaluate the interface. The code for the task was provided by Seyed Sina Mirrazavi Salehian, Nadia Figueroa, and Aude Billard from the Learning Algorithms and Systems Laboratory at the École Polytechnique Fédérale de Lausanne and adapted for use with the Baxter robot by Joshua Smith. The task was developed to demonstrate coordinated dynamic object handover using both arms of a two-arm robot. However, in our case the task was adapted for interaction using only one arm [see Section 3.9].

The interaction could be broken down into the following steps: firstly, the human actor held an object, which was tracked through some means (e.g. through computer vision techniques such as marker tracking). They initiated a handover by approaching the robot while holding the object. The behind-the-scenes processing used the velocity and direction of the object to dynamically calculate an *intercept point* where the object was predicted to intercept the robot's workspace. Once the certainty that the object would intercept at the predicted location exceeded a pre-defined threshold, the arm would switch from the default task – remaining in default position – to the *grasping* task and move dynamically to intercept the object at the predicted location. If the trajectory of the object was not as expected, i.e. the human stepped back and the object moved away from the robot, the arm moved back to default position.

1.5 Hypothesis

To test whether adding augmented reality visualisations to a handover task – such as the one described in the previous section – facilitated better collaboration between the human and the robot, we defined the following hypothesis, which we then aimed to verify by conducting an experimental user study:

Using an augmented reality interface for a human-robot collaboration task produces an improvement in task performance when compared to performing the collaboration task without an augmented reality interface.

We defined *performance* as the average *time per handover* for the handover task, as this captured the success of the system through a easily-measurable objective metric. Additionally, to measure the usability of the developed system, we had the participants in the experiment rate the interface with two subjective evaluation metrics, NASA-TLX [see Section 4.2.1] and a shortened version of UEQ [see Section 4.2.2]. We set additional hypotheses for each measured sub-scale, as well as more detailed hypotheses for time per handover, which are defined in Section 4.3.

While the research questions we investigated were similar to those explored by Rosen *et al*. [37], we chose to only use a no-visualisations interface as a control case for the augmented reality system, and not include a 2D interface as Rosen *et al*. did. The reasons for this are two-fold: firstly, the results from Rosen *et al*. showed that the 2D interface achieved lower accuracy than the mixed reality interface, and lower subjective evaluation scores than both the mixed reality interface and the interface with no visualisations. Secondly, while the task performed in the experiments in Rosen *et al*. was *passive* – deciding whether a path plan for a robot arm would collide with a stack of boxes – in our case the task was *active*. Using a 2D interface would have required the participants to divert attention away from the task to consult the visualisations; as such, we made the assumption that a 2D interface would have consistently achieved lower performance scores than the other two interfaces. While experiments could in future be conducted to verify this assumption, to limit the scope of the work for this project we decided to eliminate the 2D interface.

1.6 Summary of results

Our final developed interface visualised three aspects of the interface – the *gripper target*, the *tracked object* and the *intercept point*. We evaluated the interface against the control case (no visualisations) in an experimental user study with 14 participants. In the experiment, the participants completed 6 *simple* handovers – defined as handovers where the object was passed close to the gripper – and 4 *complex* handovers – defined as handovers where the object was passed further away from the gripper – for both interfaces. We also asked the participants to evaluate both interfaces on the NASA-TLX and UEQ scales. The process of designing the experiment is described in further detail in Section 4.1 and the process of conducting the experiment in Section 4.4.

While we did not find a statistically significant difference between the average time achieved with either interface in the simple handover case, we did find that for the complex handover, there was a statistically significant decrease with 97% confidence in average time per handover with our developed interface. Our results for the subjective evaluations were in the most part in line with those reported by Rosen *et al.*; we found that the interface with visualisations was rated as more *mentally demanding* and requiring more *effort*, but also less *frustrating* and more *supportive*. A thorough analysis of the achieved results is given in Section 4.5

1.7 Our contributions

- Adapting existing software libraries for general use to transmit any information from a ROS network to Unity over a network socket.
- Conducting an investigation into marker-based tracking methods using the Vuforia and ARToolKit libraries to evaluate their suitability for this project.
- Designing and creating visualisations for a specific collaboration task provided by EPFL LASA, involving an object handover.
- Conducting an experimental study to evaluate the effect of using the developed interface with visualisations, compared to a control case with no visualisations.

1.8 Report outline

The succeeding chapters of the report are structured as follows:

Chapter 2 will provide the required technical context for understanding the work conducted for this project. We will give an overview of the Robot Operating System (ROS) and explain the associated terminology, introduce the Baxter robot, the Unity development engine with associated terminology, and the Meta 2 headset.

Chapter 3 will outline the design and implementation of all work that was undertaken to create the interface. We will detail the process of adapting existing code to set up the ROS-Unity communication bridge, extending the communication, exploring marker tracking and creating the augmented reality interface for the handover task.

Chapter 4 will show how the implementation was evaluated, including an outline of the process of designing and conducting the experiments, and introduce the metrics that were used for both objective and subjective evaluation. We will then provide a thorough analysis of the achieved results.

Chapter 5 will summarise the work that was conducted and the results that were achieved. We will finish by discussing the limitations of the project and providing ideas for further work.

Chapter 2

Technical Background

This chapter introduces the required terminology for explaining the work performed for this project, particularly in relation to the ROS framework and Unity, which were the main platforms for software development in this project. It also gives an overview of the main hardware used in this project: the Baxter robot and the Meta 2 headset.

2.1 Robot Operating System (ROS)

We used the *Robot Operating System* (*ROS*) [35] for robot-side development in this project. ROS is a popular open-source software framework for robot development, and provides operating-system-like services such as low-level device control, network protocols and package management [7]. It was originally created by the Stanford Artificial Intelligence Lab, and is currently maintained by the Open Robotics Foundation.

Each process in ROS is referred to as a **node**; each node usually only performs one task, and a robot control system is usually made up of many nodes. As ROS was designed to be *flexible* and *loosely-coupled*, nodes can be run on different machines as long as they are connected on the same local network [7]. This means that some processes, especially ones that only consume information from other internal nodes, can be run directly on the on-board computers of the robot, while more computation heavy processes, or ones that require interaction through the command line, can run on an external computer.

While nodes in a ROS network connect to each other in a *peer-to-peer* manner (i.e. directly from one node to another), the namespace in the network is managed by a **master node**, which acts as a lookup table similarly to a DNS server. Nodes are linked into a runtime graph through information streams called **topics**, which nodes can send and read **messages** into and from using a subscriber/publisher model. Many nodes can publish/subscribe to the same topic at any one time, and each node can publish/subscribe to multiple topics. Due to the loosely-coupled nature of the ROS system, publishers and subscribers are not aware of each other's existence. This also means that subscribers and publishers can be added dynamically at run-time.

The main unit of organisation in ROS is a **package**, which can contain nodes, ROS-dependent libraries, datasets, configuration files etc. ROS is intended to facilitate collaboration and re-using code, and there are many open-source packages available for a variety of purposes [7]. For this project, we used the *RViz* visualisation tool to visualise robot position, the *Moveit!* path planning library to create trajectory plans and *Rosbridge* to expose internal ROS communication to external processes on the same local network. In Section 3.5 and Section 3.6 we extended the *ros_reality_bridge* package created by the Humans to Robots laboratory.

ROS supports development in multiple programming languages through client libraries. Our development was done using the Python programming language with the `rospy` client library.

2.2 Baxter robot

In this project, we use the Baxter collaborative robot created by Rethink Robotics. It is a humanoid robot with two arms, each of which has 7 degrees of freedom. A picture of the Baxter robot is given in Figure 2.1

While the Baxter robot was initially developed for manufacturing and industrial assembly, it is also used by universities and research laboratories for various areas of study (e.g. [37, 38, 26]). Its suitability for research is aided by the fact that the Baxter robot can be operated through a ROS SDK, rather than through proprietary software. It is also designed to be safer and more approachable than most industrial robots, with build-in proximity sensors that make it reduce force before collision, and does not require a security cage to operate. As such, it is perfect for research into close-quarters human-robot collaboration, as in this project.

Figure 2.1: The Baxter collaborative robot as used in this project.

2.3 Unity3D

Unity3D is a widely-used cross-platform game engine, which is primarily used to develop 3-dimensional games. However, it is often also used for other visualisation applications, including visualisations and teleoperation in robotics (e.g. [37, 6, 20]), and for developing VR and AR applications – for example, the Meta 2 headset is specifically targeted at Unity development.

Unity supports development through both GUI drag-and-drop functionality and **scripting** in the C# programming language. The basic components of a Unity **scene** – a full self-contained 3D environment – are **GameObjects**. A GameObject can have any number of components attached to it, such as *renderers* and *meshes* which control how the GameObject is visualised and *scripts* which control the GameObject's behaviour.

Each GameObject has a specified **position** and **rotation** defined in terms of the x, y and z coordinates; position is defined in terms of Cartesian coordinates and rotation is defined as *quaternions*[1] or *Euler angles*[2]. Unlike ROS and many other platforms, Unity operates using a left-hand coordinate system. Figure 2.2 illustrates the difference between the two coordinate systems.

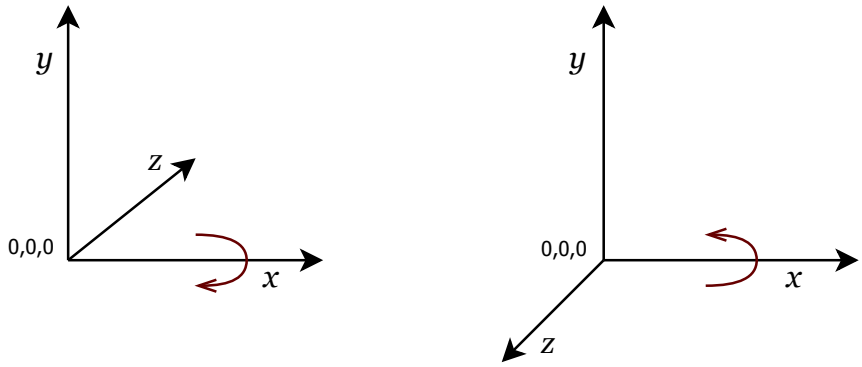


Figure 2.2: Left-hand coordinate system (left) and right-hand coordinate system (right). As we can see, the direction of the z axis is reversed. Additionally, while rotation in a left-hand coordinate system is *clockwise* around the x axis, in a right-hand coordinate system the rotation is *counter-clockwise*.

This meant that when transferring coordinates from ROS to Unity [see Section 3.4], the coordinates had to be converted from one coordinate system to the other.

GameObjects are organised in the scene in a **hierarchy**; by default all GameObjects are children of the scene root, but they can also be nested under other GameObjects. In this case, their position and rotation is dependent on that of the parent object.

[1] Quaternions are a method of representing rotation in space through complex numbers [52]; the rotation is defined by the x, y, z and w values. This is the method used internally in Unity.

[2] Euler angles are a simple method of representing rotation in 3D space by defining the rotation around each of the three axes (x, y and z) between 0 and 360 degrees [51]. Unity includes functionality to represent rotation in Euler angles.

Unity programs can be run inside the GUI or built to an executable file if deploying to other devices or operating systems. For all development we used the first option, as it was significantly faster and more convenient. Once a Unity program is run, it operates on a **frame-by-frame** basis: each script has an `Update()` function which is called every frame and the code contained within it is executed.

While the built-in functionality in Unity is extensive, it can also be extended by importing external libraries and assets. This is made simple through drag-and-drop functionality of the GUI. Libraries can contain scenes, **Prefabs** (pre-defined, or *pre-fabricated*, GameObjects), scripts, resources etc. In this project, we extended the *ROS Reality* library and used the *ARToolKit* library for marker tracking.

We used Unity for two reasons: firstly, the Meta 2 development kit is targeted at Unity; secondly, we used it because it is a flexible and powerful engine for creating visualisations, especially since the scene can be manipulated as desired using C# scripting. Finally, Unity also has strong community support and many open-source libraries that can be imported easily. We used Unity version 2017.3.0 for development, as it was the most recent version when we began development, and also the version of Unity that was bundled with the Meta SDK.

2.4 Meta 2

The Meta 2 [see Figure 2.3] is a recently-released AR headset developed by the Meta company. The version used in this project was part of the development kit, and new software was released multiple times over the course of this project. As it is very recent, we do not know of any published research using the Meta 2, either in the field of human-robot interaction or augmented reality in general.

Figure 2.3: The Meta 2 augmented reality headset as used in this project.

The Meta 2 claims to have the widest field-of-view (90 degrees) out of any head-mounted AR devices available on the market. It also has a high-resolution display and built-in tracking that is able to localise the device in space immediately without using any external tracking systems by combining computer vision from a built-in camera with acceleration and rotation information captured using on-board sensors. This provides the Meta with inside-out SLAM[3] tracking, where all information about the environment is derived from internal information. Augmented reality through Unity is implemented by keeping the virtual objects in the scene stationary, while the position and rotation of the headset moves to reflect movement of the headset in the real world. While the built-in tracking results in stable augmented reality content, the Meta software currently includes no supported way to link visualisations to real-world objects; the issues posed by this fact will be discussed in further detail in Section 3.8.2.

As the headset is still in development and the software is frequently updated, we cannot guarantee that the code developed for this project will be compatible with new releases of the Meta SDK.

[3]Simultaneous Localisation and Mapping – a tracking paradigm in which an unknown environment is mapped out while simultaneously determining the device's position in the environment.

Chapter 3

System Design & Implementation

This chapter will outline the process of designing and implementing the augmented reality interface according to the project brief given in Section 1.3. Due to the multifaceted nature of the system and the number of various tools used in development, the design and implementation processes were closely interwoven, with each successive step feeding into the next. The design decisions and following implementation of each aspect of the system will be explained thoroughly, with reference to the technical background and information about tools used in development, which were outlined in Chapter 2. We will supplement out explanation with figures and code snippets as appropriate.

3.1 System setup

As a Linux distribution was required to run ROS, and the Windows operating system was required for Meta 2 software, we used two separate computers for development; the first computer ran the Ubuntu 14.04 operating system, and the second computer the Windows 10 operating system. The two computers, as well the Baxter robot, were connected over a local wired network. The Meta 2 headset was connected to the Windows computer through USB 3 and HDMI connections. We added another Windows 10 machine when the Visualeyez tracking system was added [see Section 3.8.2]. The final setup consisted of three computers; however, we hypothesise that this could potentially be simplified in the future by using the same machine for the Meta 2 and the tracking system.

