UNIVERSITY OF TWENTE.

OpenIMPRESS

An Open Immersive Telepresence System

Emiel Harmsen

M.Sc. Thesis
March 2018

Supervisors:
dr. Mariët Theune
dr.ir. Dennis Reidsma
Jan Kolkmeier MSc

Human Media Interaction Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente P.O. Box 217
7500 AE Enschede The Netherlands

Abstract

This thesis presents OpenIMPRESS, an open source immersive telepresence system that aims to provide an immersive telepresence experience while being easy to set up and accessible for other researchers to use as well. The system is designed around three features that are intended to increase immersion and usability for the users; mobility at the remote location, an immersive view of the remote environment and a virtual embodiment.

The system is designed in an asymmetrical way, where the visitor is virtually transported to the visitee's environment using a virtual reality system that displays a virtual representation of the visitee's surroundings. A virtual embodiment of the visitor is projected into the visitee's environment using Microsoft's HoloLens augmented reality glasses. The HoloLens' low resolution geometrical scan of the environment is used as a basis for the scene that is shown to the visitor, which is extended with more detailed point clouds that are captured with commodity depth cameras.

A user test was conducted in which was evaluated how the three design aspects contribute to the performance and experience of a telepresence system in a collaborative context. Four system configurations were prepared; one "baseline", in which all design features have been implemented and three "test" configurations from which one of the three design feature's implementations was removed respectively. Thirty pairs of people participated, which had to perform a series of tasks in an escape room-like setting, once with the "baseline" configuration and once with one of the "test" configurations. We saw that especially view independence and immersive display had positive effects on the performance and the visitor's feeling of spatial presence. All aspects increased the system's usability for the visitor.

The designed system has been implemented and has been found to be useful by other researchers already as well. Whats more, we have shown that the implemented design features do indeed improve the experience and usability of the system which can be useful for other research in the area of telepresence as well.

Acknowledgements

I want to thank my supervisors for guiding me through this process; Mariët for giving me the freedom to explore different subjects, Dennis for his creative ideas and Jan for tinkering together on the system. I thank my parents for their support throughout the years. I would also like to thank everybody who took the time to participate in the user tests and my DesignLab family for letting me use the space.

Contents

Chapter 1

Introduction

Connecting directly with others over distance has become an important part of peoples everyday life. Where people previously had to wait to meet in person, texting and calling nowadays allows people to directly communicate with each other from a distance, allowing them to perform tasks that previously required to travel.

Although texting and calling has become part of our day to day private and work life, compared to being physically present, this technology is still limited in the way that callers are perceived and can act in each other's environment. Telepresence systems try to close this gap. The ultimate goal is to make a user feel like he or she is actually present with the other person in their environment.

Current research towards telepresence systems can be divided into two categories; conferencing telepresence systems, where the focus lies on making remote people feel present with each other, and collaborative telepresence systems that besides making people feel present together also focus on making them feel present in the remote environment.

In conferencing telepresence systems the focus lies on supporting conversations between people by providing tools to better express themselves compared to a simple phone call, for example by offering high-quality audio and high-definition life-size video (Feldmann et al., 2010) or stereoscopic displays (Maimone and Fuchs, 2011). A typical application of those systems is often found in conferencing over distance for companies but also in social contexts like (video-)calling with friends and family. Conferencing telepresence is mostly implemented in a symmetrical fashion, which means that at both remote locations people are captured and visualized in the same way, as there is normally no reason to bias the system in favor for somebody by for example given one user access to tools the other doesn't have; all participants are regarded equal.

In this thesis we focus on collaborative telepresence systems. Those systems support collaboration between people by virtually transporting a

remote visitor into the visitee's environment. Hereby the focus is less on conversation and more on action; the remote visitor should be able to operate in the visitee's environment as if they were really there. Typical applications can be found in use cases where a local person requires help from a remote person, for example car repairs, training and investigation work. In these scenarios, the remote visitor needs access to the environment of the visitee, but not the other way around. Therefore, collaborative telepresence systems are ideally asymmetrical (Steed et al., 2012). The visitee for example needs help with their car or with the crime scene they are at, while the environment of the visitor is of less interest to the visitee. I.e., the visitor needs to feel present in the visitee's environment.

The main aspect that contributes to this feeling of presence is immersion (Slater, 2003). A system that is more immersive will make a remote visitor feel more present than a less immersive system. The immersiveness of a system cannot be expressed on a one dimensional scale, as it consists of multiple aspects that together contribute to the overall immersiveness. In the next section we will discuss those aspects in more detail.

1.1 Telepresence System Design Aspects

We recognize three main aspects that contribute to an immersive telepresence system; mobility at the remote location, an immersive view on the remote environment and the ability to have an effect on the remote environment. Those aspects are used throughout this thesis to discuss other telepresence systems and make design decisions for our own system. We will now discuss each aspect in more detail by using examples from previous work.

Mobility at the remote location Allowing the visitor to view the scene independently instead of being fixed to a static view or to a local user, is an important aspect in telepresence systems that focus on remote collaboration. It allows users to simultaneously work in different parts of the scene and gives the remote visitor more situational awareness (Fussell, Setlock, and Kraut, 2003).

Different types of view independence have been explored over the years. One solution is to physically manipulate the remote camera, a popular implementation of this are telepresence robots. They usually consist of a camera and screen on wheels that can be controlled be the remote visitor over the internet (Figure 1.1). A video stream of the visitors face is displayed on the robot's screen while the robot captures images of the environment and sends them back to the visitor. This way a remote visitor has the ability to navigate a space and get a sense of the environment. A lot of different telepresence robots have been developed, most of them are also commercially

Figure 1.1: A Double telepresence robot.

available (Kristoffersson, Coradeschi, and Loutfi, 2013). Although telepresence robots give a remote visitor a physical body and complete freedom to navigate the environment, it also has its drawbacks. A telepresence robot can be considered clunky, as the unawareness of the visitor about the robot's body means it could easily bump into things or get stuck somewhere. Also the speed and the amount of perspectives the robot allows for are often limited.

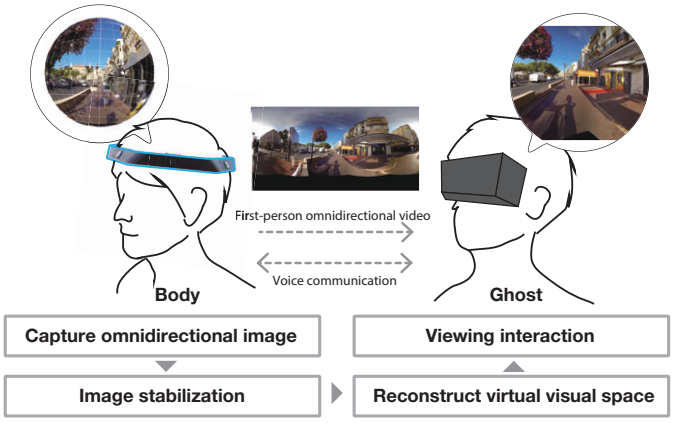


Figure 1.2: Overview of JackIn Head (Kasahara and Rekimoto, 2015).

Other solutions make use of a synthesized view based on imagery captured from the visitee's perspective. In such a scenario, a local person wears a camera that streams a first-person perspective to the visitor. This limits

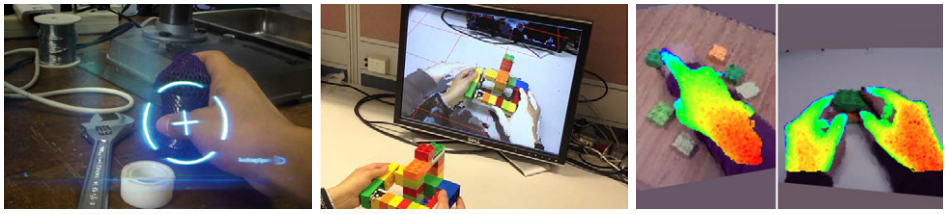
the navigational freedom of the visitor, although solutions have been made that still give the visitor some degree of independence from the local person. For example, *JackIn Head* (Kasahara and Rekimoto, 2015) uses a 360° degree camera mounted on a person's head to allow a remote visitor to look around in the person's environment (Figure 1.2). Similarly, Gao et al. (2016) made a system that uses a tracked head mounted depth camera instead of a 360° camera. The images are oriented to the visitor in such a way that when the worker turns his/her head, the projected image at the visitor would turn as well and therefore requires the visitor to turn his/her head to follow the image. This doesn't give the visitor any extended navigational freedom, but helps the visitor to keep track of the worker's current orientation.

Another solution is demonstrated by Tait and Billinghurst (2015), which used a statically mounted depth camera above a table to create a live 3D scan of the workplace. The visitor was then free to navigate this live 3D scan with a virtual camera using a desktop user interface. They researched the effect of this in a collaborative setting and showed that it results in "faster task completion time, more confidence from users, and a decrease in the amount of time spent communicating verbally during the task".

An immersive view on the remote environment The way the remote environment is presented to the visitor has effect on the telepresence experience as well. A more immersive display makes the visitor feel more present in a virtual environment (Baños et al., 2004). A display with a wider field of view will for example fill more of the visitor's view with the remote environment which draws their attention away from the local environment. Telepresence systems have used different display systems that can be distinguished by the degree of immersion they provide the visitor. Earlier systems used 2D monitors on which a video stream of the local user is displayed, like the *HandsOnVideo* system (Huang and Alem, 2011). Newer systems try to immerse the visitor more in the remote environment, this is often done with head mounted displays as shown in by Tecchia, Alem, and Huang (2012). Other display systems are also used, like CAVE projection systems where multiple walls of a room are projected with live images from a remote environment as shown by Komiyama, Miyaki, and Rekimoto (2017). Those systems are often used in combination with head tracking to make the movement of the images match the visitor's head movements.

Have an effect on the remote environment Besides allowing a remote visitor to observe an environment, telepresence systems usually also allow visitors to have a certain degree of effect on this environment. This can be done by adding virtual elements to the environment or by physically manipulating the environment.

Virtual elements can be used for supporting interaction between people



(a) (b) (c)

Figure 1.3: Various styles of virtual assistance overlays. a) Spatial remote pointing (Kasahara and Rekimoto, 2014); b) Remote helper's hands displayed on a monitor (Tecchia, Alem, and Huang, 2012); c) Remote helper's hands displayed in a HMD (Gao et al., 2016).

by showing things to the local users through a display. This can be done by overlaying instructions from the remote visitor onto the local user's view by for example using a monitor in front of the local user or using a head mounted display (HMD). The instructions can consist of annotation-style elements, like arrows and markers as shown by Kasahara and Rekimoto (2014) (Figure 1.3a). Those are mainly used to direct the local user to certain elements in the environment.

To assist the local user in tasks that require more fine control, an overlay of the remote user's hands are often used, as shown by Amores, Benavides, and Maes (2015), Tecchia, Alem, and Huang (2012), Gao et al. (2016), and Huang and Alem (2011) (Figure 1.3b and Figure 1.3c). This allows the remote users to guide the local user in a more natural way when dealing with object manipulation tasks, like repair or construction tasks by using gestures one would intuitively use.

Physical manipulation is another way to make visitors have an effect on the remote environment, for example by allowing them to move objects. By themselves telepresence robots already manipulate the environment, as they are a physical entity that makes part of an environment. By driving around (and maybe even by bumping into things) they allow a remote visitor to have an effect on this environment. By equipping a telepresence robot with a dedicated component for object manipulation, like a robotic arm, the meaningfulness of those manipulations can be increased (Paulos and Canny, 2001).

Benavides, Amores, and Maes (2015) made a system called *Remot-IO* that allows a remote visitor to manipulate dedicated internet connected objects by using virtual gestures, without requiring a physical embodiment. They demonstrate the system with a radio that is connected to the system. The remote user can turn the knobs by using hand gestures that are captured and translated into physical rotations on the radio knobs.

The current state of the art in the field of mixed reality telepresence is Holoportation (Orts-Escolano et al., 2016). This system uses consumer grade AR and VR display technologies in combination with a new 3D capture system to capture full 360° 3D models of people and objects and visualize them at a remote location in real-time. All of this is done with the goal of creating an experience that resembles physical presence as close as possible. Users at both locations use a HoloLens HMD to get an independent view on each other's environment. Although no physical manipulation of each other's environment is possible, the high resolution scans of the users allow for very precise gestural instructions that are properly aligned with their context. Orts-Escolano et al. (2016) conducted a preliminary qualitative user study with 10 participants, using a social interaction and a physical object manipulation task. The participants noted how more intuitive it is compared to normal phone calls or using Skype.