We also attempted to simplify the system by running ROS in a virtual machine on the Windows computer. However, we found that due to the low processing power of the virtual machine, this introduced too much lag into the system; as such, we anticipate that at least two computers will be required to setups with AR systems and ROS-enabled robots, such as the one described in this project.

3.2 ROS-Unity communication

As explained in Section 2.3, we used Unity to create visualisations for the interface, and ROS to develop the robot-side code. The first challenge was to create a bridge to transmit information from ROS to Unity. On the Unity side, information could be received through a network socket using the `websocket-sharp` [40] C# websocket library. However, a number of methods were considered to identify the best way to expose information from ROS to the local network, and the best way to organise the communication.

3.2.1 Rosbridge

Rosbridge [27] is a standard ROS package that provides a JSON API to expose local ROS communication to external programs connected on the same network [see Section 2.1]. The first option we identified to create the ROS-Unity bridge was to create a new implementation with Rosbridge and a websocket in Unity. However, as some pre-existing implementations were publicly available, we first examined these to determine whether we could leverage an existing project rather than creating one from scratch.

3.2.2 ROS Reality

The first codebase to be considered was developed by the Humans to Robots Laboratory at Brown University. Coincidentally, this codebase was developed in the same lab as the experiments described by Rosen *et al.* [37], which heavily influenced our project. This codebase was particularly relevant due to the use of both the Baxter robot and Unity. This project was published in two GitHub repositories, ROS Reality [46] for the Unity library and ROS Reality Bridge [45] for the ROS package. We will henceforth refer to the whole codebase as ROS Reality.

The project was created to visualise the state of the Baxter robot using Unity. The code on the ROS side created a new node, `unityNode`, which subscribed to a topic publishing the position and rotation of each link of the robot. This information was then published over a websocket and read in on the Unity side. This project used Rosbridge to open a socket that the Unity-side socket connected to, but rather than using either the JSON messaging format or defining ROS message formats on the Unity side, the authors used a custom messaging format.

To visualise the pose of the robot, this project also included functionality to parse a URDF[1] file, extract each link, create a GameObject for each link and add a 3D mesh from file. Additionally, as the coordinate frame in ROS follows the right-hand rule, while Unity uses a left-hand coordinate system [see Section 2.3], it was necessary to convert the received coordinates from one frame to another; functionality for this was also included in the ROS Reality codebase.

[1] Unified Robot Definition Format – the standard ROS specification of a robot model in XML format, including each link and the name of the corresponding 3D mesh file for visualisation.

We identified that some additional work would be required to remove unnecessary dependencies on some older libraries e.g. SteamVR, which was used to create visualisations in the original project. However, we concluded that overall this codebase was an excellent starting point for our project, as it included a base for a ROS-Unity bridge which was flexible enough that it could easily be extended to carry different information for the interface, and contained additional functionality that we would have otherwise had to create from scratch.

3.2.3 Alternatives

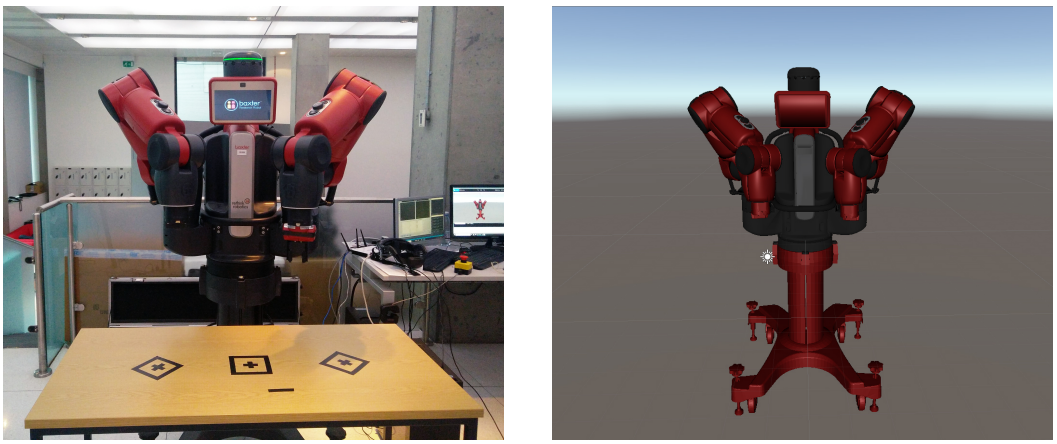
As an alternative to ROS Reality, we identified a Unity library – ROSBridgeLib [44] – that was also specifically aimed at providing a bridge between ROS and Unity. Similarly to ROS Reality, ROSBridgeLib used Rosbridge and `websocket-sharp`. However, instead of the custom messaging format used in ROS Reality, ROSBridgeLib included class definitions for a subset of ROS messages. When comparing the two options, we concluded that the benefits gained from using standard ROS message formats (rather than a custom format as in ROS Reality) were outweighed by the fact that ROS Reality also already included functionality to read the URDF definition of the Baxter robot and visualise the position of each link. As such, we decided to use ROS Reality as the basis for the ROS-Unity bridge.

Additionally, a new Unity library, ROS# [12], has been released by Siemens AG since our investigation was conducted. This library is is significantly more extensive than ROSBridgeLib and includes message definitions as well as other tools, such as functionality to read robot definitions from URDF. As such, we can see some benefit in migrating to the ROS# library if future development of this project was undertaken.

3.3 Setting up ROS Reality package

Once the ROS Reality package had been identified as a suitable starting point, the Unity project was cloned from [46] and the ROS package was cloned from [45]. Removing unnecessary dependencies on SteamVR and Kinect from the project was trivial and could be achieved by removing some lines of code from the ROS package and the Unity project. After these modifications had been performed, the initial project could be run by executing the following steps:

1. Ensuring that the system setup was correct, i.e. that Baxter robot was turned on and that the robot and computers were connected to the same local network.
2. Launching ROS Reality on the ROS computer using the provided launch file.
3. In Unity, specifying the IP address of the ROS computer and the port that ROS Reality was launched on (port 9090 by default) in the `ip_address` field of the `WebsocketClient` GameObject.
4. Running the Unity project in play mode.



(a) Real pose (b) Visualised pose

Figure 3.1: Real pose of the robot (a) compared to the visualisation of the current pose (b).

3.4 Visualising current position

The ROS Reality codebase was intended to support visualising the current pose of the robot in Unity. However, once the code had been set up, we could determine that even though GameObjects representing each link of the robot were being created and their position and rotation was being updated every frame, the links were not being visualised. We found that the default path pointing to the folder containing the 3D meshes was incorrect; once we moved the meshes to the correct folder, the robot was visualised successfully and the live state of the robot was visible in Unity when the steps described in Section 3.3 were followed. A comparison of the real robot and the visualised robot can be seen in Figure 3.1.

Additionally, we found that we could not move or rotate the robot in the Unity scene, which was due to the fact that in the original codebase, the position and rotation of each link were defined and updated *globally* rather than *locally* relative to the root of the robot. Foreseeing that we would need to move the robot as required, we altered the code such that every robot link GameObject was set as a child of the robot root GameObject in the scene hierarchy. This enabled us to move and rotate the robot as a whole as required, and was more in line with design practices for Unity, which recommend that objects should be structured in a logical hierarchy in the scene.

3.5 Extending communication

Once the work described in Section 3.3 and Section 3.4 was complete, we could communicate and visualise the current pose of the robot. This was useful to prove that the websocket was operational, the positions were read correctly and the coordinate transformation was correct. Additionally, lack of delay when comparing the visualised position to the real position while the robot's arms were moving indicated that this communication setup had potential for use throughout the project.

However, while visualising the current position could be used to debug the connection and show that the ROS-Unity bridge was performing correctly, additional information would need to be transmitted to augment the interaction task described in Section 1.4.

On the ROS side, the relevant code for our task was contained in the `unityNode.py` file. This file collected information about the current position and rotation of each link of the robot and published this information on the `ros_unity` topic using a custom message format. The Unity side code received this information through the websocket, parsed it in the `WebsocketClient` script and visualised the information.

We extended this code such that it could be made to transmit information from any topic on the ROS network if the following was specified:

- A short name for the information, e.g. `position`, which is used to create a new topic to publish the information on, e.g. `ros_unity_position`.
- A subscriber to the desired topic on the ROS side.
- A new callback function to call if the subscriber receives a message, which builds the message via the message builder and publishes it on the corresponding topic.
- A message builder to create a string representation of the information; creating the message builder can be skipped if an existing one can be used.

We will use parts of the code from the `unityNode.py` file to explain how this was achieved and outline the performed modifications. For clarity, we removed some sections of code; this will be indicated with asterisks (*), with a short description of the contents of the removed code.

The overall structure of the code before any modifications was as follows:

```
def message_builder(link_dict):
    # *** build position message ***

def get_transform(link, tf_listener):
    # *** update link dictionary for one link ***

def main():
    # *** main loop ***

if __name__ == '__main__':
    main()
```

As we can see, the code contained the `main()` function, which contained the main loop of the node, and the`message_builder()` and `get_transform()` functions. In the original code, both creating a subscriber to the `TransformListener`, which contained the position and rotation of each link, and creating a publisher to publish this information on the `ros_reality` function were performed in the main loop. However, as we had to keep track of multiple subscribers and publishers sice we were relaying information from multiple topics, we moved the functions inside a new class, `UnityNode`. This allowed us to initialise the code with multiple publishers as required and keep track of them as *local variables*.

As we would have multiple message builders, we renamed the `message_builder` function to `_build_position_msg`. To clean up the code in the main loop, we moved the functionality to update the link dictionary to a new `_update_link_dictionary` function and combined it with the `get_transform` function.

The structure of the code after modifications was as follows:

```
class UnityNode:
    def __init__(self, topics):
        # *** __init__ contents ***

    def _build_position_msg(self):
        # *** build position message ***

    def _update_link_dict(self):
        # *** update link dictionary ***

    def run(self):
        # *** main loop ***

if __name__ == '__main__':
    topics = ["position"] # list of information to publish

    node = UnityNode(topics)
    node.run()
```

The `topics` list could be extended to contain any information. The names in this list were used for two purposes in the new code: firstly, they were used to create publishers for new topics following the naming convention `ros_unity_X` where `X` was the name of the topic. For example, in this case, the topic was called `ros_unity_position`.

This was done in the `_init_` function of the `UnityNode` class as follows:

```
def __init__(self, topics):
    self.publishers = {} #dictionary of publishers

    print("------ ADDING PUBLISHERS ------")
    #create a publisher for each desired topic
    for topic in topics:
        topic_name = "ros_unity_{0}".format(topic)
        self.publishers[topic] = rospy.Publisher(topic_name, String,
    queue_size=1)
        print("publishing on topic ros_unity_{0}".format(topic))

    print("-------------------------------")

    # add other local variables
    self.current_link_dict = dict()
    self.tf_listener = tf.TransformListener()
```

Printing the list of all publishers was helpful for debugging the application.

The main processing in our modified version was performed in the `run()` function of the `UnityNode` class. The main loop in each case was given as follows:

```
def main():
    # *** setting up local variables ***

    # main loop
    while not rospy.is_shutdown():
        for link in link_dict:
            link_dict[link] = get_transform(link, tf_listener)
        pub_string = message_builder(link_dict)
        pub.publish(pub_string)
        rate.sleep()
```

```
def run(self):
    # *** setting up local variables ***

    # main loop
    while not rospy.is_shutdown():
        for topic in self.publishers.keys():
            if topic == "position":
                self._update_link_dict()
                message = self._build_position_msg()
                self.publishers[topic].publish(message)
        rate.sleep()
```

As we can see, the only significant difference between our code and the original version is the additional loop over each publisher in the `self.publishers` dictionary. Here we only have the `position` topic, but this could be extended to include as many topics as required. If we were transmitting information that required a subscriber, the subscriber would would be created in the `if` statement for the given topic.

As shown in this section, the changes we made to the code were not extensive; we moved functions into the `UnityNode` class, added functionality to create and keep track of any number of publishers, and defined where additional functionality could be added to process new streams of information. However, these changes had a significant impact on the extendability of the system. Indeed, the next section will demonstrate that having this scaffolding in place made adding publishers – e.g. a publisher for trajectory information – both significantly easier and more rapid.

The most important conceptual decision for extending communication was whether to publish each new stream of information on a separate topic, or to use the `ros_unity` topic for all information. We chose to add new topics for each publisher for the following reasons: firstly, if all information was published on the same topic, this would require a messaging format that can carry a large variety of different information, including information that was unseen at the time the messaging format was designed. As such, it would be unfeasible to use a custom messaging format. The alternative would be to use a standardised messaging format such as JSON; however, this would increase time for transmission, as the message would contain extraneous fields, and time for parsing, as we would have to extract the data field from the message.

The second reason for preferring this option was conceptual. We aimed to follow good design principles in our development and create the ROS and Unity sides such that they could operate independent of each other, such that one of them could be replaced or modified without requiring significant alterations in the other. If we used the same topic for all information, this would not allow the Unity side to opt out of receiving some streams of information, as it would always receive all messages on the `ros_unity` topic. Only once the message had been parsed and the contained information identified could the program determine whether the information was relevant to the application. This would result in significant wasted computation.

Based on these these reasons, we chose to publish each new stream of information separately on a new topic, and add new custom message formats if required.

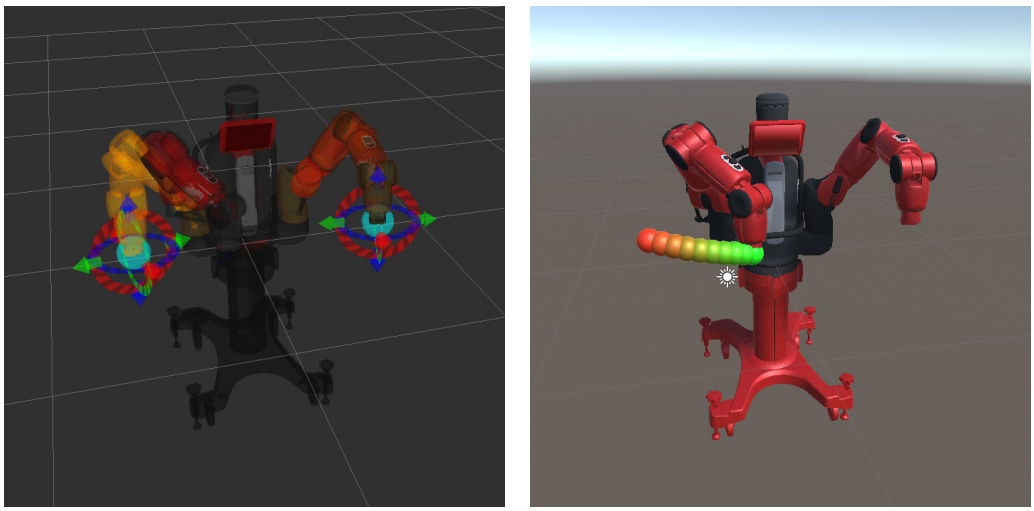
3.6 Visualising a trajectory

To verify that the functionality developed in Section 3.5 was valid, we selected a new stream of information to transmit and visualise. One suitable source for this was identified as a trajectory plan generated with the Moveit! path planning library in ROS.

Using the Moveit! GUI, it was easy to set a new pose for the robot's right arm. This generated a 10-step trajectory plan for the arm to move from the original position to the new position, which was published on the `move_group/diplay_planned_path` topic as a list of joint angles[2]. The trajectory could be simulated without moving the arm of the real robot. We added a new publisher for the `ros_unity_trajectory` topic; the process for this was simplified by using the framework developed in the previous section. On the ROS side we added the following code to the `unityNode.py` file:

- In the `main` function, we added `trajectory` to the list of topics.
- In the `run` function, we created a subscriber that listened for messages on the `move_group/diplay_planned_path` topic and called the `_got_trajectory` callback function every time the subscriber received a new message.
- We created the `_got_trajectory` callback function, which received the message containing a list of joint angles for each time step from the subscriber, performed forward kinematics to find the position of the robot's gripper in Cartesian coordinates at each time step, called the `_build_trajectory_msg` function to create a message string, and published the message on the `ros_unity_trajectory` topic using the publisher created in the `__init__` function.
- Finally, we also created the `_build_trajectory_msg` function, which used the same method as the `_build_position_msg` function to construct a custom message string. The only difference was that the trajectory message contained information about the position of only the right hand link for each of the 10 time steps, rather than the position of each link at one time step.

[2] Joint angles are sometimes used in robotics as an alternative to Cartesian x, y, z coordinates. The position of each link relative to the root of the robot can be derived using *forward kinematics*.



(a) Planning trajectory with Moveit! (b) Visualising trajectory with Unity

Figure 3.2: Visualisations of (a) how a path was planned with Moveit!, with the original position of the right arm given in red, and the new position given in orange, and (b) the path plan in Unity, with points along the trajectory indicating the position of the right hand at each time step.

We did not have to explicitly add a publisher for `ros_unity_trajectory`, as this was handled by the loop in the `__init__` function in the `unityNode` class.

On the Unity side, we could use the same functionality as for the position data to subscribe to the `ros_unity_trajectory` topic, receive and parse the message and extract the contents. As the contained information was different, we had to slightly adapt the process of parsing; however, the changes were trivial. We then added functionality to the `TFListener` script in Unity to visualise the trajectory. To achieve this, we created a new sphere GameObject for each point along the trajectory. To visually indicate the time steps, we set the colour of the spheres on a scale from *green* (time step 1) to *red* (time step 10). The resulting visualised trajectory for an example path plan, compared to the initial planned pose from Moveit!, can be seen in Figure 3.2.

As this step was only intended as *proof-of-concept* to show that the implemented extension of the communication bridge was successful, we did not spend much time on the design aspect. However, the code developed for this section could be used to create more elaborate trajectory visualisations if desired.

3.7 Setting up Meta 2

Once reliable communication had been established, the next step was to set up visualising information on the Meta 2 headset [see Section 2.4]. As explained in Section 3.1, the Meta was connected to the Windows PC with an HDMI and a USB 3 cable. The developer kit included tutorials on how to set up the headset and functionality to create calibration profiles for specific people. Once this had been completed, the headset could be used to create applications with Unity.

Figure 3.3: Real robot overlain with the virtual robot. The image was captured using the built-in camera on the headset and shows both the feed from the camera and the AR content.

To create an application for the Meta headset in Unity, the only requirement was to include the `MetaCameraRig` GameObject in the Unity scene. All Meta functionality was bundled into the `MetaCameraRig`; when the Unity scene was launched, the scripts associated with this object launched the SLAM tracking and the 3D objects in the Unity scene were visualised as augmented reality content.

To test the connection from the Baxter robot to Unity and from Unity to the Meta, we trialled visualising the current pose of the robot using the headset. To achieve this, we hard-coded the position of the visualised robot to be 1 metre in front of the headset. When standing in front of the real robot, we could then test through visual observation whether the displayed pose matched the real one. Indeed, as we can see from Figure 3.3, the visualised pose appears correct. We could also not detect any significant lag when the robot was moving, which indicates that the communication was functional.

3.8 Localisation

The fact that the virtual robot correctly reflected the pose of the real robot indicated that the pipeline was working robustly. However, hard-coding the robot position was not viable as a long-term solution, as the headset was initialised in a different position in Unity depending on where it was located in the environment when the program was started. As such, there was no way to ensure that the virtual robot was always correctly overlain; we required a method of localising the headset relative to a known point in the world to ensure that the visualisations were matched up with the real robot.

As explained in the Section 2.4, the Meta 2 by default uses SLAM tracking, which uses internal sensors rather than external indicators to determine the headset's location in space. We required a method of linking the position and rotation detection to a real-world object such as a *marker*, which would allow us to link the coordinate frames of the headset and robot and ensure that the visualisations were matched up with the real world correctly. We endeavoured to find a way to use external tracking, and in particular *marker tracking*, despite the fact that the code in the Meta's SDK included no obvious way to change the method of localisation.

3.8.1 Marker tracking

A common method of localisation in computer vision applications for augmented reality as well as robotics is tracking *fiducial markers*[3]. Markers traditionally used for localisation are black-and-white geometric patterns [19]. With advances in tracking technology, more advanced markers are possible; these include 3D models, product packaging, real-world surfaces such as floors and tables etc. However, as exploring these was outside the scope of this project, we only tested simple geometric markers. Two common[4] libraries for performing visual marker tracking are Vuforia and ARToolKit. We provide include an overview of both libraries:

Vuforia [34] is a high-level augmented reality library which has been integrated directly into the most recent releases of Unity (version 2017+). Vuforia incorporates a variety of state-of-the-art methods, including tracking 3D models and real surfaces, as well as markerless tracking. However, a developer licence is required for functionality more advanced than simple marker (*Image Target* in Vuforia) tracking. While some access to the code is available through Unity scripts, most of the software is proprietary.

ARToolKit [11] is another widely-used tracking library with support for Unity. It was originally created by Hirokazu Kato and published in 1999 [21]. ARToolKit is open-source and all code is available publicly on GitHub. As such, it has strong community support and is also a favourite for use in research applications. However, the functionality offered by ARToolKit is somewhat limited compared to Vuforia, with tracking available only for simple image markers and no extended tracking when the marker is outside the field-of-view of the camera.

While other tracking libraries exists, they are less mature and less well-documented (e.g. ALVAR [49]), or require a commercial licence to use (e.g. EasyAR [48], Wikitude [53]). As such, these were not considered in more detail.

3.8.1.1 Vuforia

For our initial exploration in to marker tracking, we used the Vuforia tracking library to test the viability of marker tracking for localisation of the Meta 2 headset.

[3] A fiducial marker is any point of reference in an image that provides information about the size, location, orientation etc. of objects in the image, e.g. a ruler. [9]

[4] In fact, both (!) claim to be the world's most widely used library for AR applications [34, 11].

(a) Vuforia (b) ARToolKit (c) ARTag

Figure 3.4: Default markers for (a) Vuforia and (b) ARToolKit, compared with a randomly-selected marker from the ARTag set (c).

Vuforia was easy to set up and use, firstly due to its integration with Unity, meaning that no external libraries had to be imported, and secondly due to comprehensive tutorials. To start, we used a default Vuforia marker, or *Image Target*, given in Figure 3.4a.

To run a simple AR scene with Vuforia, the only requirements were to include the `ARCamera` and `ImageTarget` GameObjects. Vuforia ImageTargets were stored on a database on the Vuforia server [34], and could then be downloaded and used as markers. Once the scene was run, Vuforia detected whether a camera feed was available (we used a webcam for initial testing, but for the real application the feed from the Meta headset could be used) and used input from the feed to detect the marker. Any children of the `ImageTarget` object were displayed when the marker was detected. Figure 3.5 shows a cube visualised on the marker from Figure 3.4a.

As we can see from Figure 3.4a, while default Vuforia markers are easily-identifiable patterns, they have no distinctive features to easily detect the orientation of the marker. For our application, it was essential that both the position and orientation of the marker were detected accurately so that the coordinate frames could be linked correctly. As such, we researched different sets of markers that were optimised for localisation tasks and designed such that orientation could be detected reliably.

Many different sets of visual markers exist (e.g. [14, 43, 19, 16]), often tied to specific tracking libraries (e.g. Alvar [49]), which are optimised to detect the given type of marker. After considering the available sets of markers, we decided to use ARTag [14] based on the fact that it has successfully been used for localisation in various augmented reality applications. This would lead us to assume that its stability and suitability for our application has been proven. We used a randomly-selected ARTag marker (number 66), which can be seen in Figure 3.4c.

When testing the Vuforia application with the ARTag marker, we found that detection was significantly worse than for the default marker in Figure 3.4a. Over multiple tests, it did not detect the marker at all, and in some cases it detected the orientation incorrectly. We can conclude that while Vuforia is reliable for tracking patterned surfaces, it is unreliable for correctly detecting the orientation of markers; as such, we conclude that it is not applicable to our case. However, we can see future uses for Vuforia if its more advanced features such as 3D models could be utilised.

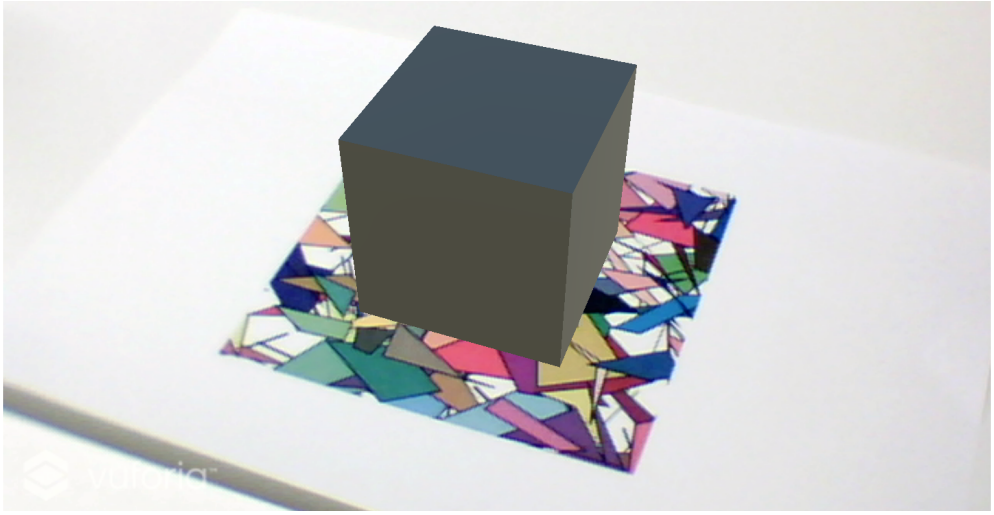


Figure 3.5: Augmented reality content visualised on a marker using Vuforia and Unity.

3.8.1.2 ARToolKit

As Vuforia proved to be unsuitable for use in our project, we explored the possibility of using the ARToolKit library for marker tracking. While it was not bundled into Unity, the Unity SDK was available on the ARToolKit website [11]. Similarly to Vuforia, setting up an ARToolKit scene in Unity was trivial when following the ARToolKit documentation. While ARToolKit included some default markers [see Figure 3.4b], we also tested it with the ARTag marker selected in the previous experiment. We found that when using the webcam, detection was reliable for both position and orientation.

We then attempted to use this library with the Meta headset. ARToolKit is intended to support see-through AR and includes functionality to generate calibration files for AR headsets. However, we found that the tools were dated and meant for older headsets, where each eye had a separate display; as such, they were not compatible with the Meta, in which the display was shared between both eyes. This incompatibility meant that we could not use the targeted AR functionality, which used the calibration files to display augmented reality content correctly.

To solve this problem, we attempted to implement similar functionality without the use of targeted AR tools. However, we found that the fundamental way in which augmented reality was implemented in ARToolKit became an obstacle; for non-see-through AR, i.e. 2D displays such as when using the webcam, ARToolKit implements augmented reality by projecting the video feed onto a canvas in Unity and placing the 3D content on the canvas. When viewing the 2D display, the augmented reality appears correct. However, the objects are not situated in the Unity scene as we would require for visualisations with the Meta. To implement augmented reality in a way that was compatible with the Meta, we had to perform the transformation from the canvas to the real world, and secondly invert the transformation between the marker and the camera. However, even though the process was known, we could not perform it correctly, as the inversion of the transformation between the camera and the marker was incorrect, giving the correct rotation for the marker but not the correct position.

We identified that the cause of the problem were operations that were being performed in low-level compiled code, which we did not cave access to to change the procedure. This meant that the transformation matrix was not directly invertible. Due to time constraints on the project, we decided not to spend additional time to attempt a work-around by changing the compiled code.

Based on the work performed in this section, we can conclude that for now, using Vuforia is not viable for precise tasks where detecting the orientation of the marker is important. Additionally, we can conclude that there is currently no support for integrating ARToolKit with newer see-through augmented reality headsets such as the Meta. However, it is possible that this might change, as a new version of ARToolKit (ARToolKit 6) is currently in beta release. As such, more support for newer augmented reality hardware such as the Meta could be included in the new release.

3.8.2 LED marker tracking

As we showed in Section 3.8.1, using simple visual marker tracking with Meta did not prove to be viable for this project. As such, we instead used an alternative tracking system, the Visualeyez professional tracking system with LED markers. We credit Joshua Smith for his work in setting up the tracking system. While the work of setting up the camera and tracking software was outside the scope of this project, we will endeavour to provide sufficient context for the reader.