Besides providing an immersive experience, a successful telepresence system will also have to be accessible. We aim to make a system that can be used by other researchers as well, therefore it should be easy for people to set up an environment for a virtual visit, as well as for people to go on a virtual visit. Next to being easy to set-up, the system should also be easy and intuitive to use, for which we can look at factors like the users' performance, usability and presence.

1.2 Research Questions

This thesis is about the design, implementation and evaluation of a mixed reality telepresence system called *OpenIMPRESS*, which stands for *open source immersive presence system*. The goal is to create a telepresence system that is both immersive and accessible. This puts OpenIMPRESS in line with the work that is done on Holoportation but where Holoportation aims to be an exercise in the field of computer vision and computer graphics, focusing on realism by pushing technologies to their limit, our work does not try to improve on that. OpenIMPRESS focuses instead on the (social) interaction and collaboration aspects that systems in this line of research may be used for. The goal is to make an open system that can be used as a basis by other researchers as well. We try to do that by making a system that only uses off the shelf consumer hardware, is flexible and easily reconfigurable and makes use of open source code or freely available software which makes it accessible for anybody that wants to use it.

This leads to the following first main research question and subquestions:
RQ1. *How to create a state of the art end-to-end mixed reality telepresence system that is immersive and accessible?*

- **RQ1.1.** How to create an immersive telepresence experience?

- **RQ1.2.** What components can be used, while keeping it accessible?
- **RQ1.3.** How to cover a large working area?
- **RQ1.4.** How to make the system easy to set-up and use?

Compared to previous telepresence systems and research, OpenIMPRESS uses puts previously tested components in a different context than the ones they have originally been tested in. This means that a lot of knowledge about how those aspects help or hinder collaboration may not apply anymore. For example; VR has been used in telepresence systems before, but in most cases only in combination with a first person view on the remote environment which, at the most only allows the remote visitor to change the rotation of the view (Gao et al., 2016). Full view independence has been tested only in combination with a 2D monitor set-up, not in combination with VR (Tait and Billinghurst, 2015). Overlaying gestures from the visitor onto the visitee's view has been tested as well, both in VR and with 2D interfaces but not with (full) view independence (Tecchia, Alem, and Huang, 2012; Amores, Benavides, and Maes, 2015; Fussell, Setlock, Yang, et al., 2004). A better understanding on how those aspects influence collaboration would be necessary to pinpoint what aspects are worth further research and development efforts.

Also, those components have always been tested in slightly different scenarios, with different types of systems and tasks. We now have the opportunity to better compare the effects of those components, by evaluating them while keeping variables like the system and tasks the same.

This leads us to the second main research question, which is: **RQ2.** *How do the three telepresence system design aspects influence the performance and experience of the users in a collaborative problem solving task?*

To answer this question we focus on the aspects presented in section 1.1. Because OpenIMPRESS does not (yet) have the ability to let the visitor have a physical effect in the on-site location directly, we limit the aspect of *affect the remote environment* to visual effects only. Which means in this case the visualization of the visitor in the visitee's environment. This results in the following three subquestions:

- **RQ2.1.** How does view independence influence the performance and experience of the users in a collaborative setting?
- **RQ2.2.** How does using an immersive display influence the performance and experience of the users in a collaborative setting?
- **RQ2.3.** How does giving the visitor a virtual embodiment influence the performance and experience of the users in a collaborative setting?

Those questions are answered with a quantitative user study detailed in chapter 6.

Chapter 2

Background: Free Viewpoint Video

Many remote collaboration tools allow the visitor to see the local environment through a fixed perspective. This can be from a fixed camera in the room or from a user-worn camera capturing the visitee's view. For OpenIMPRESS on the other hand, we try to create an immersive telepresence experience, which includes giving the visitor mobility at the local environment. The group of technologies that focus on the creation and playback of video content that enables viewers to choose their own perspective during playback is called *Free Viewpoint Video* (FVV).

This aspect is identified as a major challenge for the implementation of this system and therefore discussed in more detail[1]. Smolic (2011) divides the FVV pipeline into seven steps (Figure 2.1) of which the most important ones are discussed individually below.

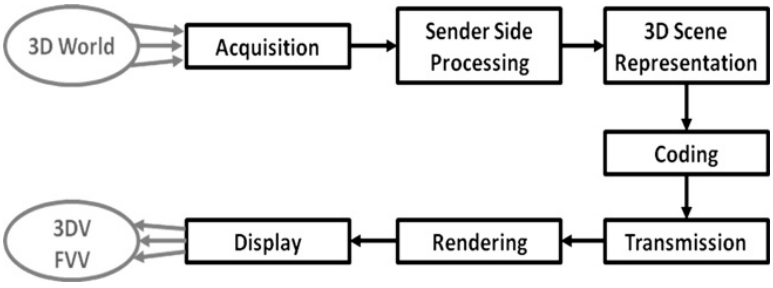


Figure 2.1: The FVV pipeline (Smolic, 2011).

In this chapter, we look at the most relevant aspects of free viewpoint video for our telepresence system. Each aspect is explained and examples

[1]Most of this section has been taken from Harmsen (2017).

are presented of their implementations in other systems. Section 2.1 *Acquisition* focuses on how FVV is captured by explaining the different kind of camera's and set-ups. Section 2.2 *Processing* present different techniques of how the acquired data can be processed and represented. Section 2.3 *Display* presents different options of how to display the FVV data. Section 2.4 concludes the chapter by summarizing the presented techniques and pointing out which are the most relevant for OpenIMPRESS.

2.1 Acquisition

To capture FVV one can use an outward or an inward acquisition topology (Figure 2.2). An outward topology means that the capturing devices are faced outwards from a center point (like 360° cameras). In this case the viewer is limited to rotational perspective changes only and one could argue whether this can be considered Free Viewpoint Video as the viewpoint remains the same. With an inward topology the capturing devices are all pointed at the same subject from different locations around that subject, so instead of allowing the viewer to look around from the same point in space, the viewer can "travel" to different points in space and observe the subject from there.

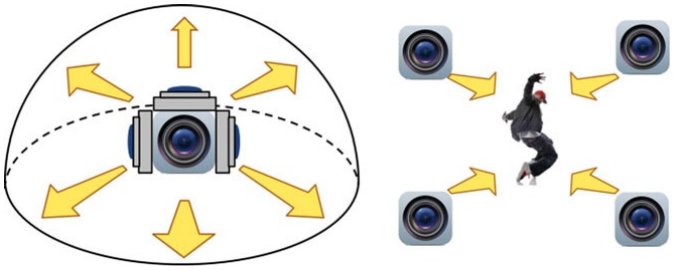


Figure 2.2: Left: Outward FVV; Right: Inward FVV (Lee, Tabatabai, and Tashiro, 2015).

A popular way to capture FFV content is to use one or more designated depth cameras that next to capturing a regular 2D image also capture a separate depth map that gives each pixel in the 2D image a depth coordinate. This property makes it especially interesting for FVV as this means one can create a volumetric capture of a scene in real-time. There are various depth cameras on the market that use different technologies to determine the depth information. Langmann (2014) conducted a comparative study, analyzing the differences between various models of depth cameras.

A depth camera that is used extensively is the Microsoft Kinect (Kuster et al., 2011) (Hauswiesner, Straka, and Reitmayr, 2011) (Cazamias and Raj, 2016). Izadi et al. (2011) present a system that generates a 3D mesh representation of a scene based on the depth data from the Kinect sensor. This

technique requires the user to move the sensor around the scene so it can be captured from multiple angles. By fusing those different views together a high quality model of the scene is generated. Because the scene is captured from multiple angles over an extended period of time, this technique is capable of producing high quality models but therefore is also less ideal for real-time streaming as it assumes no objects are moving while capturing. This technique is usually applied when creating a map of an indoor environment (Khoshelham and Elberink, 2012) (Henry et al., 2012).

Systems made to capture moving objects also exist; they usually consist of multiple Kinect sensors and have become popular among researchers to create FVV's of objects and persons (Kainz et al., 2012) (Doc-Ok, 2014). Google put this into practice in their experiment called "Virtual Art Sessions"[2], in this experiment a group of artists were asked to create a sculpture in virtual reality. Normally a spectator wouldn't be able to see both the artist and the artwork at the same time, because they exist in different "realities". To solve this, Google (2016b) recorded the artists volumetrically while they worked on their piece. Those recordings were then mixed together with the virtual environment and distributed online so spectators can watch FVV of the artists sculpting their virtual artworks in mid-air. Maimone and Fuchs (2012a) developed a system using multiple Kinect sensors for real-time volumetric 3D capturing of environments specifically for telepresence applications. Kinect sensors project infrared light patterns onto the environment in order to determine its distance from objects. This can cause problems when one uses multiple Kinects in the same environment because a Kinect then can't distinguish its own patterns from other Kinect's patterns. To solve this problem, the system uses an IR interference reducing method which is based on shaking the sensor (Maimone and Fuchs, 2012b) (Butler et al., 2012). The shaking causes the patterns emitted by the other Kinects to be blurred out, thus improving the detection of the sensor's own pattern and therefore reducing the interference.

2.2 Processing

The captured scene can be represented in a way somewhere between an image based representation or a geometry based representation (Smolic, 2011), as shown in Figure 2.3.

On one end of the spectrum is the image based approach. In this approach there isn't any knowledge about the geometry of the scene; images from each camera are processed and transmitted as separate videos. Interpolation can generate intermediate views, but a high capture density is required to get an acceptable quality.

[2] `https://developers.google.com/web/showcase/2016/art-sessions`

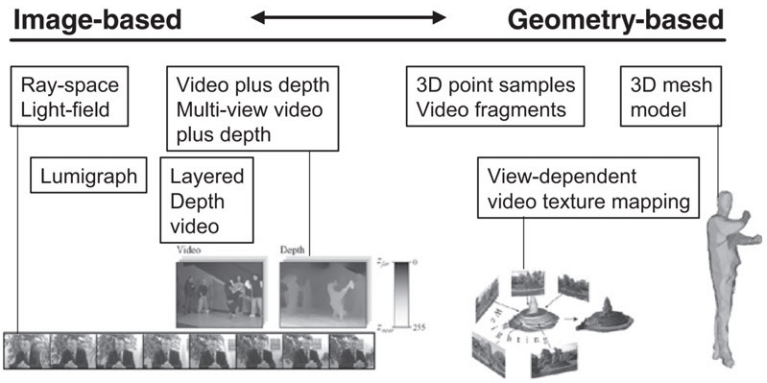


Figure 2.3: 3D scene representations for 3DV and FVV (Smolic, 2011).

On the other end, with a geometry based approach the scene is represented by three dimensional meshes on which image textures are applied. Depending on the method of acquisition, this requires an accurate and robust 3D reconstruction of the scene which can be challenging but allows more freedom of navigation when viewing. Carranza et al. (2003) made one of the first attempts and developed a system to create a geometry based representation of human actors using a seven-camera setup. A virtual human body model was put in the right pose by comparing its silhouette to the silhouettes of the actor from the different camera angles. Since then, those capture systems have become more advanced, as shown by Collet et al. (2015). They use an inward acquisition topology with 106 cameras from which high quality 3D tracked meshes are generated.

Alternative approaches that lie in between image and geometry based approaches are also possible; instead of using meshes, point clouds can be used to represent geometry. Point clouds can be generated by assigning each pixel of a 2D color image a depth value (and thus creating a 3D coordinate) and placing that pixel in 3D virtual space. Sensors like the Microsoft Kinect exactly provide this kind of data, thus making this approach relatively easy to implement. Using point clouds reduces the complexity of the processing step drastically, as no geometry has to be generated from this data. Also, it is more forgiving when capturing complex objects; "Using point clouds enables the alignment for scenes full of objects that are difficult to model; for example, trees." (Zhao, Nister, and Hsu, 2005). This is because fewer assumptions are made during the processing of the displayed data and thus relatively fewer mistakes are made. The disadvantage of skipping the 3D geometry reconstruction step is that objects look less solid because the spaces in between the points of the point cloud are not getting filled in.

In order to send the data that is captured, it has first to be encoded and compressed. Compression is necessary to keep the data stream within the

bandwidth limits. There are different encoding and compression methods available for different types of Free Viewpoint Video. For encoding geometry based FFV, the MPEG-4 standard provides an extension called the Animation Framework eXtension (AFX) which allows to define high-level geometry, texture, volume, and animation components for enhanced interactive multimedia applications (Bourges-Sévenier and Jang, 2004) (Smolic et al., 2006). A lot of research has also been done already into compression algorithms for streaming volumetric point cloud video data: (Moreno, Chen, and M. Li, 2017) (Moreno and M. Li, 2016) (Rusu and Cousins, 2011) (Kammerl et al., 2012) (Mehrotra et al., 2011) (Sohn, Bajaj, and Siddavanahalli, 2004).

2.3 Display

Before 3D data can be displayed it has to be rendered. This means that the 3D data is turned into image data that can be displayed on for example a monitor. 3D video-game engines like *Unity*® or *Unreal Engine*® are very suitable for this, as their main task is converting 3D data into 2D image data that can be displayed on screens, but besides that they also allow for integration of custom features, offer a lot of support via online forums, are free to use for these purposes and offer easy integration with virtual reality headsets.

Virtual reality (VR) headsets or head mounted displays (HMD) have recently become a popular way of displaying virtual environments. HMDs allow for the user to be completely immersed in a virtual world by means of a stereoscopic display that covers a big part of the user's field of view and sensors that register the user's movements and translate them to virtual movements. There are multiple HMDs currently on the market including models that require a phone to be installed to act as a screen, sensor and playback device and models that have the screen and sensors already built-in but need to be wired to a PC to function. Of this last group the Oculus Rift and the HTC Vive are the most popular models and are also often used in research settings. Both now support full room-scale positional tracking which gives access to the head's current position and rotation in the physical space.

The disadvantage of using a virtual reality HMD like the HTC Vive is that the user is completely shut out of the real environment and bystanders can't get a proper idea of what the operator is doing. Using an augmented reality (AR) headset instead of a VR headset solves this problem partially. AR headsets like the Microsoft Hololens (Microsoft, 2016) use transparent glasses that allow for virtual objects to be displayed on top of the user's view of the real world.

Another option would be to use a stereoscopic projection screen in combi-

nation with head-tracking to create an immersive experience like Pouliquen-Lardy et al. (2016) used for their research experiment. The advantage of this method is that bystanders can look at the same display as the operator is looking at and can therefore better place the operator's actions into context.

2.4 Conclusion

This chapter presented different aspects of free viewpoint video systems. We discussed different acquisition, processing and display techniques, together with examples of various implementations. We saw that techniques that are used in FVV systems can be a good basis for a telepresence system. In fact, the line between a FVV system and a telepresence system can often be difficult to draw, especially with a live streaming FVV system.

When a certain FVV technique is chosen, it determines many of the characteristics of the telepresence system that especially the visitor will experience. For example: the acquisition technique determines what the visitor will be able to see and the level of detail, the processing technique determines how much freedom the visitor will have in navigating the captured data and the display technique determines for a big part how immersed the visitor will be.

In the next section we discuss the design choices that are made for the OpenIMPRESS system, including parts inspired by FVV.

Chapter 3

Architecture and Global Design

OpenIMPRESS is designed as a mixed reality telepresence system that uses the latest consumer electronics to capture, stream and visualize an environment and the people in it with the goal to make people feel as present as possible from a physically remote location. This chapter describes the choices that were made when designing this system and the architecture that those choices eventually lead to.

3.1 Global Design Considerations

The design of OpenIMPRESS is based on the idea of making the visitor feel as if being present at the on-site location and making the visitee feel as if the visitor is present with him or her there as well. The system spans two different locations: the on-site location and the remote location, where respectively the on-site operator (OSO) and the remote operator (RO) are located. Compared to Holoportation (Orts-Escolano et al., 2016), this system is asymmetrical. This means that there is a clear distinction between the RO, who *visits* the on-site location and the OSO, who *is being visited*.

The basics of the system are shown in Figure 3.1. At the on-site location the environment is captured and sent to the remote location. Here, the RO can freely navigate a virtual representation of this environment. The RO's head and hand movements are captured and sent back to the OSO where they are visualized to the OSO. The OSO is wearing a Microsoft Hololens HMD, which are augmented reality (AR) glasses. They can display content on top of the real world using transparent displays that are placed in front of the user's eyes. Besides visualizing elements in an environment, the Hololens can also be used to capture an environment. The Hololens is equipped with multiple depth sensors that are used to create a spatial model of its current environment in real-time and track its own location within this environment.

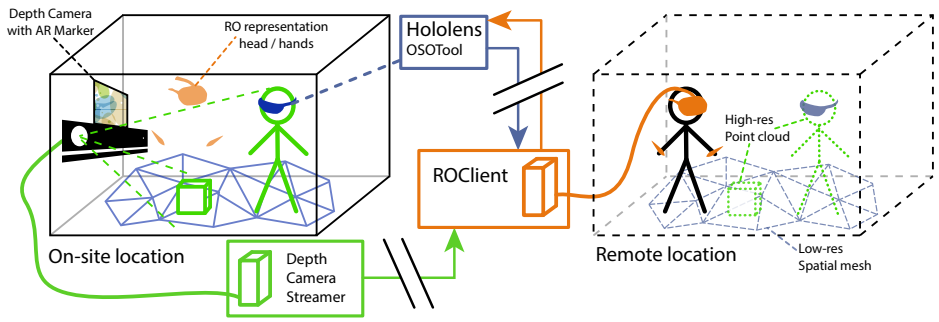


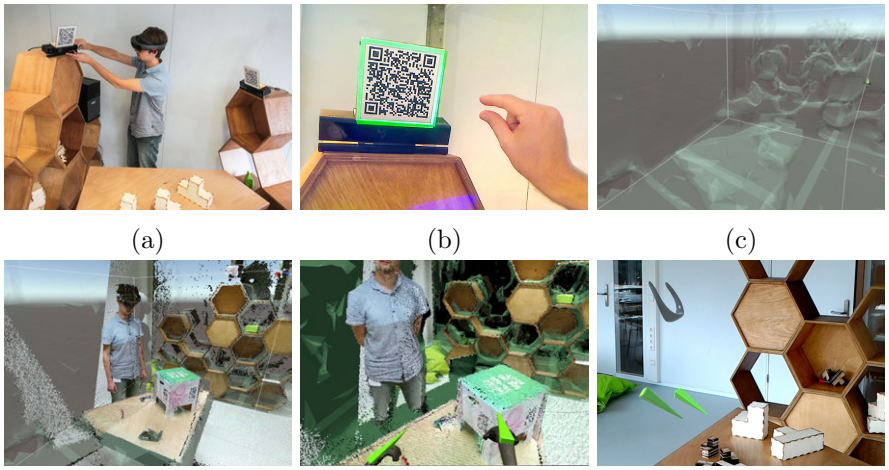
Figure 3.1: OpenIMPRESS basic system overview.

A geometrical representation of this spatial model is send to the RO, where it serves as a low quality but wide area representation of the environment. This model is extended with scans acquired by one or more depth camera's that are also installed at the on-site location. Those scans are visualized as point clouds and serve as a more detailed but narrow representation of the environment. In order to properly align the point clouds from the depth camera's with the low quality spatial model, the depth camera's are equipped with AR markers. Those markers can be scanned by the Hololens to determine their position relative to the HoloLens's spatial model. This position is then sent to the remote location where it is used to align the point cloud data. Figure 3.2 shows the envisioned set up procedure experience step by step plus a basic interaction.

3.1.1 AR vs VR

In our design we want to make the RO feel transported into the OSO's space and make the OSO feel like the RO is there with them. Orts-Escolano et al. (2016) mention in their findings that participants that used their AR condition felt as if their remote partner was there with them in their space, while participants that used the VR condition had the impression of being in the partner's space. This suggest that for making people feel transported to another location VR may be a better option, where AR is more ideal for making people feel like they are being visited.

Our design is informed by this and therefore the RO is using a VR setup and the OSO is using AR. The RO's own surroundings are completely blocked from the RO, leaving only the visualization of the on-site location's environment. The OSO on the other hand, can see the on-site location's environment as he or she would normally be able to, as well as the superimposed representation of the RO. This is assumed to be more ideal for remote assistance scenarios, where only the OSO's environment is of importance and adding information about the RO's environment would create confusion.



(a) (b) (c)

(d) (e) (f)

Figure 3.2: a) OSO installs a depth camera. b) OSO calibrates the depth camera. c) RO receives spatial mesh from HoloLens. d) RO receives point clouds, which are aligned with spatial mesh. e) RO points at objects using controllers (green cones). f) OSO sees RO representation through HoloLens.

3.1.2 On-site environment capturing

Research question *RQ1.3.* focuses on how to cover a large working area. Current telepresence systems that make use of 3D scans are often limited by the area in which the users can operate. 3D depth camera's are installed at pre-planned strategic places and their extrinsics (position and orientation) are determined during a calibration step which is necessary to properly align their captured views to each other and to the rest of the virtual representation. This fixes the size of the scanning area and thus also fixes the working area that can be used during a telepresence session.

For OpenIMPRESS we looked at solutions that allow for more dynamic reconfiguration of the scanning area. So that when the work shifts to a different location, the scanning space adapts on the fly and the RO is kept involved in the task at hand.

The different solutions that have been considered all build on top of the HoloLens' built-in tracking system. The HoloLens automatically creates a geometrical spatial model of each environment it visits by continuously scanning its surroundings and updating the spatial model when necessary. This model can then be used by developers to make their applications interact with the environment. Because it is not intended to be viewed by itself, the spatial model is a relatively low resolution model and contains no color

information, although techniques exist for adding color later on[1] [2]. Despite this limited level of detail, it still contains a lot of useful general information about the wearer's environment; the size and shape of the room, locations of furniture, doors and windows and the shape of them. Because the spatial model is extended and updated with the current working environment in real-time and without requiring any user intervention, it is an easy starting point for covering a large working space. Also, the absence of the need for the users to do any setup actions or calibration steps helps with keeping the system easy to set-up and use. This brings us to the first solution that is implemented for covering a large working area. The spatial model is streamed from the OSO's HoloLens to the RO where it is used as a base layer of the virtual representation of the OSO's environment.

This base layer is extended with high resolution point clouds. They add more detail to the overall virtual representation of the OSO's environment. They are captured by 3D depth cameras, like the Microsoft Kinect, that are placed in the working area. Compared to the spatial model, a point cloud usually contains substantially more detail, as it has a higher resolution, has a higher refresh rate, is more accurate and contains color. When available, point clouds are clearly the preferred way of visualizing the OSO's environment. The problem is that it is difficult to cover OSO's complete environment with 3D cameras to allow the point cloud to be available everywhere, that's why the spatial model is used to fill the "empty gaps".

The point clouds will have to be positioned correctly relative to each other and to the rest of the virtual representation. To do this, a calibration step is necessary in which the extrinsics of the cameras is determined. There are several methods that can be used for calibration.

A dedicated positional tracking system can be used, like OptiTrack or the HTC Vive tracker, to track each depth camera's location in the environment. This method is the most reliable but requires an extra component to be set up and therefore compromises the ease of deployment.

Another option is to attach the depth camera on top of the Hololens. The location of the Hololens is already known and therefore, by attaching the depth camera to the Hololens, the depth camera's location will also be known. Garon et al. (2016) have used a similar set-up for different purposes. To keep the whole system mobile they also attached a stick-PC and a battery pack to the Hololens. This method retains the ease of deployment as it doesn't introduce extra components (it makes use of the tracking from the Hololens) but it is limited to tracking the location of only one depth camera.

The relative positions between the depth cameras and the spatial mesh can be calculated by creating an algorithm that searches for similar geomet-

[1] `http://www.matrixinception.com/wp/hololens-roomscan-reloaded/`
[2] `https://github.com/ywj7931/Hololens_Image_Based_Texture/`

rical patterns. This method is the most versatile as it doesn't need any extra components to be added to the system and scales when more depth cameras are added. The downsides are that there has to be overlapping data and it requires recognizable features in the geometry and complex algorithms that make it prone to making mistakes.

We chose to attach printed markers on each depth camera and solving their locations using the image camera on the Hololens. This method is scalable as an unique marker can be added for each depth camera, but is limited by the camera on the Hololens, which has to be able to see the markers to be able to solve their locations. Because the cameras are not supposed to be moving this is not a big problem. If cameras are moved from their initial location anyway, their locations will have to be solved again by "observing them" with the camera of the Hololens.

3.2 System Configurations

Besides telepresence, a system like OpenIMPRESS can be used for a variety of different applications by modifying how the system is configured. Four main system configurations have been identified that differentiate themselves by either displaying the captured data in real-time or at a different point in time and having the captured data being played back at either the same or at a different location from where it was captured.

		physically	
		co-located	***dislocated***
temporally	***co-located***	modified reality	remote assistance
	dislocated	in-place training	relive memory

Table 3.1: The basic system configurations with corresponding application examples

Table 3.1 shows the four basic possible configurations in relation to each other. Temporal co-location means that the viewer receives the recorded data as it's being recorded. Temporal dislocation means that the viewer is looking at data that is prerecorded. Physical co-location means that the viewer is standing physically at the same location from where the data was or is being recorded; the virtual environment is aligned with the physical alignment. Physical dislocation means that the viewer is looking at the virtual environment while being at a physically different location from where the data was or is being recorded.