First, the tracking system consisted of three cameras mounted on a tripod, which were used to track two *LED markers*. The markers were placed in 3D-printed mounts and attached to the headset and the object [see Figure 3.6]. The unique frequency and configuration of lights is used to identify each marker and determine its position and rotation. The system was calibrated such that all coordinates were given relative to the root of the robot, similarly to coordinates in both ROS and Unity. The position and rotation of both the headset and the object were transmitted continuously over websockets on different ports. The coordinates of the object were read by the collaboration task code in ROS and the coordinates of the headset were read by the `WebsocketClient` in Unity. As the coordinates were already relative to the root of the robot, no additional transformations were required in ROS and the only alteration that was required in Unity was to convert the values from a right-hand to a left-hand coordinate system.

The most significant challenge for implementing tracking was in how to update the position and rotation of the headset in Unity. In the code, this was integrated into the classes that also handled SLAM tracking, which meant that we couldn't easily switch out the localisation to get the position and rotation from LED tracking. In fact, while using custom localisers had been supported in previous versions of the Meta SDK, this functionality has been deprecated in recent releases. To tackle this problem, we were able to use the legacy `MouseLocalizer` class, which had been used in previous versions of the Meta SDK to move the headset using the mouse in Unity. We adapted this code by replacing the coordinates with those received from the websocket and replacing the `SLAMLocalizer` with the `MouseLocalizer`.

(a) Object for handovers.

(b) Meta 2 headset.

Figure 3.6: The object and the Meta 2 headset with LED markers attached.

However, we found that this also resulted in the cameras for the left and right eyes facing in the opposite direction to the rest of the headset; we identified that SLAM localisation performed additional transformations in proprietary code. As we did not have access to the exact operations that were being performed, we implemented a best-guess method by rotating the left and right eye cameras by 180 degrees. Since we have no means of checking the exact operations performed by the SLAM localisation short of de-compiling the source code, we can merely hypothesise that the procedure was in large part correct based on visual observation of the displayed visualisations.

While this tracking system was more complicated to set up, it also had significant benefits: firstly, the tracking was more stable than with image markers. Secondly, this meant that a separate tracking system was not required for the object in the handover task, as LED markers could be used to track both the object and the headset. However, the most significant negative was that the requirement that markers always be visible to the camera meant that freedom of movement for participants was significantly reduced.

3.9 Visualising for collaboration task

The collaboration task to be visualised is described in detail in Section 1.4. While the task was originally intended for two-arm interaction, we found that due to the setup of the tracking system, we had to modify the task to use only the left arm, as otherwise the LED marker on the object was often obscured by the right arm. We would like to credit Joshua Smith for setting up the code for the collaboration task. In this section, we will describe how we selected aspects of the task to create visualisations for and how these visualisations were implemented.

3.9.1 Selecting topics to visualise

To identify the aspects of the collaboration task to visualise, we first examined the sources of information that were available to us. The nodes responsible for the collaboration task published information on the following topics:

- *Tracked object* – this represented the position of the real object relative to the robot, as tracked by the tracking system.
- *Gripper target* – this represented the target position of the gripper.
- *Virtual object* – this represented the system's internal representation of the object. While the object was not gripped, this was equivalent to the tracked object. However, once the robot gripped the object, the virtual object followed the gripper position, and was no longer dependent on the tracking.
- *Gripping point on object* – this represented the point on the object that the robot was planning to grip. In other applications, especially two-arm applications where it was important that the arms not collide, the point could be configured to differ from the actual object position. However, in our case this was always equal to the real object position.
- *Intercept point* – this represented the point where the robot predicted the object would intersect with the robot's workspace, i.e. the area in front of the robot where the robot was configured to grip the object. In our case this was defined 80cm in front of the robot.

We could eliminate some of the above aspects that were duplicating information; specifically, we identified that the virtual object was already represented by a combination of the tracked object and the gripper target. Similarly, since the gripping point on the object was always the point where the object was being tracked, it did not encode new information and could be eliminated. We were left with the following list:

- *Tracked object*
- *Gripper target*
- *Intercept point*

For each of these aspects, we could identify the potential benefits they would provide if the visualisation was present while performing the handover task.

Firstly, we hypothesised that visualising the tracked object would enable the human to easily identify when the object was not being tracked, indicating that the handover would be unsuccessful as the robot could not see the object. This would allow the human to correct for tracking errors, either by moving/tilting the object if the marker was being obscured, or by restarting the tracking system if the tracking failed.

Secondly, we hypothesised that visualising the gripper target would clearly visualise when the robot's side of the handover procedure was initiated and it moved to grip the object, as well as indicating when the object was moved back far enough that the robot retracted the arm to default position. It could also be indicative of when the arm got stuck, especially in cases when arm was not moving but the gripper target indicated that it was planning to do so. The human could also see whether the robot was going to grip or whether the object was too far away if there was a delay.

Thirdly, we hypothesised that visualising the intercept point could help indicate the best place to pass the object to maximise probability of success, since it represents the place that the robot expects the object to go, and as a result would not have to perform any recalculations. Additionally, similarly to the gripper target, the intercept point could be used to ascertain whether the robot was not moving because of a delay or because the object was not aligned with the intercept point.

3.9.2 Creating visualisations

Once we had identified the information to visualise, we set up the communication from the specified topics to Unity. Adding new streams of information to the `unityNode` node was simple when using the framework set up in Section 3.5.

The messages for each of the three visualisations contained a single position and rotation relative to the root of the robot. These were transmitted on the following topics:

- `/ros_unity_tracked_object`
- `/ros_unity_gripper_target`
- `/ros_unity_intercept_target`

The information was then received on the Unity side and converted from the right-hand coordinate system to the left-hand coordinate system.

As we aimed to create simple visualisations we chose to represent each source of information as a simple sphere. The position of the spheres was updated each frame; however, using spheres meant that we did not have to update the rotation. To ensure that the three visualisations were distinguishable, we chose three complimentary but distinct colors, *yellow*, *green* and *white*, to represent the *object*, *gripper target* and *intercept target* respectively. All three streams of information could then be visualised while conducting the handovers. Figure 3.7 illustrates how the three spheres were visualised in different locations over the course of a handover.

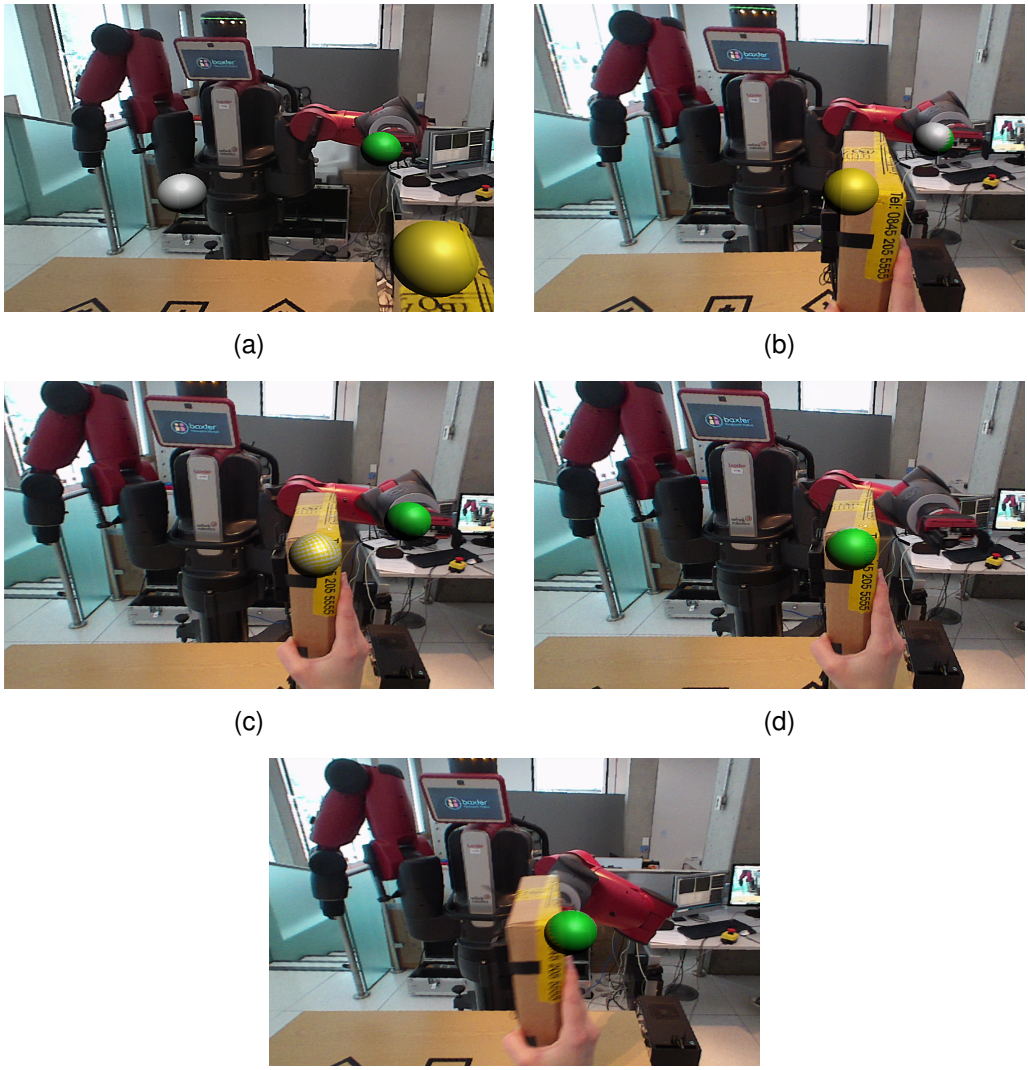
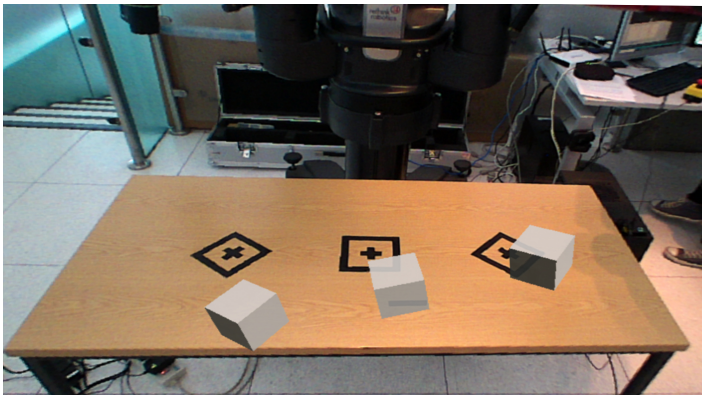


Figure 3.7: Visualisations over the course of a handover. We can see that the visualisations help to illustrate the step-by-step process of the handover from the robot's side. In (a), the object is too far away from the robot and not moving. As such, the intercept point is defined in the middle of the workspace, the gripper is in default position and the object is being tracked. In (b), the object begins to move closer, and the intercept point is initially set at the same location as the gripper (the white and green spheres overlap). In (c) the object moves closer and the robot can make a better guess of where the object intersects the workspace. We can see that the object is already on the border of the workspace (white and yellow spheres overlap). However, it is not close enough yet for the arm's task to switch to gripping (the green sphere has not moved). In (d) we can see that the gripper target has switched to overlap with the yellow and white spheres. This indicates that the arm is about to move. Finally, in (e), we can see that the arm has moved and the handover was successful.

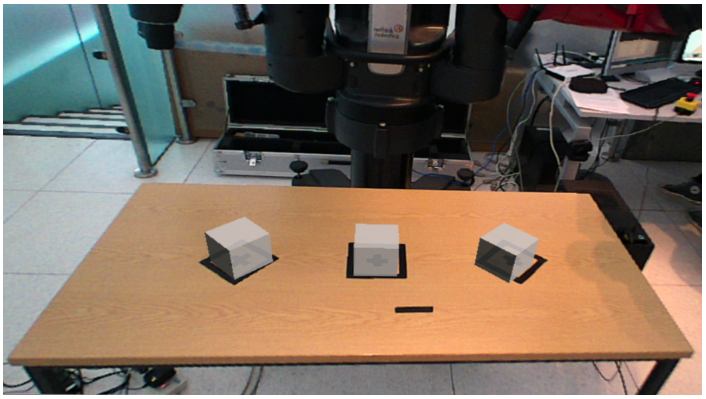
3.10 Adding calibration

During testing of the visualisations we created in the previous section, it became apparent that the position and rotation of the headset was often misaligned with the real world. We identified the cause of this as the transformation from the marker attached on the side of the headset, to the centre of the headset, which was located behind and above the eyes. We found that this was dependent on how the headset was fitted on the wearer's head and whether the markers were jostled during the process. As such, we chose to to fix this issue by introducing an additional calibration procedure.

For accurate calibration, we required a location in the real world which we could measure precisely to the root of the robot. We chose to utilise the table in front of the Baxter robot. To perform calibration, we marked three 10x10cm squares on the table, one in front of the root with no rotation, and two 30cm to the left and right of the middle square, rotated at 45 and -45 degrees respectively. We added three cubes in Unity at the same positions and rotations relative to the root of the virtual robot. We could then change the x, y and z position and rotation of the headset to match up the cubes and the squares; since we knew the exact location of the squares in the real world and the cubes in Unity, we could be sure that if the two were lined up, the visualisations would be correct. Figure 3.8 shows the cubes before and after calibration.



(a) Before calibration



(b) After calibration

Figure 3.8: Cubes used for calibration before and after the calibration procedure was completed.

Chapter 4

Evaluation

Once the implementation [see Chapter 3] was complete, an experimental user study was conducted to evaluate the developed interface. This chapter will outline the process of designing and conducting the study, introduce the objective and subjective evaluation metrics, and present an analysis of the achieved results.

4.1 Experiment design

The experiment was conducted using the collaborative handover task provided by EPFL LASA [see Section 1.4] with the Baxter robot. The interface was created for the Meta 2 augmented reality headset, with LED marker tracking on the headset and object [see Section 3.8.2]. We used the visualisations described in Section 3.9, representing the *gripper target*, the *object* and the *intercept point*.

To evaluate the effect of the interface on task performance, we decided to conduct a within-subjects experiment with two interfaces: with and without visualisations. Initially, we intended to conduct within-subjects experiments for each individual visualisation (*gripper target*, *object* and *intercept target*). However, after testing the visualisations, we determined that there were important aspects of the interface that were only available if multiple visualisations were present – for example, a reliable indication that the handover would be successful was if all three visualisations converged to the same position [see Figure 3.7]; this information would not be available if each visualisation was tested individually. Additionally, constraints on time and number of participants meant that we could not gather sufficient data if the experiment was run for every combination of visualisations; this limited our choice to either testing each visualisation individually or all of them together.