Depending on how the system is configured, it can be applied in different application scenarios. Of course those basic configurations can be extended to suit different scenarios by for example supporting multiple people viewing or capturing at the same time, using different playback or recording devices

and using different ways of distributing the recorded content. For this thesis however, we will focus on the remote assistance application, which means we use a temporally co-located and physically dislocated configuration.

3.3 Architecture

The architecture of the system is based on the design considerations from section 3.1. A architecture diagram is shown in Figure 3.3. The system is split up into seven main components. The system can be reconfigured by disabling or rearranging certain components to make it more suitable for different scenarios. Each component can also be used separately and be integrated into other systems if necessary.

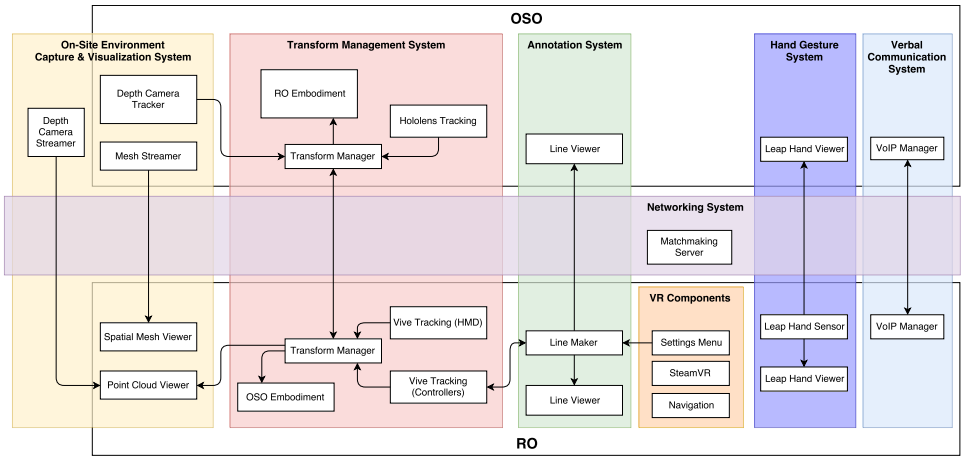


Figure 3.3: The system architecture diagram of OpenIMPRESS.

The *On-Site Environment Capture and Visualization System* focuses on capturing the on-site environment with as much detail as possible and visualizing it at the remote location. The *Transform Management System* manages the sharing of multiple transform between the RO and OSO. The *Annotation System* enables the RO to create annotations in the OSO's environment. The *Networking System* handles all the data connection between components that have to communicate over a network. The *VR Components* manage the experience the user has in VR. The *Hand Gesture System* captures the RO's hands and displays them at the OSO. The *Verbal Communication System* allows the RO and OSO to talk with one another by using their voice.

Each component is described in more detail in chapter 4 together with implementation details.

Chapter 4

Implementation

In this chapter we present how the different components that are described in section 3.3 are implemented.

4.1 On-Site Environment Capture and Visualization System

The On-Site Environment Capture and Visualization System contains all the components that are used to capture the OSO's environment and visualize it at the RO. It can be divided into two subsystems; one for point-cloud data and one for mesh data.

4.1.1 Depth Camera Streamer

The *Depth camera streamer*[1] is a standalone component written in C++ that captures depth and color image data from a depth sensing camera and transmits them over a network using an UDP Connector. Different versions of this component exist that are made to work with different brands of depth cameras or with different libraries for the same camera. Versions for the Microsoft Kinect V2 using the default Microsoft library[2] and the libfreenect2 library[3] have been made, as well as for the Intel RealSense camera[4].

The current version of the streamer uses JPEG compression on the color image data, encoding each frame as a separate JPEG image, and no compression on the depth data. A next step in the development of this component would be to find and implement an appropriate compression algorithm for the depth data as well.

[1] Most of this component is developed by Jan Kolkmeier

[2] `https://developer.microsoft.com/en-us/windows/kinect`

[3] `https://github.com/OpenKinect/libfreenect2`

[4] `https://github.com/IntelRealSense/librealsense`

4.1.2 Point Cloud Viewer

The *Point-Cloud Viewer* contains a collection of components that receive, process and visualize point-cloud data from various sources in the 3D environment.

FrameSource is a component that standardizes the way that sources of point-cloud data provide their data to the rest of the system. Several types of *FrameSources* exist, like: *KinectSource* which provides data directly from a Kinect V2 camera and *RandomSource* which provides random data that is useful for testing the performance of the renderer. *DepthStreamingSource* is the only source that is actually used in OpenIMPRESS at the moment, this source allows data that is sent by a *Depth Camera Streamer* to be received and passed on to be visualized.

After data from the *Depth Camera Streamer* is received by the UDP Connector, it is passed on to the *DepthStreamingListener*. Here the type of data the packet holds is detected and fed into the appropriate parser. Then, the parsed data is passed to a processor that calculates the xyz-coordinates of each pixel based on the camera intrinsics and combines it with the data from color packets into a coherent *PreFrameObj* consisting of:

- An array of xyz-coordinates for each pixel
- An array of color values for each pixel or raw DXT1 or JPEG compressed data representing the color data for each pixel
- The camera's position and rotation values

When a new *PreFrameObj* is completed it is queued into the frameQueue of the *FrameSource*.

The *PointCloudViewer* is the component that manages the visualization of the point cloud data. It starts by calling the public function *GetNewFrame* of a *FrameSource*. When this function is called, *FrameSource* will retrieve the newest *PreFrameObj* from the queue and convert it into a *FrameObj*. This conversion converts the raw color and depth data into Unity Texture2D objects which allow them to be read directly by the shader for rendering. Because only the newest frame is retrieved from the queue and the rest is discarded, this conversion step is only executed when it is certain that a particular frame will be visualized as it takes a relatively long time to execute and can only be executed in the main loop and thus affects the frame rate directly in a negative way.

The *PointCloudViewer* contains a mesh object that holds one vertex for every point in the point cloud. A material with a shader is applied to this mesh that reads the color and position textures and colorizes and displaces the corresponding vertices accordingly, which is how the point cloud is visualized. After the *PointCloudViewer* receives a *FrameObj* it will apply the depth and color textures to the material of the mesh object. It

will create a new mesh object if one doesn't exist already or if the resolution doesn't match the one of the *FrameObj*. The mesh object is basically a dummy, as the positions of the vertices are not actually used by the shader, but it is still necessary to make the shader work properly as the shader needs to be applied to an object that holds the same amount of vertices as the amount of points in the point cloud. Because Unity has a maximum limit of 65534 vertices per object, the mesh is split up into multiple horizontal segments depending on the resolution of the point cloud. The mesh also contains two UV maps that map every vertex in the mesh to a different pixel in the color and position textures.

The shader works by first repositioning the vertex based on the corresponding position in the position texture (rgb = xyz) which can be found using the UV map that was created during the creation of the mesh. This happens in the vertex shader. The output is passed on to the geometry shader. In the geometry shader the vertex is extended to four vertices in order to create a quad. The vertices are converted from world space coordinates to screen space coordinates, while made sure that they face the user no matter from what direction they're viewed from. This is passed on to the fragment shader, where the color for each vertex is retrieved from the color texture using the UV map.

4.1.3 Depth Camera Tracker

To properly align the point clouds with the rest of the scene at the RO, the location and orientation (extrinsics) of the corresponding depth camera need to be known. Those are determined by making use of AR markers that are placed on top of the camera and the Vuforia toolkit[5] running on the OSO's Hololens. A QR code is chosen to serve as a marker because of its distinct features, which make it easily tracked by the Vuforia toolkit. Marker holder tools have been made that make it easy to attach a marker to a Kinect V2 in a consistent manner. The OSOTool that runs on the Hololens allows the OSO to scan markers on the fly. This is necessary when the cameras have been moved or when the system has just been started. The OSOTool will display a white border around a marker as soon as it's detected. When the user faces in the same direction as the marker, the border will turn green. This indicates that the marker is ready to be scanned (Figure 4.1). By making the "tap" gesture, the OSOTool will start collecting measurements of the marker's position and rotation during two seconds. During this time, the border will turn red. After the measuring has completed, any outliers are removed and the rest is averaged. This averaged value is considered the true transform of the marker, and it will start being sent to the RO every couple of seconds using a UDP Connector.

[5] `https://developer.vuforia.com/`

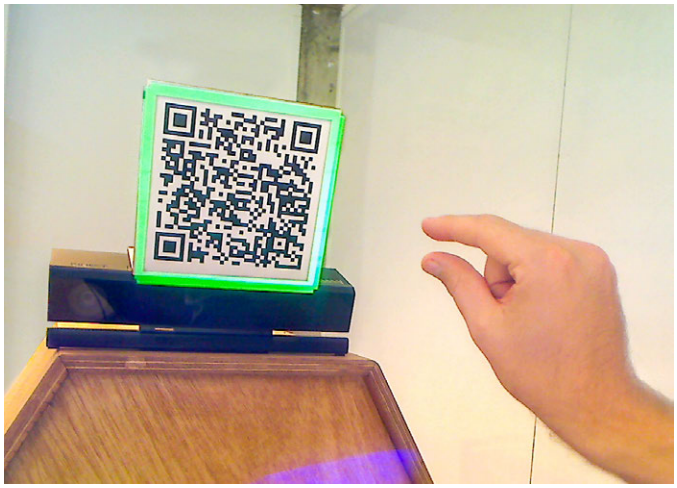


Figure 4.1: A marker with the green border viewed trough the Hololens and the OSO performing the *tap* gesture.

At the RO, marker transforms are received and the (constant) offset between the marker and the camera lens is subtracted from it. Then, the transform is linked to the corresponding *Point Cloud Viewer* so that the frame objects contain the correct camera position and rotation values and the point clouds can thus be rendered at the correct location in the scene.

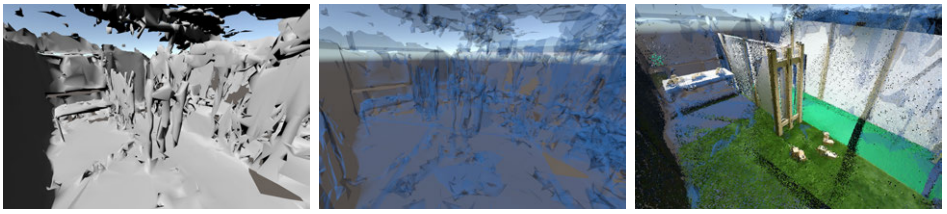
4.1.4 Spatial Mesh Streamer

The spatial mesh is the virtual geometrical representation the Hololens generates of its environment. The spatial mesh is used as a base layer to create a basic understanding of the OSO's environment. To do this, the spatial mesh data is retrieved, serialized and sent from the OSO's Hololens to the RO, where it is deserialized and rendered.

The complete spatial mesh is subdivided into multiple smaller meshes called surfaces. Those surfaces, which are each about 2 m^3, are continuously updated where necessary. A surface usually gets updated when the Hololens gets pointed towards the area in the real world it corresponds to, as the sensors will only then be able to scan that particular area.

Example code from Microsoft's Mixed Reality Toolkit was used as a starting point. The "RemoteMapping" Unity example from this toolkit shows how to: retrieve the spatial mesh data, serialize it, send it, receive it and render it. Although the example already shows how to do each of the required steps, a couple of things still needed to be changed in order to properly integrate it with OpenIMPRESS.

The example code is designed to send all the currently available surfaces and thus complete spatial mesh model from a Hololens to a PC using a



(a) (b) (c)

Figure 4.2: The spatial mesh, rendered with a solid shader (a), with a transparent shader (b) and in combination with a point cloud (c).

voice command. The networking component has been replaced with a UDP Connector so it connects over UDP instead of TCP and automatically find the correct IP and port numbers to connect to. Instead of sending the mesh on command, for OpenIMPRESS the mesh has to be sent automatically in order the keep the spatial mesh at the RO up to date with the one at the OSO. Surfaces of the spatial mesh that are being scanned get updated around every three seconds, so the automatic send interval was set to three seconds as well. Because the complete mesh can be substantially large, retrieving, serializing and sending it usually results in a noticeable stutter. Luckily, it is unnecessary to always send the complete mesh, as only small portions of it get updated at a time. In order to keep track of which surfaces have been updated and thus have to be sent, surface objects have been modified to include a "sent" boolean that is set to false when it is created or updated and to true as soon as it has been sent. As soon as a surface gets removed, its ID will be added to a list that keeps track of which surfaces have been removed. Every three seconds this list is sent to the RO as well. After it's sent it is cleared so that a removed ID doesn't get sent twice.

4.1.5 Spatial Mesh Viewer

The *Spatial Mesh Viewer* receives spatial mesh data from the OSO and creates mesh objects based on this data that are placed in the 3D scene at the RO. A point cloud representation is often used together with the spatial mesh. By default the spatial mesh is rendered using a solid shader (Figure 4.2a), this makes the mesh look like a solid object and one won't be able to see through it. This can become problematic when used in combination with other elements in a scene, especially with a point cloud representation of the same environment. Because the spatial mesh is not very precise nor accurate, blobs of spatial mesh often end up covering parts of the point cloud when the point cloud would actually have give a better impression of how that specific part of the scene looks like. Therefore a transparent shader was chosen for the rendering of the spatial mesh at the RO (Figure 4.2b). This allows the point cloud to be seen even it if it is positioned behind a spatial

mesh surface (Figure 4.2c) and, as an additional benefit, gives in a visual way less meaning to the spatial mesh (as it's not that accurate anyway).

4.2 Transform Management System

There are several (virtual) objects of which the current transforms (position and orientation) have to be shared from the RO to the OSO and the other way around. The *Transform Management System* includes those objects, their virtual representations and the component that manages those objects and makes sure the right transform is applied to the right object.

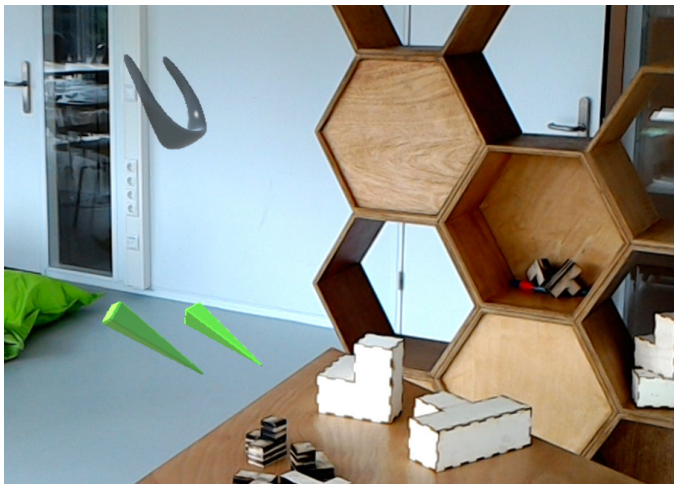


Figure 4.3: The RO's head and controller transforms visualized in the Hololens

Objects of which the transforms are sent from the RO to the OSO include the head and controllers that are tracked by the HTC Vive system (Figure 4.3). Transforms that are sent from the OSO to the RO include the head transform as tracked by the Hololens and the transforms of various AR markers which are tracked by the Vuforia AR plugin running on the Hololens (as further explained in section 4.1.3).

4.2.1 Transform Manager

The *Transform Manager* keeps track of incoming and outgoing transforms. It makes use of a serializer that reads an object's transform and encodes it into a byte stream so that it can be sent over the network using an UDP connector instance. Once received, the data is deserialized to obtain a transform that can be applied to an object again. Using an extra value, the *transformID*, that is passed into the serializer as well, the *Transform Manager* at the receiving side is able to distinguish between transforms from

different objects and can apply a received transform to the corresponding object correctly.

4.2.2 Hololens Transform

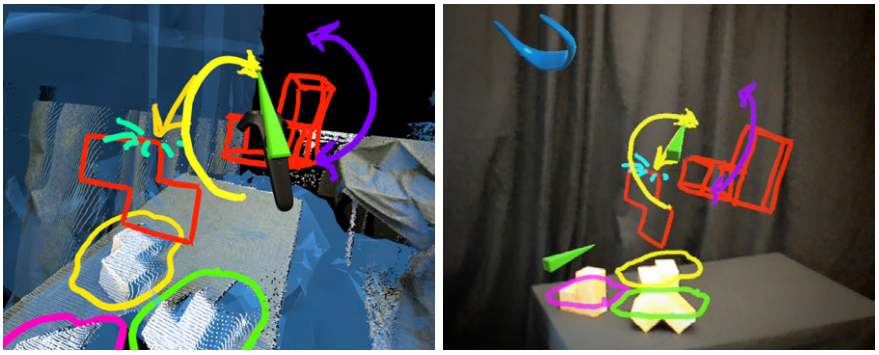
One of the features of the Hololens is that it automatically creates a 3D model of its environment and tracks its own position and orientation in this environment using various built-in sensors that are managed in the background without the user having access to them. The position of the Hololens is used to make sure virtual objects are always rendered to appear at the same place in the environment even when changing perspective. The position is tracked very precisely because any errors or jumps would break the illusion. Besides using this transform for rendering things from the right perspective, it can also be used to visualize the OSO at the RO at places where the depth cameras can't reach. Because of the precise tracking, small head movements are copied as well, which can communicate meaning by themselves and thus can help with collaboration. A visor-like 3D model has been made to represent the location of the Hololens at the RO.

4.2.3 HMD and Controller Transforms

The RO's head and hand transforms are sent to the OSO to allow the OSO to see where the RO is currently located in their own environment. The head and hand transforms are determined by using the transforms from the HMD and the controllers tracked by the HTC Vive system. At the OSO, the same visor model that is used at the RO to visualize the OSO's head transform is used to visualize the RO's head transform. A stretched pyramid model is used to visualize the transforms of the controllers at the OSO. This creates a kind of pointer which makes it easier for the RO to point at things because of its pointy end.

4.3 Annotation System

Annotation functionality is added in the form of a line drawing tool. With this tool, the RO is able to draw lines with varying colors and thicknesses in the OSO's environment. All "brush strokes" the RO makes are sent to the OSO in real-time, where they are displayed as holograms in OSO's surroundings using the Hololens (Figure 4.4). The interface is similar to what is used in the Tilt Brush VR application (Google, 2016a); the RO uses the Vive controllers to paint in 3D space by holding down the trigger button. By pressing the touchpad button a menu appears on which the user can select the size and color of the brush and toggle between making the brush act as an eraser or not.



(a) (b)

Figure 4.4: a) The RO annotating the OSO's environment. b) The annotations as seen by the OSO.

For every type of line drawing event, a serializer and deserializer has been developed. This allows to encode the events and send them over a network, and therefore visualize the drawing somewhere else in real-time. The data is sent and received using instances of the UDP Connector.

The following components are developed as part of the line drawing system.

4.3.1 Line Maker

The *Line Maker* is a component that generates different events for the actions related to line drawing based on user input. Those events include:

- *NewPoint* The NewPoint event gets triggered when a line is extended with a new point. The arguments are the ID of the corresponding line (lineID), the index number of that point on the line (index) and the coordinates of the point in the 3D space (point).
- *LineSettings* The LineSettings event gets triggered when information is available about a line. The arguments are the ID of the corresponding line (lineID), the color of the line (col) and the width of the line (width).
- *RemoveLine* The RemoveLine event gets triggered when a line is removed. The only argument is the ID of the corresponding line (lineID).
- *ResetLines* The ResetLines event gets triggered when all lines should be removed from the scene. It contains no arguments.

The *Line Maker* component reads the position of the tip of the pointer, which acts at the brush, and uses that position as the source of the lines. It contains a couple of variables that can be set by other components; a

color and size variable that should be set to the user selected values, a pause boolean for pausing the drawer when for example the brush moves over a menu and a mode selector that can be switched between *Draw* and *Erase*. While the user holds down the trigger button *Line Maker* will continuously emit new points from the pointer's tip by generating *NewPoint* events. The events will all have the same index number, thus overwriting the previous point, unless the distance is large enough compared to the previous point or the direction of the brush makes a large enough change. In that case the index number will be increased, thus leaving the previous point as a permanent point of the line.

4.3.2 Line Viewer

The *Line Viewer* component receives events from a *Line Maker* component and draws lines according to those events. To actually visualize the lines it uses the *Line Renderer*[6] component that comes by default with Unity. The *Line Renderer* component draws a line between two or more points given in an array, with a given width and material. *Line Viewer* creates a new *Line Renderer* instance for every line that has to be drawn and keeps track of all created instances by putting them in a list.

When the *NewPoint* event is triggered it checks whether a line with the LineID of that new point already exists. If it exists, the new point will be added to that line's points array, possibly replacing an existing point if the new point's index in the array was alreay filled. It the line doesn't exist, a new *Line Renderer* instance will be created to which the new point will be added. When a *LineSettings* event is received *Line Viewer* will search for the *Line Renderer* instance with the LineID from the event, creating a new one if it doesn't exist, and applying the width and color settings to it. The *RemoveLine* event will make *Line Viewer* search for the corresponding *Line Renderer* instance and delete it if it exists. The *ResetLines* event will make *Line Viewer* remove all the current lines.

4.4 Networking System

Several components were developed for connecting the different components of the system with each other over the internet and managing the data streams between them.

4.4.1 UDP Connector

The UDP Connector is the component that sits in between the various software components that need to send or receive data from another remote

[6] `https://docs.unity3d.com/Manual/class-LineRenderer.html`

component and the actual networking library. If a pair of components need to exchange data they'll each need one instance of the UDP connector with the same "socketID", but opposite "pairID's". An instance of the UDP connector component is always given a "socketID", which describes the type of data that this connection is used for. There always need to be two instances of the UDP Connector with the same socketID in a session but with opposite pairID's; A and B. The UDP Connector will independently try to connect with an UDP Connector with the same socketID but opposite pairID. To do this, it communicates with the Matchmaking Server, registering itself and requesting the IP address and port number of the other UDP Connector instance. After receiving this data the UDP Connector will try to directly communicate with the other UDP Connector instance. It will send an empty packet every 2 seconds to let the other instance know the connection is still working. If it didn't receive a packet for more than 5 seconds, it will assume the connection is lost and it will ask the Matchmaking Server for a new port number and IP address.

4.4.2 Matchmaking Server

The Matchmaking Server keeps track of all UDP Connector instances, and makes sure they are all kept up to date with the latest IP address and port number of the UDP Connector instances they have to connect to. When a UDP Connector registers itself it also sends an *unique ID* (UID) based on the device it is running on. With this UID, the matchmaking server can group UDP Connector instances together by device.

"Sessions" can be used to allow multiple OpenIMPRESS environments to run simultaneously while making use of the same Matchmaking Server. A session contains a list of devices that belong together and of which their UDP Connectors are allowed to make connections between each other. In order to make a pair of UDP Connector instances connect, not only will they need to have the same *socketID* and opposite *pairID*, their devices also need to be assigned to the same session.

There are two main advantages of using a matchmaking server instead of trying to connect the various parts with each other manually. Firstly, IP addresses usually don't stay the same over time. Public IP's change because the ISP[7] is often free to assign a different IP to a connection and local IP's often change after the router's DHCP server's lease time has expired which means a new IP will be assigned. This makes it difficult to manually keep track of what IP addresses need to be filled in, let alone what port number to use. The matchmaking server automatically keeps track of both IP addresses and port numbers by just reading the source IP address and port number from the *register* packet that it receives from a client.

[7]Internet Service Provider

Secondly, a matchmaking server allows for connecting clients that are behind a firewall, this is also known as *UDP hole punching* (Ford, Srisuresh, and Kegel, 2005). When manually trying to connect over the internet to a computer or device that is connected through a router, one often finds they have to configure the router's NAT[8] port-forwarding settings to allow data that comes in at a certain port to be forwarded to the wanted device. By using a set-up with a Matchmaking Server, any necessary port-forwarding rules will be automatically set in the background. This works as follows: when a packet is sent out by a device behind a NAT, the NAT will set an exception for incoming data that is received at the port from which the device's packet was sent to be routed back to that same device. This way communication becomes possible without having to set port-forwarding rules manually. As soon as the Matchmaking Server receives a packet from a client, the packet's source port will already be forwarded to the client's private IP by the NAT. This means that all the IP addresses and port numbers the matchmaking server receives will always direct packets to the wanted destination, even when they're sent from a different client after the matchmaking server shared IP and port numbers between two matching clients.