Through extensive testing, we additionally determined that due to random variance in the robot's behaviour, each participant would have to perform multiple handovers per interface to accurately measure average time. To ensure that the experiment would remain within reasonable time limits (~30 minutes), and for the reasons described above, we chose to test the interface with all visualisations present.

Initially, we only planned to calculate the average time for "simple" handovers, i.e. the participant handing the object to the robot in the proximity of the gripper's resting position. However, through testing it became clear that this case was not sufficiently challenging, as it could be completed fast (given no tracking issues) either with or without visualisations. As such, to adequately utilise the hypothesised benefits provided by the visualisations, we added a "complex" handover case, in which we would ask the participant to pass the object to a new position further away from the gripper, either higher, lower, to the left or to the right.

To allow for sufficient data, we decided to have each participant perform 6 simple handovers, followed by 4 complex handovers (10 handovers in total) for each interface. The order in which the participants received the interfaces was randomised to eliminate bias; as all participants were unfamiliar with the interface and the task, we anticipated that there would be a *learning effect*[1] which would affect the final averages.

To ensure consistency in results, the participants started the handover from the same position every time, which was measured at the limit of the tracking system and marked on the floor with tape. To measure the *time per handover*, the timer was started when the participant was told to initiate the handover – to approach the robot and hand the object to the robot – and the timer was stopped when the gripper closed. Additionally, to ensure that results for both interfaces were comparable, the participant wore the headset at all times, regardless of the order in which they received interfaces. This ensured that the effects of wearing the headset – such as discomfort and limited freedom of movement – remained constant between experiments.

Figure 4.1 shows the full setup of the experiment.

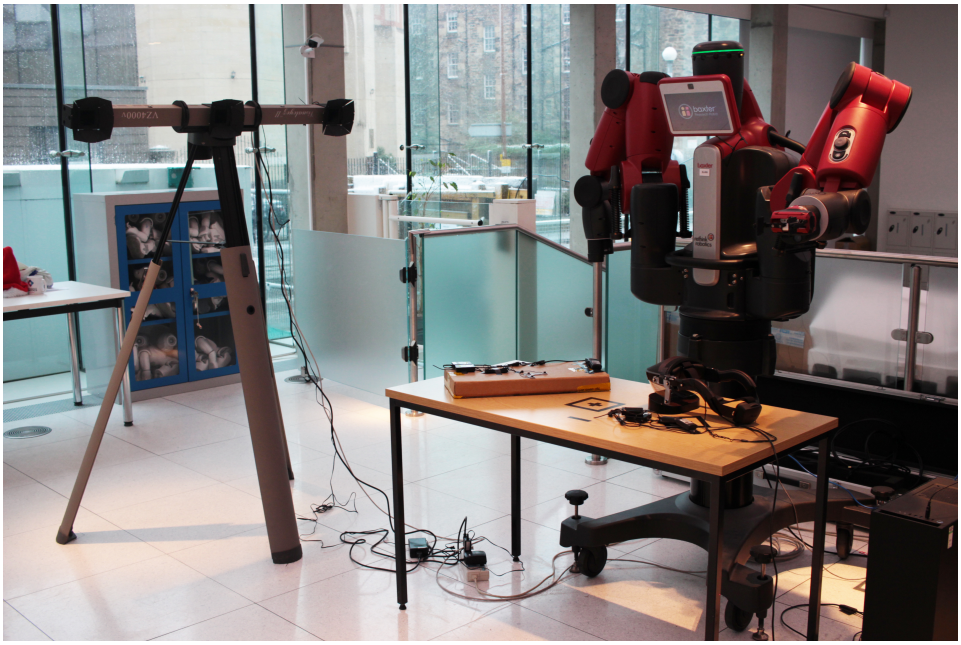


Figure 4.1: Experiment setup with tracking system (left), Baxter robot (right), and object and Meta 2 headset (on table).

[1]Meaning that we would expect the second interface to perform better than the first to some degree.

4.2 Evaluation metrics

To adequately evaluate the performance of the designed system, we selected three different metrics to measure different aspects of the system. First, we used an objective numerical method, which was based on measuring the *average time per handover* for both the simple and the complex case. Secondly, we used two subjective rating scales: *NASA Task Load Index* and *User Experience Questionnaire*. Section 4.2.1 and Section 4.2.2 will introduce these scales in detail. Finally, we included one question in the exit interview, in which the participants were asked to choose which interface they would prefer if asked to perform more handovers.

4.2.1 NASA Task Load Index (NASA-TLX)

The NASA Task Load Index (NASA-TLX) was developed by the *Human Performance Group* at the National Aeronautics and Space Administration (NASA) and published by Sandra Hart and Lowell Staveland in 1988 [18].

It is a commonly-used metric[2] for measuring subjective workload of human operators in human-machine systems. While it was originally applied to the space and aviation industries [55, 31, 42], it has since also been used in various domains such as healthcare [56, 50] and robotics [22, 13, 41]. Most importantly, as far as this project is concerned, it has been used to evaluate augmented reality interfaces for human-robot systems [5, 37]. Analyses have shown that for many tasks, there is a high correlation between the NASA-TLX score and objective performance [47].

The six dimensions measured by NASA-TLX are defined as follows:

- *Mental demand*
  How mentally demanding was the task?
- *Physical demand*
  How physically demanding was the task?
- *Temporal demand*
  How hurried or rushed was the task?
- *Performance*
  How successful were you in accomplishing what you were asked to do?
- *Effort*
  How hard did you have to work to accomplish your level of performance?
- *Frustration*
  How insecure, discouraged, irritated, stressed and annoyed were you?

Each scale is accompanied by a short (standardised) description, which gives additional context and has been shown to aid participants in providing accurate ratings [18].

[2] Google Scholar returns 15,800 results for "NASA-TLX", including 351 publications in 2018 alone [at time of writing, 2-April-2018].

The participants rated mental, physical and temporal demand, effort and frustration on a scale from 0 (*very low*) to 20 (*very high*). Perceived performance was rated on a scale from 0 (*failure*) to 20 (*perfect*).

In the original publication [18], Hart and Staveland included an additional step, in which a weighted workload score was calculated by asking the participants to evaluate the importance of each dimension in the workload estimate by comparing dimensions pairwise. However, as it is common to omit this step and use the raw scores for analysis [17], we chose to not perform any weighting procedures or calculate an overall score.

The form that was used in the experiment can be seen in Appendix C.

4.2.2 User Experience Questionnaire (UEQ)

The User Experience Questionnaire (UEQ) is a subjective user experience scale. It was originally developed in German by Bettina Laugwitz, Theo Held and Martin Schepp in 2006 [25] and released in English in 2008 [24]. The full version of the UEQ is a 26-item questionnaire that measures six factors: *attractiveness*, *perspicuity*, *efficiency*, *dependability*, *stimulation*, and *novelty*. However, as this scale measures aspects that our analysis is not concerned with, such as *stimulation* and *novelty*, we selected the 6 most relevant scales out of the full 26 from UEQ to conduct our analysis, similarly to the official shortened version of UEQ (UEQ-S) [39]. We did not calculate a combined score, but rather used the raw results for the selected subset, which consisted of the following scales:

- *Obstructive – Supportive*
- *Complicated – Easy*
- *Pleasant – Unpleasant*
- *Clear – Confusing*
- *Annoying – Enjoyable*
- *Friendly – Unfriendly*

Each aspect was rated on a 7-point scale. The form that was presented to participants during the experiment can be seen in Appendix D.

4.3 Expected results

Before conducting the experiment, we expanded on the hypothesis set in Section 1.5 to create metric-specific hypotheses.

For the performance (*time per task*), we hypothesised that there would be an improvement (lower average time) with both the simple and complex handovers, but we hypothesised that the visualisations would be utilised more for the complex case, and as such the improvement would be more significant.

For NASA-TLX, we expected no change in the *physical* and *temporal demand*, as we did not anticipate that these aspects would change between the two interfaces. Based on the results shown by Rosen *et al*. [37], we hypothesised that we would see an increase in the *mental demand*. We expected to see lower *effort* and *frustration*, as we hypothesised that the task would be easier to complete with the help of visualisations. Finally, we expected the rating for perceived *performance* to be higher for the interface that the participant got second, as they would more confident and familiar with the task.

For UEQ, we hypothesised that the interface with visualisations would be rated as more *supportive*, *pleasant*, *clear*, *enjoyable* and *friendly*, but that the participants would also rate it as more *complicated*, as there would be more information to consider.

In Section 4.5 we will compare the achieved results to the expected values given above and provide an analysis where the results did not confirm our hypotheses.

4.4 Conducting experiment

The experimental study was conducted in the Inspace robotics lab at the University of Edinburgh. 14 participants were recruited for the final study, all of whom successfully completed the experiment. Each participant conducted the experiment individually, with no other participants present to avoid introducing a bias. The design of the study is outlined in Section 4.1. The following section will describe the step-by-step process that was followed to conduct the experiment.

As all participants were unfamiliar with the robotics lab, they were first introduced to the lab and the Baxter robot. They were then presented with the *information sheet* [see Appendix A] and asked to read the contents carefully. The participant could then ask questions, if they had any, to clarify the contents of the information sheet; however, we avoided any questions that could bias the participant for the experiment, such as explaining exactly what the goal of the study was. The participants then signed the *consent form* [see Appendix B]. Before starting the study, we also conducted a safety briefing on how to interact with the Baxter robot to avoid injuries.

The first step in conducting the experiment was to fit the participant with the Meta 2 headset and perform the eye calibration procedure from Meta. This created an individualised calibration profile, which enabled us to ensure that the visualisations were displayed correctly relative to the participant's eyes.

We then introduced the handover task, presented the participant with the object [see Figure 3.6a] and explained that the object and headset had to be held in a specific way to ensure that the LED markers were visible to the tracking system. The participant was allowed to perform a few test handovers to ensure that they were comfortable with where to stand and how to release the gripper to take the object from the robot.

The participant was then presented with the first interface; whether the first interface was with or without visualisations was assigned randomly. Out of the 14 participants who completed the experiment, 6 received the interface with visualisations first and 8 received the interface without visualisations first.

(a) Participant initiates handover

(b) Participant passes object to Baxter

(c) Baxter grabs object

Figure 4.2: Illustration of the handover process from initiating the handover to when the Baxter robot grabs the object; we timed this process from start to finish. Photos are illustrative and included with the participant's permission.

For the interface with visualisations, an additional calibration step was conducted to ensure that the visualisations were aligned correctly with the real world [see Section 3.9]. The participant also got a short introduction into the three visualisations (*gripper target*, *object* and *intercept target*); however, we included as little detail about the task specifics as possible to ensure that the group who got this interface did not have an advantage over the group who got the interface without visualisations.

We then began the experiment: the participant conducted six simple and four complex handovers [see Figure 4.2 for an example of the simple handover]; each handover was timed from the moment the participant initiated a handover to when the robot gripped the object. We did not aid the participant if the robot took longer than expected to respond or if the participant had to move the object slightly to make the robot grip. However, we restarted the handover in case of technical faults, e.g. if the robot jammed due to sub-optimal inverse kinematics or the tracking system failed.

Once the handovers had been completed, the participant was asked to evaluate the interface on the NASA-TLX [see Section 4.2.1] and UEQ [see Section 4.2.2] scales, as well as provide free-form comments on their experience.

We then performed the experiment with the second interface following the same procedure, and asked the participant to rate that interface using the same method. Finally, we included one exit question, in which the participant was asked to select which interface they would prefer to use if they were asked to perform more handovers.

4.5 Results

The experiment was conducted with 14 participants. The participants were split 8/6 female/male. All participants were students between 20 and 25 years of age with no previous experience with collaborative human-robot interaction.

The results for performance (time-per-handover) will be presented in Section 4.5.1. Results for NASA-TLX and UEQ will be presented in Section 4.5.2 and Section 4.5.3 respectively. We will also provide interpretations and analysis for all achieved results.

4.5.1 Performance

The achieved average times for each participant for both interfaces are given in Table 4.1 for the simple handovers and in Table 4.2 for the complex handovers.

The average time differs significantly between participants for both the simple and complex case, mostly due to variance in the resting position of the robot's arm and the exact way in which the participant passed the object. As such, our analysis will focus on the difference between average times for both interfaces *within* subjects. In particular, we will investigate whether the average time improved when using the interface with visualisations. We will also investigate the extent to which the order in which the participants received the interfaces influenced the results.

	Simple handover		
ID	No visualisations	Visualisations	Improvement
1	6.950s	7.128s	−0.178s
2	7.365s	5.810s	1.555*s*
3	6.365s	6.346s	0.019s
4	6.335s	7.292s	−0.957s
5	9.103s	6.298s	2.805s
6	4.125s	4.090s	0.035s
7	5.285s	7.363s	−2.078s
8	5.476s	5.757s	−0.281*s*
9	5.333s	5.247s	0.086s
10	3.894s	4.100s	−0.206s
11	4.190s	5.620s	−1.430s
12	4.552s	4.117s	0.435s
13	6.873s	7.158s	−0.285*s*
14	4.725s	4.137s	0.615s
Average	5.757s	5.747s	0.010s

Table 4.1: Average time per handover for the simple handover for each participant with both interfaces, and the average improvement when using the interface with visualisations. Participants who received the interface with visualisations first are given with a shaded background.

	Complex handover		
ID	No visualisations	Visualisations	Improvement
1	7.605s	6.805s	0.800s
2	9.105s	10.363s	−1.258*s*
3	7.343s	6.437s	0.906*s*
4	5.540s	7.093s	−1.553s
5	12.235s	6.960s	5.275s
6	6.115s	7.225s	−1.109*s*
7	5.150s	4.430s	0.720s
8	7.060s	6.035s	1.025*s*
9	8.088s	7.948s	0.140s
10	7.918s	6.495s	1.423s
11	7.270s	4.880s	2.390*s*
12	8.035s	5.758s	2.277s
13	10.263s	8.690s	1.573*s*
14	6.948s	6.820s	0.128s
Average	7.763s	6.853s	0.900

Table 4.2: Average time per handover for the complex handover for each participant with both interfaces, and the average improvement when using the interface with visualisations. Participants who received the interface with visualisations first are given with a shaded background.

Table 4.1 and Table 4.2 show the average improvement in performance for the interface with visualisations for simple and complex handovers. As we can see, while the average improvement for the simple handovers was very small (0.01s), the average improvement for the complex handovers was 90 times larger at 0.9s.

To evaluate whether the improvement was statistically significant, we performed a *paired t-test* for both the simple and the complex cases. This indicated that while there was no statistically significant improvement for the simple case, for the complex case there was statistically significant improvement between the two interfaces in the direction of the interface with visualisations with 97% confidence (*p-value* 0.03). From this we can conclude that the first part of the hypothesis from Section 4.3 for *time per handover* – that there would be an improvement for the simple case – was not supported, but the second part – that there would be an improvement for the complex case – was supported. This also supports our original hypothesis from Section 1.5, as we have shown an increase in performance for the interface with visualisations.

We can find additional support for this hypothesis by separately examining the group that got the interface with visualisations as their first interface [rows with shaded background in Table 4.1 and Table 4.2]. While we expected the *learning effect* – in short, that the participant would always perform better on average with the second interface, as they are more familiar with the task – to be spread evenly between the two interfaces when taking the overall average, examining the groups separately will further allow us to analyse the effect of the visualisations.

We observe that the learning effect was significant for the simple handover: the average improvement for the group that got the interface without visualisations first was 0.06525s (significantly above the overall average improvement), while for the group that got the interface with visualisations first, the average was −0.0645s (significantly below the overall average improvement). This tells us that the visualisations were not effective for the simple case if the participants were already familiar with the interface. However, in addition to the learning effect, we can also observe a *training effect*: the participants performed significantly above average without visualisations if they had previously experienced the interface with visualisations. We support this with a quote from one of the participants:

> *The behaviour I learned using the visualisations helped me perform better, even without them.*

In the complex case, the same trend was evident, but the effect was less pronounced: the average improvement for the group that got the interface without visualisations first was 1.15125s, while the average for the other group was 0.58783s. We can attribute this to the fact that both the learning and training effects were reduced as the participants already had experience from the simple handover.

Overall, we can conclude that the empirical results support our hypothesis that visualisations would improve performance for complex handovers. Additionally, for both simple and complex handovers, participants performed better without visualisations if they had previous experience with visualisations; this indicates a possible application for the visualisations – training – which will be discussed further in Chapter 5.

4.5.2 Results for NASA-TLX

The first subjective scale that the participants rated the interface on was NASA-TLX [see Section 4.2.1]. This section will outline the results achieved on this scale and compare the achieved results to the expected results given in Section 4.3.

The average ratings for each subscale (perceived *mental*, *physical* and *temporal demand*, *performance*, *effort* and *frustration*), over all participants for the interface with no visualisations and the interface with visualisations are given in Figure 4.3.

4.5.2.1 Mental demand

The first measured scale was the perceived mental demand. We can see that the average score for the interface with visualisations is significantly higher than for the interface with no visualisations. This is in line with the expected results as given in Section 4.3 and with the results achieved by Rosen *et al*. [37] on a similar task. We can explain the result by considering the fact that more information is available to the participant when using this interface, which they have to spend more mental effort to parse.

4.5.2.2 Physical demand

The second measured scale was physical demand. While we predicted no difference between the two interfaces, the results indicate a slight increase with visualisations. We attempt to provide an explanation: while the Meta 2 is intended to be safe for use with glasses [see Section 2.4], from participants' comments it became clear that people with glasses found the interface more uncomfortable to use and experienced physical discomfort when viewing the visualisations. As such, we can hypothesise that this lead them to rate the interface as more physically demanding.

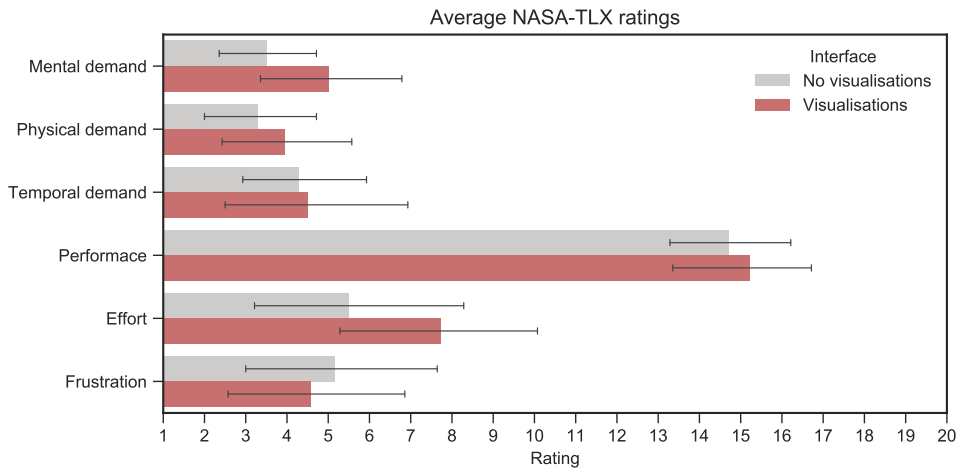


Figure 4.3: Average ratings with error bars indicating uncertainty in the average rating for each subscale of NASA-TLX, with the interfaces with and without visualisations.

4.5.2.3 Temporal demand

The third scale measured perceived temporal demand. While the mean and error was slightly higher for the interface with visualisations, the difference was not significant, which is in line with the expected results as given in Section 4.3.

4.5.2.4 Performance

For perceived performance, we expected to see a general increase in results for the second interface, as we predicted that the participants would feel more confident and rate the interface higher as a result. We can see that overall, the average rating was slightly higher for the interface with visualisations. However, we can provide more accurate analysis by examining the ratings for each participant individually. The scores are given in Figure 4.4. As we can see, regardless of which interface the participants received first, they generally rated their performance with the second interface higher. We confirm our hypothesis by running a paired t-test, which shows that the improvement is statistically significant with 95% confidence.

4.5.2.5 Effort

The most significant difference between the two interfaces, as well as between achieved results and the expected results from Section 4.3, was in the results for perceived effort. We can see that the rating for effort was significantly higher for the interface with visualisation, while we predicted that the opposite would be true.

We can hypothesise that this metric is tied to *mental demand*, which also showed an increase with visualisations. We attribute this to the fact that participants had to think about the task more than with the other interface, where less information was available.

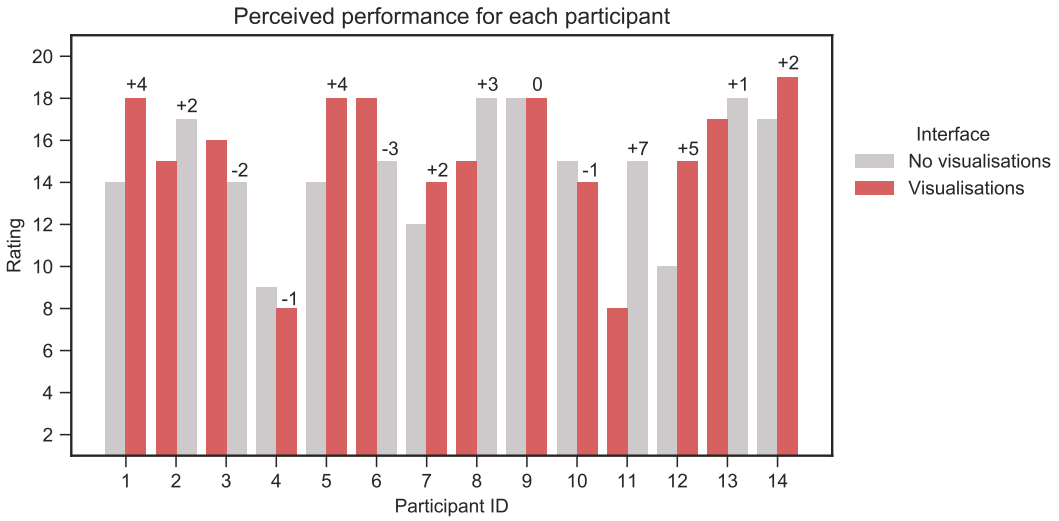


Figure 4.4: Perceived performance for each participant with the first and the second interface (either with or without visualisations), and the difference between the second and first rating.

Additionally, we can use comments from the participants to further analyse the results. Multiple participants reported that they both felt compelled to and were able to help the robot more in completing the handover when using the interface with visualisations. For example, one participant reported that without visualisations,

> *"...there were more times when [they were] waiting for the robot to grab the item, whereas with the visualisations [they] could help it more".*

Another participant described the experience as follows:

> *The [interface with spheres] was a bit more 'hostile' in the sense of making you feel like there is something to accomplish with them, when they are actually there for information purposes.*

These comments, as well as remarks from multiple other participants who reported that they felt they were most successful with visualisations when they aimed to match up the *object* and *intercept target* spheres, lead us to hypothesise that the participants approached the task itself differently with the two interfaces. Without visualisations, many participants approached the task more *passively*, expecting the robot to do most of the work, since the participant had less information to use to engage with the handover. However, with visualisations, the participants were more *active* in engaging with the task and used the visualisations as a "target" to complete the handover.

This is a fundamental difference in how the participants perceived the task, and we can conclude that while this may have contributed to a significantly higher perceived effort, the visualisations also allowed the participants to approach the task as it had been intended, i.e. an active collaboration on both the robot and human side.

4.5.2.6 Frustration

For the final scale, frustration, we observe a lower average for the interface with visualisations, which is in line with the expected result from Section 4.3. This implies that the participants felt less *insecure*, *discouraged*, *irritated*, *stressed* and *annoyed* [see Appendix C]. However, the difference is less significant than we would have hoped.

Multiple participants reported feeling less insecure with visualisations, for example:

> *[With visualisations] you can predict where the robot will move, so there is no "I'm threatened by unpredictable machine" effect.*

However, some participants also reported frustration with the discomfort from the headset and the visualisations, as well as with jitter from tracking and mis-aligned visualisations. As such, we can hypothesise that these factors drove up the frustration score and obscured the benefit that we predicted would stem from having more information available to complete the task.

We hypothesise that a more significant decrease in frustration could be achieved with visualisations; however, to mitigate adverse factors and measure decrease in frustration more accurately, future experiments could be performed with a more stable tracking system and better calibration, as well as a more comfortable headset.

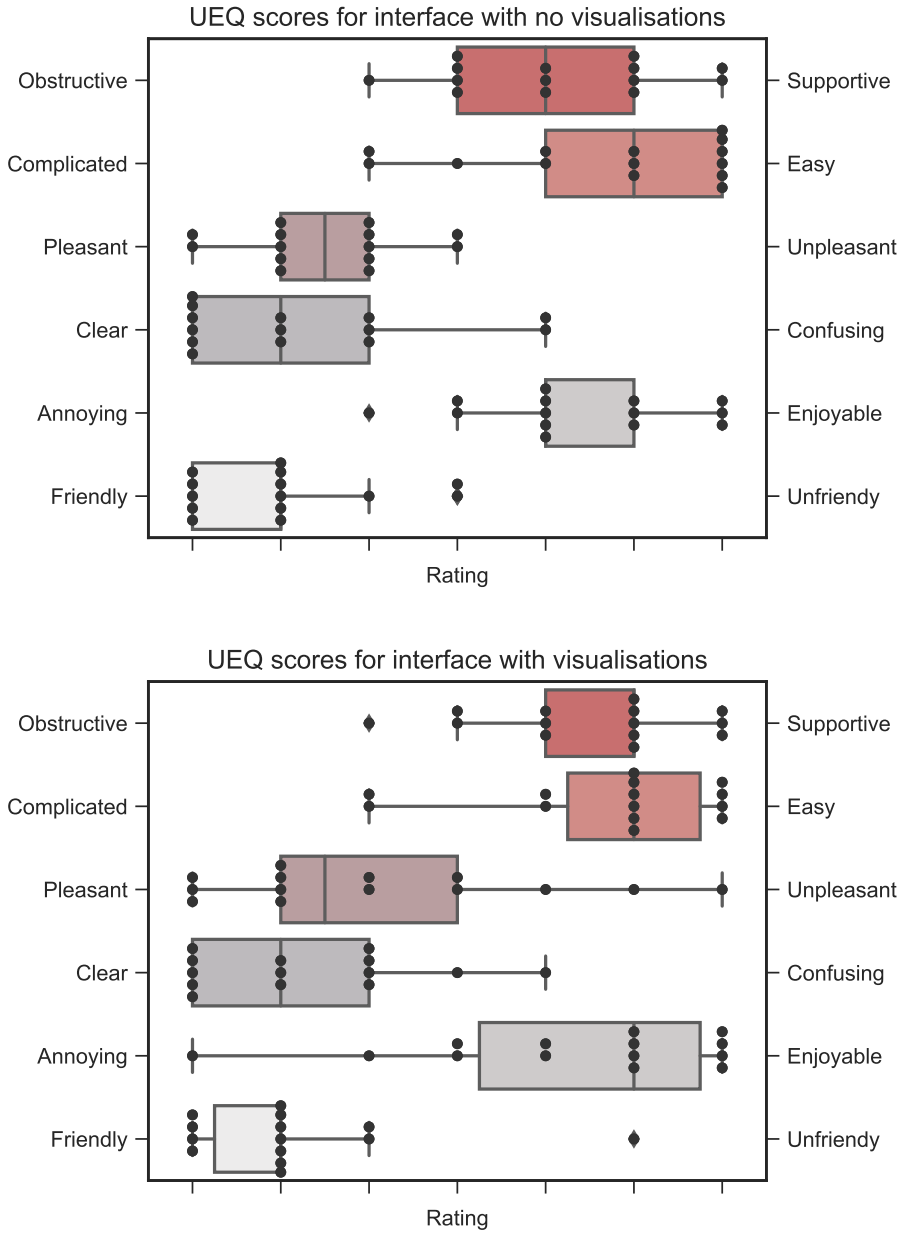


Figure 4.5: Scores for each sub-scale of the shortened version of the User Experience Questionnaire for both interfaces: without visualisations (top) and with visualisations (bottom). The colours represent the different sub-scales.

4.5.3 Results for UEQ

The second subjective metric used in our evaluation was the User Experience Questionnaire [see Section 4.2.2]. Figure 4.5 shows the achieved results for each subscale for both interfaces. This section will examine the results for each subscale individually and compare the results with the hypotheses from Section 4.3.

4.5.3.1 Obstructive-Supportive

The first measured scale was *obstructive-supportive*. We can see that the results for this scale experience a shift towards supportive for the interface with visualisations. This is in line with the expected results from Section 4.3.

4.5.3.2 Complicated-Easy

The second measured scale was *complicated-easy*. We can see that for this scale, there is a clear shift in the results to indicate that the interface without visualisations was perceived as easier than the one with visualisations. This is in line with what we would expect, as there is less information to consider with this interface; in fact, it is as simple an interface as could be concieved of.