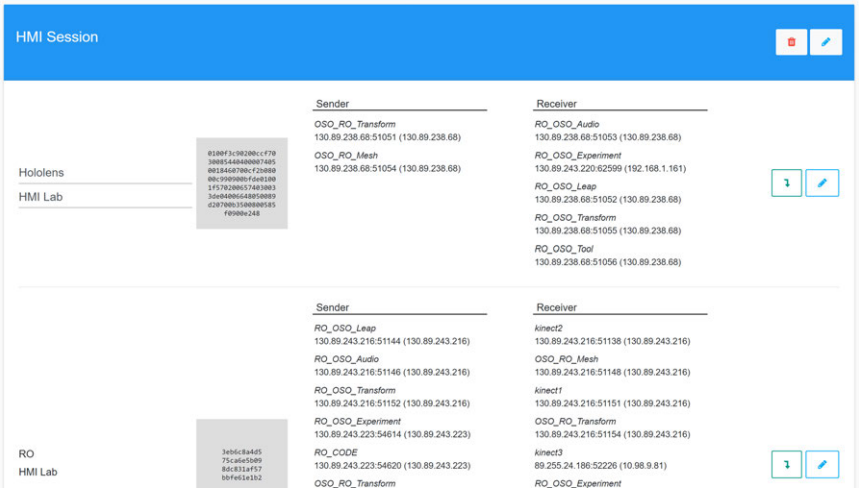


Figure 4.5: The Matchmaking Server web GUI

A web user-interface has been developed[9] for the Matchmaking Server where a user can see which UDP Connector instances have been registered, create sessions and modify the devices that are assigned to them. (Figure 4.5)

[8] Network address translation
[9] Developed by Jan Kolkmeier

4.5 VR Components

The VR Components contain all the elements that add to providing the user a good experience in VR. First of all, the VR system itself of course is part of this, as well as a component that controls the navigation and a component that displays a settings menu and manages the interaction of the RO with it.

4.5.1 VR System

An HTC Vive is used which is an off the shelf consumer VR headset. Besides the headset itself it also includes two controllers. The headset and the controllers are all being tracked using the Lighthouse tracking system (Niehorster, L. Li, and Lappe, 2017), which makes use of two base stations that have to be placed a maximum of 5 meters apart at the borders of the "play area". The Lighthouse tracking system tracks the positions and orientation of the headset and controllers with sub-millimeter accuracy, which means that even small movements by the user are picked up. This is used to make the movements of the images that are shown inside the headset correspond with the movements of the user, and make the visualizations of the controllers in VR match the locations in reality.

4.5.2 Navigation

When using a free viewpoint system a navigation interface is necessary to allow the RO to choose their perspective. Because the RO is viewing the scene in VR the system had to satisfy some extra requirements compared to free viewpoint systems that use a 2D screen solution.

Navigation in VR is a problem that many VR game developers are struggling with and many solutions have already been proposed. Early VR games often used classic game controls like mouse and keyboard or a dedicated game-pad to move the player through the VR environment. This means that the movement that is perceived in the headset does not correspond with the actual movement of the player's body. For many people this resulted in motion sickness and VR game developers started looking for different ways of implementing navigation in VR. With the new generation of VR headsets like the HTC Vive starting to support room-scale tracking, VR game developers were given a couple of meters of walking area that they could let the players navigate in. This meant the navigation problem was solved for games that didn't require the player to navigate more than a couple of meters. To allow players to navigate further developers started making use of a navigation paradigm they call teleportation. It is usually implemented by letting the users point their controller to a free spot in the VR scene around them and instantly transporting them there after they press one of the con-

troller's buttons. This then gives the player a new area to walk around in, confined by the borders of the players physical room. From an experience point of view this is less ideal, as with every "jump" the player makes, they have to reorient themselves at their new area. This also means that players can now navigate way faster and further than they would normally be able to. That is why the teleportation feature is often limited in the distance that it allows the player to travel and the number of times it can be used in succession without waiting.

Another solution that has been used is to have the player stand on a virtual vehicle that can be controlled to navigate around the virtual environment. This reduces motion sickness because all the physical movements of the player still correlate with the observed movements relative to the vehicle.

For OpenIMPRESS a new solution has been implemented that is best described by calling it *dragging*. To navigate, users press and hold down the grip buttons of one controller and move the controller in the opposite direction of where they want to go. It can be explained as "gripping the environment and dragging yourself around in it". Besides using their head to rotate, users can also use *dragging* with two controllers to rotate around the world's up-axis.

A similar control mechanism is implemented in VR games such as *Echo Arena*[10]. This game takes place in a zero gravity environment, where players can float around by grabbing onto virtual objects and pulling themselves forward. The difference with our implementation is that in the game the player can only *drag* if he/she is holding a virtual fixed object, where in OpenIMPRESS the user can use it anywhere.

From our own experience *dragging* resolves the motion sickness issue while still avoiding the "jumps" that occur when using teleportation or the need for a virtual vehicle. The absence of motion sickness can be explained by the fact that all observed movements are still directly linked to the movements of the player, only instead of making use of their feet they are using their hands. The fact that players still have to move themselves and put effort in if they want to be fast also removes the need for unintuitive restrictions that usually are needed when using teleportation.

4.5.3 Settings Menu

A simple menu system has been implemented to allow the RO to change various settings. For now, only one menu has been implemented for changing parameters of the *Line Maker*. It lets the user select the color and thickness of the brush and select whether the brush should draw or erase lines (Figure 4.6).

[10] `https://www.oculus.com/experiences/rift/1369078409873402/`

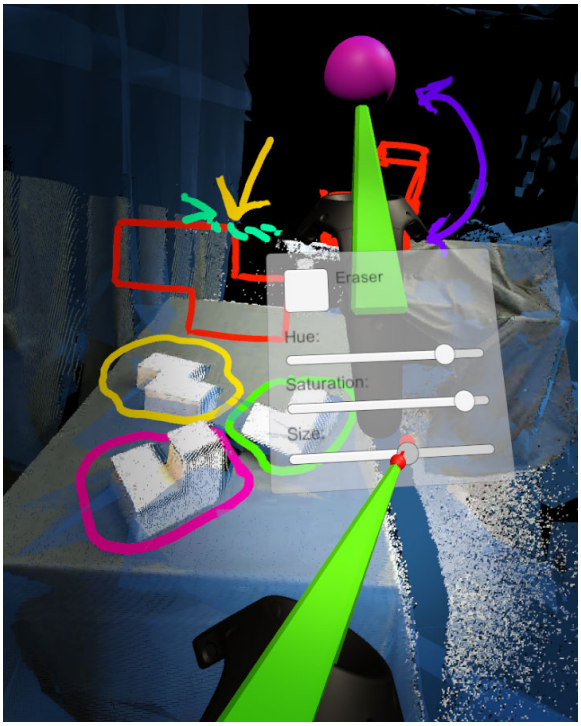


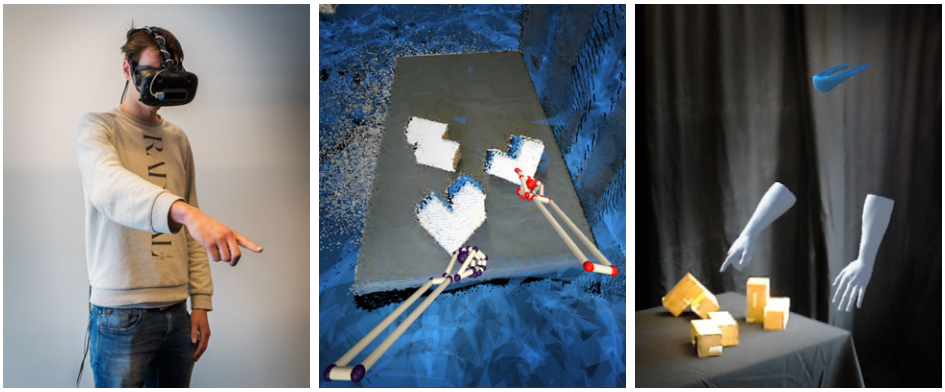
Figure 4.6: The settings menu being used to change the brush size.

The menu is opened when pressing the touchpad on either of the two controllers and will be attached to the controller on which the button was pressed. Settings can then be adjusted by pointing the other controller to the menu and using the trigger button for making selections. The menu was created using Unity's UI system[11] and support for input using the Vive VR controllers was added using a modified version of *Unity-VRInputModule*[12]. *Unity-VRInputModule* was modified to only show the laser pointer when pointing at a menu to make a clear distinction for the user between when the controller is used for menu inputs or for something else.

4.6 Hand Gesture System

Besides using the Vive controllers for making gestures, the RO can also use his or her hands on their own by making use of the *Hand Gesture System*. The difference with using the controllers is that this system doesn't require the RO to hold anything in their hands and it allows for more detailed gesturing as all joints in the hands are individually tracked and visualized. In OpenIMPRESS the hands are captured at the RO and sent to the OSO,

[11] `https://docs.unity3d.com/Manual/UISystem.html`
[12] `https://github.com/wacki/Unity-VRInputModule`



(a) (b) (c)

Figure 4.7: a) The RO using the gesture system to point. b) The RO's hands as seen by the RO and the OSO (c)

where they are visualized as floating holograms (Figure 4.7). This system consists of a *Leap Hand Sensor* to capture the position of the hand and individual fingers and a Leap Hand Viewer to visualize the captured data.

4.6.1 Leap Hand Sensor

The Leap Motion sensor is a sensor that is used to track the hands of a user with high precision without the need of special trackers or gloves on the hands themselves. It makes use of infrared light and two infrared cameras to light up the hands and track them in 3D space. It is often placed facing upwards between the keyboard and the monitor in desktop set-up scenarios. Since the Orion driver was released, Leap Motion started support for using the sensor in VR applications. In this scenario, the sensor is attached to the front of the HMD, so that it can always see the hands of the user when they are held up on front of them. The data the sensor collects is encoded in so called *frames* which can be sent to a viewer so a visualization can be made.

4.6.2 Leap Hand Viewer

The *Leap Hand Viewer* takes hand frame data from a *Leap Hand Sensor* and visualizes it by applying this to a 3D model of a hand (or multiple models, depending on how many hands are encoded in the frame). This way, the RO's hands can be displayed with a highly detailed 3D model in the OSO's Hololens but also in the RO's VR environment so the RO can see what he or she is doing. At the OSO's Hololens, a modified version of the default viewer[13] is used as Leap Motion's original code doesn't work on the Hololens.

[13] `https://github.com/ZhengyiLuo/LeapMotion_Hololens_Asset`

4.7 Verbal Communication System

VoIP support has been included in OpenIMPRESS, allowing the RO and OSO to talk to each other. A *VoIP Manager* component has been developed to do this. This component runs at both the OSO and the RO. It accesses the system's microphone using Unity's built-in Microphone class[14]. Because this class is supported on both the desktop version of Unity and the Hololens, the same code can be used on both devices.

The latest audio samples are fetched from the microphone and are sent to the *VoIP Manager* of the other system. Both the sample frequency and number of channels that the samples are recorded with are sent as well. This meta information ensures that the receiving end can properly play the samples back again. When an audio packet is received, the samples are applied to an audio clip that will be created if it didn't exist yet or will be overwritten with a new one if it wasn't set up with the correct sample frequency or amount of channels. This clip can contain a maximum of 10 seconds of audio samples that will be continuously extended with new samples as they are received. If (after 10 seconds) the array of samples is filled, the samples will be added to the start of the array again. This clip is played back using an Unity *AudioSource* component[15].

[14] `https://docs.unity3d.com/ScriptReference/Microphone.html`
[15] `https://docs.unity3d.com/ScriptReference/AudioSource.html`

Chapter 5

Preliminary evaluation

A simple preliminary evaluation was done after a large part of the system had been implemented. This chapter first explains what the differences were compared to the final system, then the details of the user test that was conducted and finally what the choices are that were made based on the results of the user test.

The research questions of this evaluation are *What is the minimum size of object features that can be correctly observed by the RO?* and *How do users experience using this system for collaboration?*.

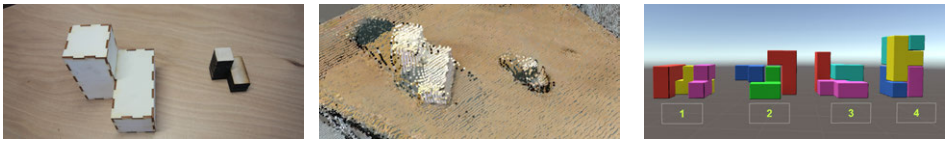
5.1 System differences

When this evaluation was conducted, the OpenIMPRESS system was not yet as completely implemented as described in chapter 4. This is because some features described in chapter 4 are actually based on the results of this evaluation. The purpose of this evaluation was to check what worked and what didn't work at the time to get a better idea of what should be improved or added.

Verbal Communication System The system didn't include the *Verbal Communication System*. For verbal communication a Skype call was started between two computers that were standing in both rooms.

Hand Gesture System The system didn't include the *Hand Gesture System*. Only the controllers were used to make gestures and to point at things.

Annotation System It was not possible for the RO to make annotations in the OSO's space, as the *Annotation System* was not yet implemented.



(a) (b) (c)

Figure 5.1: a) Both sizes of one of the 5 puzzle blocks used in experiment. The shortest edges on the larger objects are 80mm; 36mm on the smaller. b) Point cloud representation of puzzle blocks at RO. c) Target puzzle constructions used during test, visible only to RO.