4.5.3.3 Pleasant-Unpleasant

The third scale was *pleasant-unpleasant*. The results for this scale are interesting, as they show that the interface with visualisations was significantly more divisive than the interface without visualisations – the results for the interface with no visualisations are clustered at 2 and 3 (mostly pleasant), while the results are significantly more spread out for the interface with visualisations, varying from 1 (pleasant) to 7 (unpleasant).

We can attribute this trend in results to the discomfort multiple people experienced with the visualisations and the headset, which we hypothesise could have led people to rate the interface as more unpleasant. We can conclude that it is possible this score could be increased significantly if using a headset that was better-suited for people with glasses, and if the tracking system was more reliable and the visualisations were better calibrated.

4.5.3.4 Clear-Confusing

The fourth measured scale was *clear-confusing*. As we can see, the results for this scale are fairly consistent between the two interfaces. While there is a small shift in ratings in both directions for the interface with visualisations, with one fewer rating for 1 (clear) but also one fewer rating for 5 (somewhat confusing), we cannot conclude any strong trends for this scale. While the interface with visualisations wasn't significantly more clear than the one without visualisations, it was also not significantly less clear.

4.5.3.5 Annoying-Enjoyable

We can see that the interface with visualisations is divisive on the *annoying-enjoyable* scale, similarly to *pleasant-unpleasant*. While the results for the interface without visualisations are centred around 5-6 (not very enjoyable, but not annoying), the results for the interface with visualisations range from 1 (annoying) to 7 (enjoyable).

As the responses were clustered around 5 for the interface without visualisations, and around 6-7 for the interface with visualisations, we can conclude that the interface with visualisations was overall more enjoyable. Similarly to the results for the *annoying-enjoyable* scale, the spread in values can be attributed to a subset of participants who reported discomfort with the interface due to incompatibility with glasses.

4.5.3.6 Friendly-Unfriendly

The final UEQ scale was *friendly-unfriendly*. The results for this scale are fairly consistent between interfaces, with a shift towards 2 from both directions for the interface with visualisations, and one outlier at 6 (mostly unfriendly). Overall, we can conclude that the participants did not find the interface with visualisations either more or less friendly than the interface without visualisations.

4.5.4 Exit question

To conclude the experiment, we asked the participants to select the interface they would prefer if they had to perform more handovers. The results are given in Table 4.3.

As we can see, 6 participants selected the interface without visualisations and 8 participants selected the interface with visualisations. While the main reason why participants selected the interface with visualisations was that they felt it was helpful for the task, the main reason for selecting the interface without visualisations was discomfort. Two participants expressed the opinion that they found the interface with visualisations helpful and would prefer it if it was more comfortable. Only one participant said that the visualisations were distracting, rather than helpful, for this task.

Participant 6 said that while the interface was helpful for complicated tasks, such as the complex handover, they thought that visualisations were more distracting than helpful for the simple handover. This supports our results from Section 4.5.1, where we found that participants achieved significantly lower average time per handover with visualisations for the complex handover, but not the simple handover.

Participant 2 said that while visualisations helped them become familiar with the robot's behaviour, they felt they had learned enough to use the interface without visualisations in future. However, if presented with an unfamiliar task or robot, they would prefer the interface with visualisations. This ties into the results from Section 4.5.1, where we discovered a *training effect* – participants performed better on the interface without visualisations if they had performed the task with visualisations first.

ID	*Interface*	*Explanation*
1	With visualisations	Found the interface with visualisations more helpful.
2	Without visualisations	Felt that they had learnt enough from this interface, so it was no longer useful; however, would prefer visualisations if working with a new robot
3	With visualisations	Found the interface with visualisations more helpful.
4	With visualisations	Found the interface with visualisations more helpful.
5	With visualisations	Felt more secure when the visualisations were present and overall found the interface more enjoyable.
6	With visualisations	Visualisations would help with more complex handovers, but not with simple tasks such as the simple handover.
7	Without visualisations	Found the visualisations too uncomfortable for the eyes, but thought that they were helpful and would prefer visualisations if they were less uncomfortable.
8	Without visualisations	Found the visualisations too uncomfortable for the eyes, but thought that they were helpful and would prefer visualisations if they were less uncomfortable.
9	Without visualisations	Did not find the visualisations helpful for the task.
10	Without visualisations	Found the interface with visualisations too uncomfortable for the eyes.
11	With visualisations	Found the interface with visualisations more helpful.
12	With visualisations	Found the interface with visualisations more helpful.
13	With visualisations	Found the interface with visualisations more helpful.
14	Without visualisations	Found the interface with visualisations too uncomfortable for the eyes.

Table 4.3: The interface selected by each participant in the exit question, along with the provided justification. The most common reason for selecting the interface with visualisations was that it was perceived as helpful, while the main reason for selecting the interface without visualisations was that the visualisations caused discomfort to the participant.

4.5.5 Conclusions

We conducted an experiment with a collaborative handover task with the Baxter robot, in which we tested the interface with visualisations against the interface with no visualisations. The experiment was conducted with 14 participants. Overall, results achieved through the experiment support the original hypothesis that the interface with visualisations performs better, both objectively and subjectively, than the interface without visualisations.

We found that there was a statistically significant decrease with with 97% confidence in *time per handover* with visualisations for the complex handover, but no significant decrease for the simple handover. On the subjective scales (NASA-TLX and UEQ), we found that the interface with visualisations was rated as more *mentally demanding*, requiring of *higher effort*, more *complicated* and less *clear* and *pleasant*, but also less *frustrating* and more *supportive*.

We found that when asked to select an interface to complete more handovers with, 8 participants selected the interface with visualisations and 6 participants selected the interface with no visualisations. The most common reason for selecting the interface with visualisations was that participants found it to be helpful when completing the task. Discomfort was provided as the most common reason for selecting the interface with no visualisations.

Chapter 5

Conclusions

This section will summarise the results that were achieved from the experiment as described in Chapter 4. We will then use the achieved results to answer the hypothesis set in Section 1.5. We will also discuss the limitations that affected this project and offer suggestions for further work.

5.1 Results

In Chapter 3, we detailed the process of creating an interface for the collaborative handover task described in Section 1.4. We then conducted an evaluation of this interface through an experimental user study described in Chapter 4.

As defined in Section 1.5, we measured the performance of the interface using the metric of *time per handover*. During the experiment, we measured the average time per handover *within participants* for both simple and complex handovers when using our interface with visualisations and when using the control case interface with no visualisations.

We then used a paired t-test to measure whether there was a statistically significant decrease in average time per handover when using the interface with visualisations. The results showed that there was no significant decrease in average time per task for the simple handover; however, we did find a statistically significant decrease for the complex handover. We can use these results to answer the hypothesis from Section 1.5, which claimed the following:

Using an augmented reality interface for a human-robot collaboration task produces an improvement in task performance when compared to performing the collaboration task without an augmented reality interface.

Based on the achieved results, we can conclude that the hypothesis is true for the complex handover, but not for the simple handover. We hypothesise that this is due to the fact that the simple task was too trivial for the visualisations to provide additional

value; however, we conclude that this was not true for the complex case, as we witnessed a decrease in the average time per handover.

Additionally, we found that there was a *training effect*, whereby the participant performed significantly better without visualisations if they had experienced the interface with visualisations before, while there was no training effect in the other direction.

We also asked the participants to rate both interfaces on the NASA-TLX [see Section 4.2.1] and UEQ [see Section 4.2.2] scales. The NASA-TLX scale measured the perceived *mental*, *physical* and *temporal demand*, *performance*, *effort* and *frustration* when using the interface. The shortened version of the UEQ scale measured the interface on six aspects: *obstructive-supportive*, *complicated-easy*, *pleasant-unpleasant*, *clear-confusing*, *annoying-enjoyable* and *friendly-unfriendly*.

We found that on the NASA-TLX scale, participants reported higher mental demand and effort for the interface with visualisations; we can explain this by considering free-form comments from participants, which indicated that when no visualisations were present, they expected the robot to handle the workload for performing the handover, while with visualisations they felt a shared responsibility for the success of the handover and thus expended more mental effort. We found that participants consistently perceived their performance to be higher on the second interface regardless of which order they received the interfaces in. Finally, we found that participants reported lower frustration with the interface.

On the UEQ scale, we found that the participants overall rated the interface as more supportive, but also more complicated. The results showed small and inconclusive differences in the *clear-unclear* and *friendly-unfriendly* scales. The greatest variance was observed on the *pleasant-unpleasant* and *annoying-enjoyable* scales, where some participants rated the interface with visualisations significantly worse than the interface without visualisations. This indicates that even if the interface with visualisations was generally successful in supporting the participants and making the task more successful, there is more that could be done to make the interface more pleasant and enjoyable for users.

After the experiment was completed, we also asked the participants to select which interface they would prefer if they had to complete more handovers. 8 participants selected the interface with visualisations and 6 participants selected the interface with no visualisations. The most common reason for preferring the visualisations was that participants found them helpful for completing the task. Discomfort was provided as the most common reason for selecting the interface with no visualisations. This shows that further work is required to develop more comfortable equipment that is also suitable for participants with glasses. However, the fact that a majority of participants said that they felt the visualisations aided them in completing the task provides strong evidence for the use of visual augmented reality interfaces for collaborative tasks, such as the one used in this project.

5.2 Completion criteria

The two completion criteria for this project [see Section 1.3] were defined as follows:

1. Completed and tested integration between ROS and Unity for AR display.
2. Developed and tested use-cases where augmented information improves interaction between a human and a real robot manipulator.

Chapter 3 describes in detail the process of developing and evaluating a bridge between ROS and Unity, as well as development of an augmented reality interface for a specific use case (the collaborative handover task), while Chapter 4 details the evaluation that was conducted to determine whether the interface improved the interaction. As described in the previous section, we found that both objectively and subjectively, the interface had a positive effect on the task. As such, we can conclusively say that both completion criteria were met.

5.3 Limitations

Our project was limited by multiple factors. The most recurring limiting factor was the technology. Firstly, we found that both Vuforia and ARToolKit, the two most common marker tracking libraries, were incompatible with the Meta headset [see Section 3.8.1]. Secondly, the Meta code did not include functionality to add external localisation methods short of adapting legacy code [see Section 3.8.2].

Additionally, we found that the main reason for why some participants rated the interface as unfriendly, annoying and unpleasant was due to the fact that visualisations with the Meta 2 hardware were not user-friendly for people with glasses, and many people complained about the overall discomfort of wearing the headset. We can hypothesise that this also limited the impact of the positive aspects of the interface, as these were skewed by these hardware-related negative factors.

The second limiting factor was time, which meant that we could not investigate the software limitations, for example for marker tracking, in more detail. Additionally, we were also limited in the number of experiments we could run. Ideally, we would have liked to conduct more experiments to investigate some of the interesting results achieved in the first experiment, such as those for *effort* and *performance*.

Finally, the final limiting factor was instability in the available software. This was first evident in the frequent releases in Meta software, some of which deprecated required functionality (e.g. external tracking). Additionally, new versions of both Unity and Vuforia were released during the period of this project, some of which were incompatible with each other. Finally, as of the point of writing, ARToolKit is in the process of being migrated to a new website as well as getting a new release (ARToolKit 6.0); this means that the previous website, including all documentation, is available for previous releases. Additionally, as DAQRI is a for-profit company selling augmented reality solutions, it is not clear whether new releases of ARToolKit will remain open source.

5.4 Further work

We will suggest some features that we believe are worth investigating based on the results achieved in the evaluation in Chapter 4.

The first aspect that could be investigated is to perform further experimentation to verify and expand on some of the unexpected results achieved with our experiments. In particular, it would be interesting to look further into the effect observed for *effort*, where the participants approached the task differently when presented with the interfaces with and without visualisations. The experiment could be conducted with different tasks to evaluate whether the same effect is present for more active or more passive tasks. Another aspect that could be investigated further is the *training effect* found in Section 4.5.1. This could for example be evaluated by comparing the performance and confidence of two groups, one of which receives the visualisations and one of which does not (however, this would differ from out experiments in that the second group would never receive the visualisations).

Another avenue of research would be to deploy this interface to different robots and different tasks. Indeed, expanding the interface to a different robot was part of the planned extension for this project, but was not completed as technical and time constraints meant there was not enough time for an extension. However, as the code is designed to be flexible and easily extendable, we hypothesise that porting the application to new robots is feasible with minimal modifications.

Finally, if the project was extended, we would like to improve on the quality of the current interface, for example by working to make the tracking system more stable, expanding on the visualised information and spending more time on the visual design to make the interface more user-friendly. Ideally, we would see the interface perform better on all subjective scales from both NASA-TLX and UEQ.

5.5 Final words

Despite the technical limitations, we found the work conducted for this project to be extremely satisfying and interesting. In particular, we appreciated the multi-faceted nature of the system, from human-robot interaction to networking to computer graphics, each of which offered a different challenge to tackle. While our results from the experiments conducted as part of this project indicate that there is potential for augmented reality interfaces with head-mounted AR displays to be used successfully to facilitate human-robot communication, we believe that if the interface was improved and expanded through further experiments, the benefit could be even more significant.

Bibliography

[1] R. S. Andersen, O. Madsen, T. B. Moeslund, and H. B. Amor. Projecting robot intentions into human environments. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 294–301, Aug 2016.

[2] Raffaele Cioffi Antonella Petrillo, Fabio De Felice and Federico Zomparelli. Fourth Industrial Revolution: Current Practices, Challenges, and Opportunities, Digital Transformation in Smart Manufacturing. 2018.

[3] R. T. Chadalavada, H. Andreasson, R. Krug, and A. J. Lilienthal. That's on my mind! Robot to human intention communication through on-board projection on shared floor space. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6, Sept 2015.

[4] J.W.S. Chong, S.K. Ong, A.Y.C. Nee, and K. Youcef-Youmi. Robot programming using augmented reality: An interactive method for planning collision-free paths. *Robotics and Computer-Integrated Manufacturing*, 25(3):689–701, 2009.

[5] Ashirwad Chowriappa, Syed Johar Raza, Anees Fazili, Erinn Field, Chelsea Malito, Dinesh Samarasekera, Yi Shi, Kamran Ahmed, Gregory Wilding, Jihad Kaouk, Daniel D. Eun, Ahmed Ghazi, James O. Peabody, Thenkurussi Kesavadas, James L. Mohler, and Khurshid A. Guru. Augmented-reality-based skills training for robot-assisted urethrovesical anastomosis: a multi-institutional randomised controlled trial. *BJU International*, 115(2):336–345, 2014.

[6] Robert Codd-Downey, P Mojiri Forooshani, Andrew Speers, Hui Wang, and Michael Jenkin. From ROS to Unity: Leveraging robot and virtual environment middleware for immersive teleoperation. pages 932–936, 2014.

[7] ROS Wiki contributors. Introduction — ROS wiki, 2014. [Online; accessed 5-April-2018].

[8] Wikipedia contributors. Augmented reality — Wikipedia, the free encyclopedia, 2004. [Online; accessed 27-January-2018].

[9] Wikipedia contributors. Fiducial marker — Wikipedia, the free encyclopedia, 2004. [Online; accessed 9-April-2018].

[10] Michael D. Coovert, Tiffany Lee, Ivan Shindev, and Yu Sun. Spatial augmented reality as a method for a mobile robot to communicate intended movement. *Computers in Human Behavior*, 34:241–248, 2014.

[11] DAQRI. ARToolKit. https://artoolkit.org. [Online; accessed 4-March-2018].

[12] Siemens AG Dr. Martin Bischoff. ROS#. https://github.com/siemens/ros-sharp, 2018.

[13] Genevieve Dulan, Robert V. Rege, Deborah C. Hogg, Kristine M. Gilberg-Fisher, Nabeel A. Arain, Seifu T. Tesfay, and Daniel J. Scott. Proficiency-based training for robotic surgery: construct validity, workload, and expert levels for nine inanimate exercises. *Surgical Endoscopy*, 26(6):1516–1521, June 2012.

[14] Mark Fiala. ARTag, a fiducial marker system using digital techniques. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 590–596, Washington, DC, USA, 2005. IEEE Computer Society.

[15] FRANKA EMIKA GmbH. https://www.franka.de, 2018. [Online; accessed 10-April-2018].

[16] S. Garrido-Jurado, R. Mu noz Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014.

[17] Sandra G. Hart. Nasa-Task Load Index (NASA-TLX); 20 years later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9):904–908, 2006.

[18] Sandra G. Hart and Lowell E. Staveland. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Human Mental Workload*, 1(3):139–183, 1988.

[19] Martin Hirzer. Marker detection for augmented reality applications. 2008.

[20] Yuchao Hu and Wei Meng. ROSUnitySim. *Simulation*, 92(10):931–944, October 2016.

[21] Hirokazu Kato and Mark Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *1999 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR)*, pages 85–94, October 1999.

[22] Annica Kristoffersson, Silvia Coradeschi, and Amy Loutfi. A review of mobile robotic telepresence. *Advances in Human-Computer Interaction*, 2013:3–3, January 2013.

[23] KUKA AG. https://www.kuka.com, 2018. [Online; accessed 10-April-2018].

[24] Bettina Laugwitz, Theo Held, and Martin Schrepp. Construction and evaluation of a User Experience Questionnaire. In *Proceedings of the 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the*

Austrian Computer Society on HCI and Usability for Education and Work, USAB '08, pages 63–76. Springer-Verlag, 2008.

[25] Bettina Laugwitz, Martin Schrepp, and Theo Held. Konstruktion eines fragebogens zur messung der user experience von softwareprodukten. In *Proc. Mensch und Computer*, pages 125–134. Oldenbourg, 2006.

[26] Jeffrey I. Lipton, Aidan J. Fay, and Daniela Rus. Baxter's homunculus: Virtual reality spaces for teleoperation in manufacturing. *IEEE Robotics and Automation Letters*, 3(1):179–186, 2018.

[27] Jonathan Mace. Rosbridge. http://wiki.ros.org/rosbridge_suite. [Online; accessed 2-March-2018].

[28] Sotiris Makris, Panagiotis Karagiannis, Spyridon Koukas, and Aleksandros-Stereos Matthaiakis. Augmented reality system for operator support in human-robot collaborative assembly. *CIRP Annals*, 65(1):61–64, 2016.

[29] Ivo Malý, David Sedláček, and Paulo Leitão. Augmented reality experiments with industrial robot in industry 4.0 environment. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pages 176–181, July 2016.

[30] T. Matsumaru. Mobile robot with preliminary-announcement and display function of forthcoming motion using projection equipment. In *2006 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 443–450, Sept 2006.

[31] Jeffrey W. McCandless, Robert S. McCann, Kelsey W. Berumen, Stephen S. Gauvain, Victoria J. Palmer, William D. Stahl, and Andrew S. Hamilton. Evaluation of the space shuttle Cockpit Avionics Upgrade (CAU) displays. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 49(1):10–14, 2005.

[32] Stefanos Nikolaidis, Przemyslaw Lasota, Ramya Ramakrishnan, and Julie Shah. Improved humanâĂŞrobot team performance through cross-training, an approach inspired by human team training practices. *The International Journal of Robotics Research*, 34(14):1711–1730, 2015.

[33] Scott Ososky, David Schuster, Elizabeth Phillips, and Florian Jentsch. Building appropriate trust in human-robot teams. In *AAAI Spring Symposium: Trust and Autonomous Systems*, 2013.

[34] PTC Inc. Vuforia. https://www.vuforia.com/. [Online; accessed 4-March-2018].

[35] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[36] Rethink Robotics. http://www.rethinkrobotics.com, 2018. [Online; accessed 10-April-2018].

[37] Eric Rosen, David Whitney, Elizabeth Phillips, Gary Chien, James Tompkin, George Konidaris, and Stefanie Tellex. Communicating robot arm motion intent through mixed reality head-mounted displays. *CoRR*, abs/1708.03655, 2017.

[38] Andres F. Salazar-Gomez, Joseph DelPreto, Stephanie Gil, Frank H. Guenther, and Daniela Rus. Correcting robot mistakes in real time using EEG signals. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 6570–6577, 2017.

[39] Martin Schrepp, Andreas Hinderks, and Jörg Thomaschewski. Design and evaluation of a short version of the User Experience Questionnaire (UEQ-S). *International Journal of Interactive Multimedia and Artificial Intelligence*, 4(6):103–108, 12/2017 2017.

[40] STA. websocket-sharp: A C# implementation of the WebSocket protocol client and server. http://sta.github.io/websocket-sharp/, 2010.

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

[42] E. Svensson, M. Angelborg-Thanderez, L. Sjöberg, and S. Olsson. Information complexity-mental workload and performance in combat aircraft. *Ergonomics*, 40(3):362–380, 1997.

[43] K. Tateno, I. Kitahara, and Y. Ohta. A nested marker for augmented reality. In *2007 IEEE Virtual Reality Conference*, pages 259–262, March 2007.

[44] Mathias Ciarlo Thorstensen. ROSBridgeLib. https://github.com/MathiasCiarlo/ROSBridgeLib, 2017.

[45] Humans to Robots Laboratory. ROS Reality Bridge: Bridge to connect ROS network to Unity. https://github.com/h2r/ros_reality_bridge, 2016.

[46] Humans to Robots Laboratory. ROS Reality: Connect a ROS-enabled robot to Unity. https://github.com/h2r/ros_reality, 2016.

[47] Susana Valdehita, Eva Ramiro, Jesús García, and José M. Puente. Evaluation of subjective mental workload: a comparison of SWAT, NASA-TLX, and Workload Profile methods. *Applied Psychology*, 53:61–86, 01 2004.

[48] VisionStar Information Technology (Shanghai) Co., Ltd. EasyAR. https://www.easyar.com/. [Online; accessed 4-March-2018].

[49] VTT Technical Research Centre of Finland, Ltd. ALVAR. http://alvar.erve.vtt.fi. [Online; accessed 4-March-2018].

[50] Rishi K. Wadhera, Sarah Henrickson Parker, Harold M. Burkhart, Kevin L. Greason, James R. Neal, Katherine M. Levenick, Douglas A. Wiegmann, and Thoralf M. Sundt. Is the “sterile cockpit” concept applicable to cardiovascular surgery critical intervals or critical events? The impact of protocol-driven communication during cardiopulmonary bypass. *The Journal of Thoracic and Cardiovascular Surgery*, 139(2):312 – 319, 2010.

[51] Eric W. Weisstein. Euler Angles. From MathWorld–A Wolfram Web Resource, 2018. [Online; accessed 10-April-2018].

[52] Eric W. Weisstein. Quaternion. From MathWorld–A Wolfram Web Resource, 2018. [Online; accessed 10-April-2018].

[53] Wikitude Inc. Wikitude. https://www.wikitude.com. [Online; accessed 4-March-2018].

[54] P. Wyeth. *The Matter of Vision: Affective Neurobiology & Cinema*. Bloomington: Indiana University Press, 2015.

[55] Zheng Yiyuan, Yin Tangwen, Dong Dayong, and Fu Shan. Using NASA-TLX to evaluate the flight deck design in design phase of aircraft. *Procedia Engineering*, 17:77 – 83, 2011. The 2nd International Symposium on Aircraft Airworthiness.

[56] Gloria Young, Lyubov Zavelina, and Vallire Hooper. Assessment of workload using NASA Task Load Index in perianesthesia nursing. *Journal of PeriAnesthesia Nursing*, 23(2):102 – 110, 2008.

Appendix A

Participant information sheet

THE UNIVERSITY *of* EDINBURGH

Augmented Reality Interface for Human-Robot Interaction
Participant Information Sheet

This information sheet is for participants in an experimental study conducted to evaluate an augmented reality interface for a collaborative task using the Baxter robot. The research is conducted as part of a Year 4 Honours Project in the School of Informatics.

Before participating in this study, it is important for you to understand what the experiment will involve. Please read the following sections carefully. Please ask the researcher if you have any questions about anything in this document. In case of further questions, please contact the student researcher or project supervisor on the following addresses:

Contact information

Eliisabet Hein (Researcher)
s1447681@sms.ed.ac.uk

Dr. Michael Mistry (Supervisor)
mmistry@inf.ed.ac.uk

Purpose of this experiment

This experiment is concerned with evaluating an interface that has been developed for a collaborative task with the Baxter robot. The collaborative task was provided by the Learning Algorithms and Systems Laboratory at EPFL and involves a object handover between the human and the Baxter robot. The Baxter will anticipate the object's trajectory and move it's arm to successfully receive the object being handed to it. The experiment is being conducted to gather data about the effect on task performance when using an augmented reality interface, as well as the subjective user experience.

Experiment procedure

During the experiment, you will be asked to perform the handover task while wearing the augmented reality headset. Before the start of the the experiment, you will be set up with the headset and will perform a calibration procedure. The researcher will guide you through the specifics of the task and the visualisations.

You will then perform 10 handovers with visualisations (*Experiment 1*) and 10 handovers without visualisations (*Experiment 2*). The order in which the experiments will be performed is decided randomly. You will be asked to evaluate the task and the interface using two scales (the NASA Task Load Index and the User Experience Questionnaire) after each experiment. All handovers will be timed, but please keep in mind that we are evaluating the interface, not you, all we ask is that you perform the handovers at the speed that you are comfortable with.

Total time to perform the experiment should be approximately 30 minutes.

Data confidentiality

All collected data will be anonymised and assigned an ID number, which will be used to refer to the results if necessary and will never be connected to your name or other identifying information. The data will be stored securely for up to two weeks after the project submission date, and will be destroyed after this point.

THE UNIVERSITY *of* EDINBURGH

Benefits

There are no monetary benefits to participation. However, you will have the opportunity to contribute to research and will have the opportunity to use cutting-edge equipment and interact with the Baxter robot.

Risks

While the Baxter is designed to be safe for close-quarters human-robot interaction, it is still large and with many moving parts; as such, a reasonable degree of caution is advised when interacting with the robot. Prior to the start of the experiment, the researcher will instruct you in how to ensure that the experiment proceeds safely. The researcher will have the security switch on hand throughout the experiment to stop all movement from the robot — you are encouraged to inform the researcher to stop the robot if you feel unsafe or uncomfortable.

Voluntary nature of study

Participation in this study is completely voluntary and you are free to stop the experiment at any point without any justification. In this case, data collected up to that point will not be used in the study. There will be no penalty of any kind for withdrawing from the experiment.

Once you have read this document, and wish to participate in the experiment, please inform the researcher and sign the attached consent form.

Appendix B

Participant consent form

THE UNIVERSITY *of* EDINBURGH

Augmented Reality Interface for Human-Robot Interaction
Consent Form

Contact information

Eliisabet Hein (Researcher)
s1447681@sms.ed.ac.uk

Dr. Michael Mistry (Supervisor)
mmistry@inf.ed.ac.uk

By signing this document, you agree to the following statements:

- I confirm that I have read and understand the Participant Information Sheet.
- I accept the conditions of the experiment as given in the Participant Information Sheet.
- I confirm that I have had the opportunity to ask questions and had them answered.
- I understand that all personal information will remain confidential and that all efforts will be made to ensure I cannot be identified (except as might be required by law).
- I agree that data gathered in this study may be stored anonymously and securely until two weeks after the submission date for this project.
- I understand that my participation is voluntary and that I am free to withdraw at any time without giving a reason.
- I agree to take part in this study.

Participant name: _____

E-mail address: _____

Additional information

This information will only be used for statistical purposes. If you would rather not answer, leave blank.

Gender (please circle): M / F / Other

Age (please circle): under 18 / 18–20 / 20–25 / 25–30 / 30–35 / above 35

Signature: _____ Date: ____ / ____ / _____

Appendix C

NASA-TLX form

THE UNIVERSITY *of* EDINBURGH

Augmented Reality Interface for Human-Robot Interaction
NASA Task Load Index

Name: _____

Interface (please circle): no visualisations with visualisations

Please rate the task you just completed using the following scales:

Mental Demand — How mentally demanding was the task?

Very Low — Very High

Physical Demand — How physically demanding was the task?

Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?

Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?

Failure — Perfect

Effort — How hard did you have to work to accomplish your level of performance?

Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed were you?

Very Low — Very High

Appendix D

UEQ form

THE UNIVERSITY *of* EDINBURGH

Augmented Reality Interface for Human-Robot Interaction
User Experience Questionnaire (Shortened)

Name: _____

Interface (please circle): no visualisations with visualisations

Please rate the task you just completed using the following scales:

Obstructive	○	○	○	○	○	○	○	Supportive
Complicated	○	○	○	○	○	○	○	Easy
Pleasant	○	○	○	○	○	○	○	Unpleasant
Clear	○	○	○	○	○	○	○	Confusing
Annoying	○	○	○	○	○	○	○	Enjoyable
Friendly	○	○	○	○	○	○	○	Unfriendly

Comments

Please add any other comments about your experience with the task and the interface.