Networking System No *Matchmaking Server* was yet implemented, therefore IP addresses and port numbers had to be manually entered by the experimenters before an experiment took place.

5.1.1 Method

An informal pilot test has been conducted using the OpenIMPRESS system as it is described above. The test focused on getting a better understanding of the applications that are possible with the current level of detail and what steps still need to be taken in order to increase the usability and allow the system to be used in more applications.

A object construction task was chosen, as it involves collaboration between users and can be controlled for differently sized building blocks. The test involved two differently sized collections of puzzle blocks (Figure 5.1a). Constructions made out of the puzzle blocks were shown in the virtual environment of the RO (Figure 5.1c). The participants were given the task to recreate those constructions at the capture scene. The RO could use gestures to point at the blocks and indicate how to move them and talk to the OSO using a Skype voice connection. The OSO had to manipulate the blocks while being guided by the RO's instructions. After a construction was finished, the next construction was built using smaller pieces. The participants were asked to join in an open interview after a construction with the smallest pieces was built or they gave up. The interview focused on: the overall quality of the system, whether they felt present and the experience of playing as the RO and OSO.

One pair of participants (one female, one male) was invited to engage with the system and both participants completed the test once as RO and once as OSO.

5.2 Results

Both participants were able to recreate the given construction using the large blocks. When using the smaller sized blocks it became more difficult for both participants; the first participant acting as RO did not manage to finish the (correct) construction while the second took longer and resorted to verbal descriptions to describe the construction instead of making use of visual cues and gestures, as the blocks became too small to distinguish.

The participants were impressed with the rendering of the point cloud and had the feeling they were present at the same location. However, they noted how it was sometimes difficult to get on the same page and described a mismatch between the RO's and the OSO's realities, mainly because of misalignment in the point clouds and the lack of detail, especially when using the smaller sized blocks.

When asked about the pointers and gestures, one participant noted that she mainly used the pointers for gesturing instead of pointing. Also, when asked whether they would prefer more detailed methods for gesturing, such as hands, they responded that pointers were good enough.

One participant remarked that in the role of RO, he missed the capability of manipulating objects himself. He suggested to add purely virtual blocks that can be manipulated by the RO and are displayed as holograms at the OSO as well. This would allow the RO to describe the instructions more clearly without having to rely too much on the point cloud's level of detail.

The participants also noted how objects become more clear when held closer and/or parallel to the depth cameras. The first RO asked repeatedly to pick up a block and hold it in front of the camera in order to get a better view while the second RO asked to align the pieces on the table in parallel to the camera's picture plane.

One observation made is that the Hololens' tracking inside the environment may drift over time or get lost completely when its view gets obstructed. This may result in the HoloLens's coordinate system to change, causing inconsistencies between the worlds of the RO and the OSO. A recalibration of the depth sensors is required in this case.

5.3 Discussion

To answer the first research question (*What is the minimum size of object features that can be correctly observed by the RO?*) we see that the participants had no trouble finishing the puzzles using blocks with edges of 80mm but started having trouble when blocks with sides of 36mm had to be used. We therefore assume the minimum size of object features that can be correctly observed by the RO is between 36 and 80mm.

The second research question (*How do users experience using this system*

for collaboration?) can be answered by looking at the participant's responses during the open interview. Overall, it is a positive experience; The users had the feeling of being at the same location and the pointers worked good enough for making their intentions clear. A negative point was the level of detail of the point cloud, which made it sometimes difficult for the users to get on the same page.

Based on the results of the preliminary evaluation some points for improvement can be found. First of all, the overall quality and level of detail of the point-cloud appears to limit the experience in multiple ways. Because the alignment between the cameras is often a bit off, it becomes unclear what the shape of a certain object is when it is being captured by multiple cameras from different angles. The resolution of the cameras themselves seems also to be a limiting factor, judging by the fact that users would hold an object closer to them to increase the level of detail.

There are several solutions that could help improve the point-clouds' quality. To get the point-clouds to align with each other more precisely, an algorithm could try and match overlapping pieces geometry and slightly adjust the calibration, similarly to what Kowalski, Naruniec, and Daniluk (2015) showed in their refined camera pose estimation step. Another solution could be to average out any errors that occur when scanning a marker, which can be done by sampling the camera's calibration AR marker over a longer period of time or to add a second marker that is fixed on a different angle. To be sure about what solutions are actually needed, a more in-depth experiment would be necessary in which the cameras and the AR marker tracking are tested in a controlled environment and a overview is made of where the observed errors in the alignment are actually being introduced.

The level of detail of the point-clouds themselves can also be improved by making use of filters that take into account the various errors that the Kinect v2 camera is known to introduce (Sarbolandi, Lefloch, and Kolb, 2015). Effects like "flying pixels" are relatively easy to correct for by making use of filters like the one described by Kowalski, Naruniec, and Daniluk (2015).

The RO noted that he was lacking the ability to manipulate anything from his side, which he felt was needed to explain certain action more efficiently. This can be a sign of the limitations of the current gesturing system that solely relies on pointers. Ways to convey more meaning could be added to the system. Instead of using controllers, the design could focus more on capturing natural gestures without hardware interference like having to hold controllers. Also, the ability to draw annotations in the scene could help give the RO more possibilities to make their instructions clear.

We noticed how the OSO sometimes loses track of where the RO is located. There are different factors can be the cause of this; the limited field

of view of the HoloLens, the absence of spatial sound and the clearness of the RO's visualization. The limited field of view of the HoloLens' display makes it difficult to find the RO, the OSO often has to scan the whole environment around themselves by rotating their head to find the current location of their partner. Because upgrading the displays of the HoloLens is not within the scope of this project we have to look at possible workarounds. A possible solution could be a guidance system that points the OSO to the RO when the RO is currently not in view. By integrating the audio component into the system, 3D sound effects can be used to make it appear the source of the other user's voice matches the virtual position he or she is currently located. This enables situations where the RO can draw the OSO's attention into their direction by just talking to them.

The current visualization of the RO consists of three relatively small objects that float in the OSO environment; a visor for the head and a pointer for each hand. This minimalistic representation can make it difficult for the OSO to spot the RO. A solution could be to extend the representation with more body parts, which increases the total volume and increases the chances of being seen. Also, the colors of the parts that make up the RO's representation should be easily recognizable; the current color of the visor is black, which in a holographic display is difficult to see. Choosing a brighter color could help with this problem as well.

From the experimenter side of things, we noticed how setting up the system took a considerable amount of time. A big part of that time was spent on updating IP addresses and port numbers in the different components so that data was being sent to the right destinations. Human mistakes often slipped in that would delay the setup process significantly. A solution would be the addition of a system that keeps track of which data streams are available and updates the IP addresses and port numbers automatically.

Based on the findings of this preliminary evaluation, the system has been adapted and extended with extra components. To give the RO more possibilities to express him/herself, the system has been extended with the *Hand Gesture System* and the *Annotation System*. The *Verbal Communication System* has been added to make it easier to set up an audio connection between users and to make locating the RO easier by making use of 3D located sounds. To further assist the OSO in locating the RO's virtual embodiment, an arrow is added that slowly appears when the RO's visor has not been in the OSO's view for 5 seconds and points the OSO into the direction of the RO's current location. Also, the color of the visor is changed from a dark grey color to a light blue color. The *Networking System* has been added to make it easier for researchers to set up the system.

Chapter 6

Main evaluation

This chapter presents the main evaluation that was conducted of OpenIMPRESS. The main evaluation focused on determining how the three telepresence design aspects (view independence, using an immersive display and the use of a virtual embodiment of the visitor) contribute to the performance and experience of a telepresence system in a collaborative context.

Different implementations of those design aspects have been evaluated in studies before. Those studies often had the main focus on only one of the three design aspects, with the remaining two aspects only partially or not at all implemented. Immersive VR displays have for example been used in telepresence systems before, but using a fixed viewpoint (Amores, Benavides, and Maes, 2015) or with view independence only limited to rotational motions (Gao et al., 2016). Also, full view independence has been used in telepresence before, but not with any remote embodiment (Tait and Billinghurst, 2015) or an immersive VR display. Giving the visitor a (virtual) remote embodiment is often tested with a view locked to the visitee's perspective but not with full view independence.

A collaborative task was chosen as the main context for this evaluation, because it is one of the main use cases for telepresence systems and it is a context in which the system's various components play a role.

To gain a more complete understanding of how these design aspects affect the performance and experience of telepresence systems, we perform three separate experiments to individually test them. The research question of this evaluation is composed as follows:
RQ2. *How do the three telepresence system design aspects influence the performance and experience of the users in a collaborative setting?* The research question is split up into the following three subquestions:

RQ2.1. *How does view independence influence the performance and experience of the users in a collaborative setting?*

RQ2.2. *How does using an immersive display influence the performance and experience of the users in a collaborative setting?*

RQ2.3. *How does giving the visitor a virtual embodiment influence the performance and experience of the users in a collaborative setting?*

We will first discuss how the three design aspects are expected to affect performance and different dimensions of the experience. Then, the setup of the system and design of the study are presented. Finally, the results are discussed and the research questions answered.

6.1 Performance

We expect all three telepresence design aspects (section 1.1) to increase the performance of the users in a collaborative context. The performance of the users is described by the the time that it takes to complete a task and the amount of errors that are made. A lower time equals a higher performance and a lower amount of errors equals higher performance as well. The exact performance measures that are used are explained in section 6.7 after the task has been explained in more detail.

View independence By giving the RO an independent view it is expected that the time required to complete a task decreases. This is because less effort needs to be spent on adjusting the RO's view which decreases the total time spent on the task. Less errors are expected to be made as well, because it is easier for the RO to maintain an overview of the current situation, he/she is therefore more effective in completing the task without errors.

Immersion Using an immersive display at the RO is expected to decrease the amount of time that is required and the amount of errors that are produced during a task. The lack of certain spatial cues when not using an immersive display, like depth and a wide field of view, make it more difficult for the RO to support the OSO in tasks that especially require the ability to recognize three dimensional features of objects and require the RO to have an overview of a relatively large area. For example when using a 2D monitor to determine how far something is away in a 3D environment. Allowing the RO to recognize those cues is expected to decrease the time that is required and the amount of mistakes that are made.

Embodiment Giving the RO a virtual embodiment is expected to decrease the time that is needed and the amount of errors that are made during a task. The embodiment supports the RO when explaining him-/herself by increasing the amount of channels that can be used for communication. This decreases the amount of time that is required for communication while increasing the effectiveness and thus decreasing the amount of errors.

6.2 Experience

The three telepresence design aspects (section 1.1) are each expected to affect a different set of dimensions of the users' experience. Where giving the RO an embodiment to express him/herself with gestures towards the OSO is, for example, expected to help the communication between the two, letting the RO use an immersive diplay is expected to have more effect on his or her feeling of presence in the remote environment. We will first present the different dimensions of experience that are relevant and afterwards, explain for each design aspect which dimensions are expected to be affected and why.

Usability refers to the extent to which a system can be used in a particular task by the intended users regarding effectiveness, efficiency and satisfaction.

Spatial presence is a measure of the feeling of being in a certain environment. People can feel spatially present in real environments although the term is most often used to describe the feeling in a virtual environment. It is closely related to immersion, although the two terms are not interchangeable. Spatial presence is a subjective measure based on the user's experience, whereas immersion is a variable of the technology. A technology that is more immersive tends to create a stronger feeling of presence at a user (Schubert, Friedmann, and Regenbrecht, 2001). "The definition implies that an individual perceives and experiences media stimuli almost in such a way as if they were real, even though they are not" (Hartmann et al., 2015).

Co-presence is a measure for how aware the users are of each other. It focuses on whether the user feels like he or she is not alone and is aware of the other but also on whether the user feels like the other is aware of them.

Perceived message understanding is a measure for how well the users think they understand messages from the other user and how well the users think the other user understands their messages.

Perceived behavioral interdependence is a measure for how much the user feels like his or her behavior affects the other user's behavior and how much the user feels like his or her own behavior is affected by the other user's behavior.

How those dimensions of experience are exactly measured is explained in section 6.7. For each of the three design aspects, we will now describe which of the dimensions of experience we assume are affected and in what way we assume they are affected.

View independence With view independence we describe the aspect of allowing the RO to independently change the perspective he or she has on the OSO's environment.

We expect the *usability* of the system to increase when allowing the RO to view the scene from a different perspective than only the OSO's. In a scenario without view independence, what the RO sees is determined by the OSO. Therefore, the RO has to communicate to the OSO if he/she wishes to view the scene from a different perspective. *With* view independence, the RO can look at what is needed without the help of the OSO, this decreases the time that is spent on system related communication, leaving more time to focus on the task at hand, increasing the efficiency and therefore also the usability.

We expect the user's feeling of *spatial presence* to increase when enabling view independence. Especially when used in combination with VR, view independence allows the RO to experience the virtual representation of the OSO's environment in a way that is more natural and more closely resembles how one would navigate through an environment in reality.

We expect the RO's feeling of *co-presence* to increase with view independence. Without view independence, the RO won't be able to see the OSO, as the RO will be observing the scene through the OSO's perspective. By adding view independence, the RO will now be able to observe the OSO as if he or she is standing in the same room, which makes the RO more aware of the OSO and thus increases the co-presence.

We expect the *perceived behavioral interdependence* of both users to decrease when view independence is added. Without view independence, there is a strong connection between the user's behaviors. Especially the RO's behavior is highly dependent on the OSO's, as the RO only sees what the OSO sees and has to base his/her behavior on that. Therefore, when allowing the RO to view the scene independently from the OSO, their behaviors are also expected to become more independent.

Immersion With immersion we describe the aspect of visualizing the virtual representation of the OSO's environment using immersive viewing technologies like a VR display instead of using a 2D desktop monitor.

We expect the use of immersive displays to to have positive effect on the *usability* for the same reasons we expect them to increase the user's performance. As mentioned above: it takes less effort and feels more natural to engage with a virtual environment when using this technology. This makes performing tasks and helping others in the same environment more effective and efficient, which makes the system more usable.

We expect *Spatial presence* to increase when using an immersive viewing technique, as the immersiveness of a system is one of the main contributors to a user's feeling of spatial presence.

We expect the feeling of *co-presence* to increase, mainly at the RO. The immersive view on the remote environment will make the RO less distracted by his/her own environment and make it easier for the RO to locate people in the remote environment, therefore he/she becomes more aware of them and the feeling of co-presence is increased.

We expect the *perceived message understanding* to increase when system is more immersive. Because an immersive viewing technique makes it easier for the RO to locate people and makes him/her feel like being in the same environment as the OSO, communication is expected to become easier and therefore increasing the perceived message understanding.

Embodiment With embodiment we describe the aspect of giving the RO a virtual embodiment in the OSO's environment. This embodiment allows the OSO to see the where the RO is currently standing and the direction he/she is currently looking in. It also allows the RO to make gestures, point to things and express him-/herself with basic body language.

By adding an embodiment of the RO, we expect the *usability* to increase. The ability to make use of gestures and the addition of information about the what the OSO is currently focusing on will have the result that less verbal instructions are required to make a point come across and therefore the efficiency of the system is increased which makes it more usable.

Gestures have been shown to support visuo-spatial aspects of the speaker's meaning (Wu and Coulson, 2007) and support speech comprehension especially in difficult communication conditions (Obermeier, Dolk, and Gunter, 2012). We therefore expect that adding the ability for the RO to make use of gestures to increase the *perceived message understanding*.

We expect that giving the RO an embodiment will increase both user's feeling of *co-presence*. The virtual embodiment of the RO will make the OSO more aware of the RO and direct more attention to him/her. This will make the RO also feel like the OSO is more aware of him/her, thus increasing the feeling of co-presence of both users.

6.3 Hypotheses

Based on the expectations described in section 6.1 and section 6.2, hypotheses have been formulated. For each of the three subquestions the hypotheses are listed below.

RQ2.1. ***How does view independence influence the performance and experience of the users in a collaborative setting?***

H1.1 View independence decreases the total task duration.

H1.2 View independence decreases the amount of errors.

H1.3 View independence increases usability for the RO.

H1.4 View independence increases spatial presence of the RO.

H1.5 View independence increases co-presence for the RO.

H1.6 View independence decreases perceived behavioral interdependence of the RO.

H1.7 View independence decreases perceived behavioral interdependence of the OSO.

RQ2.2. ***How does using an immersive display influence the performance and experience of the users in a collaborative setting?***

H1.1 Immersion decreases the total task duration.

H2.2 Immersion decreases the amount of errors.

H2.3 Immersion increases usability for the RO.

H2.4 Immersion increases spatial presence for the RO.

H2.5 Immersion increases co-presence for the RO.

H2.6 Immersion increases perceived message understanding of the RO.

H2.7 Immersion increases perceived message understanding of the OSO.

RQ2.3. ***How does giving the visitor a virtual embodiment influence the performance and experience of the users in a collaborative setting?***

H3.1 Giving the visitor a virtual embodiment decreases the total task duration.

H3.2 Giving the visitor a virtual embodiment decreases the amount of errors.

H3.3 Giving the visitor a virtual embodiment increases usability for the RO.

H3.4 Giving the visitor a virtual embodiment increases usability for the OSO.

H3.5 Giving the visitor a virtual embodiment increases perceived message understanding of the RO.

H3.6 Giving the visitor a virtual embodiment increases perceived message understanding of the OSO.

H3.7 Giving the visitor a virtual embodiment increases co-presence of the RO.

H3.8 Giving the visitor a virtual embodiment increases co-presence of the OSO.

6.4 Task

The task for the user experiment was selected using the following requirements.

Collaborative The task needs to be collaborative by nature and require the participants to work together. It should be impossible to solve the task without any communication between the two participants. Information that the OSO requires to complete the task should for example be only given to the RO, so that only by collaborating the task can be completed.

No prior knowledge possible The task should remove the possibility for people with prior knowledge of the task to have an advantage. For example: imagine a task where the OSO has to construct an engine but only the RO has the manual, if the OSO turns out to be a car mechanic any remote help from the RO won't be useful anymore; the OSO will already know what to do. This would mean that no collaboration is necessary and create invalid results.

Encourages the use of OpenIMPRESS features The task should provide reasons to use the features of OpenIMPRESS. In particular, the task(s) should contain a navigation component that makes use of the wide area scanning capabilities, an object recognition component that makes use of the detailed scanning capabilities and an object manipulation component that makes use of the verbal and non-verbal communication capabilities of the system.

Fun With a task that is regarded fun it will become easier to recruit participants to run the study with. The hypothesis is that a fun experiment description will lower the threshold for people to participate and leaving participants with a positive experience will increase the likelihood that they recommend other people to participate as well.

An escape room like environment was used for the experiment in which the participants were tasked with retrieving a four-digit code by solving three escape-room-like puzzles. This task was chosen as it fulfills all the requirements that were mentioned before. Escape rooms are designed to

(a) The escape room where the OSO is located.

(b) The room nearby where the RO is located.

Figure 6.1: The two rooms that are used in the experiment.

be a collaborative experience, as mentioned by Nicholson (2016): "Escape rooms require teamwork, communication, and delegation". They are also being played by males and females equally which suggests that there is not strong bias for a certain gender. They are becoming a popular leisure time activity, which is expected to easily draw people to participate in the study. Because the tasks are presented as puzzles in a playful context, they can be designed to be more abstract in order to disconnect them from any real-world knowledge and remove any bias that this could give. This design freedom also makes it easier to incorporate aspects that encourage the use of OpenIMPRESS features like navigation, object recognition and object manipulation.

One participant, the on-site operator (OSO), is located in the escape room (Figure 6.1a) and the other participant, the remote operator (RO), is located in a room nearby (Figure 6.1b). From this nearby room, the RO is remotely present in the escape room to help the OSO complete the tasks.

The escape room consists of the following three tasks that need to be completed in the order in which they appear. A map of the escape room is shown in Figure 6.2, depicting where each task is located.

1. **Navigation Task - Radiation beam avoidance** The RO is tasked to lead the OSO through a maze of virtual radiation beams that only the RO can see while making sure the OSO's head never touches one of the beams. This puzzle can be thought of as a laser maze, but with the exception that only a remote partner can see the lasers. Also, to increase the visibility and difficulty, beams with a thickness of around 25 cm are used instead of thin laser rays.

2. **Recognition Task - Block shape collection** Six blocks with different shapes are lying on the ground and the OSO needs to collect three specific ones to use in the next step. Only the OSO is able to see a virtual display depicting which blocks need to be collected. There-

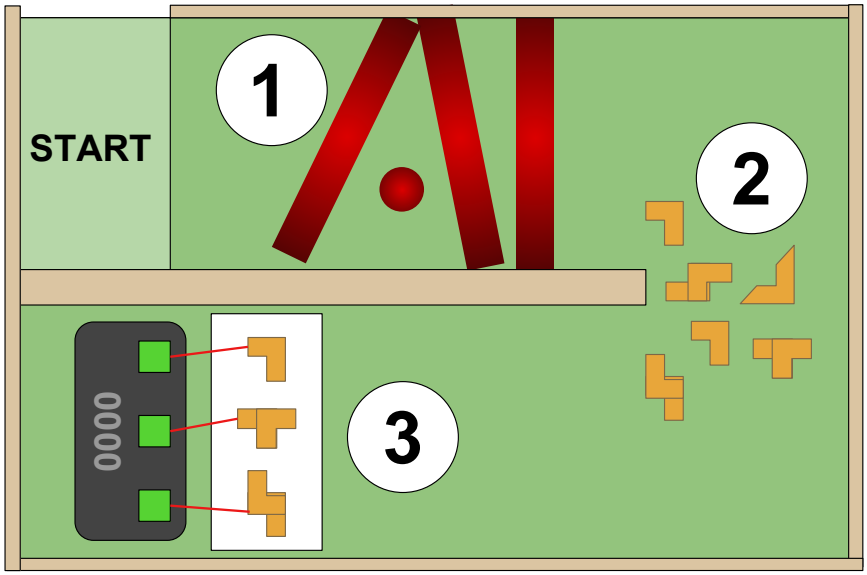


Figure 6.2: Map of the escape room and the relative locations of the three tasks. 1) Radiation beam avoidance task; 2) Block collection task; 3) Block alignment task.

fore the RO needs to recognize the correct blocks on the ground and communicate to the OSO which ones he or she needs to collect.

3. **Manipulation Task - Block alignment** The three blocks that are collected during the previous puzzle will start emitting a virtual laser beam that only the RO can see once they are placed on top of the table. The blocks need to be positioned in such a way that the lasers point to the markers behind the table. To do this, the RO has to communicate to the OSO how to manipulate the blocks. Once each of the markers are lit by a laser beam, a part of the code will be displayed on the screen.

In the instructions to the participants, no specific priority on speed or accuracy was given. When asked, the participant was instructed to complete the experiment in the way that he/she felt the most comfortable with. A video of a pair of users completing all three tasks can be found on YouTube[1].

[1] `https://youtu.be/X1dxVtRf0ws`

6.5 System set-up

6.5.1 Configurations

This evaluation is based on a comparative design, where the performance and experience of the users are compared between a system with all three design aspects implemented and with one of them, respectively, not implemented. To do this, four different system configurations have been prepared; one with all the design aspects implemented and three in which every time a different design aspect is not implemented individually. We will now present each configuration separately for more details.

Baseline In the *baseline* configuration the system is configured in the same way as it it described in chapter 3. The RO wears the VR headset, is free to walk around and its head position and hands are visualized as holograms at the OSO.

No embodiment The *no embodiment* configuration disables the holographic representation of the RO at the OSO. The visor, representing the RO's head position, and the visualization of the RO's hands are removed. The visualization of the hands that the RO would see in the virtual representation of the on-site environment are disabled as well. This effectively removes any visual representation of the RO in both the view of the RO and the OSO.

Non-immersive The *non-immersive* configuration replaces the VR headset with a conventional 2D monitor, keyboard and mouse set-up, similar to the set-up shown by Tait and Billinghurst (2015). Instead of navigating by physically walking, the RO has to use the same mouse and keyboard controls that are often found in first person shooter video-games; the arrow keys are used to translate the view to a different position and the mouse is used to rotate the view in a different direction.

The RO can still use his or her hands for pointing or gesturing; the leap motion hand sensor that is otherwise attached to the front of the VR headset is now positioned pointing up below the keyboard so the hands are detected when the RO holds them up in front of the screen. This may introduce extra difficulty, as the RO now has to switch between using his/her hands for navigating and using them for making gestures.

Dependent view The *dependent view* configuration removes the ability for the RO to independently walk around by fixing the RO's viewpoint to the viewpoint of the OSO, similar to what Kasahara and Rekimoto (2015) did with *JackIn head* or what Gao et al. (2016) did in one of the conditions of their research experiment.

This configuration however, introduces a mismatch between the RO's movements and the movements that are perceived in VR, as the movements in VR are now copying the movements of the OSO instead of the RO. This mismatch between perceived movement and actual movement is known to introduce nausea (Groen and Bos, 2008). Therefore, a couple of precautions are made in order to reduce this effect. First, the RO is asked to sit down in a chair and is recommended to keep head movements to a minimum as they won't affect the RO's view anyway. Secondly, the movements of the OSO are filtered with a low-pass filter before they are copied to the RO. This removes any abrupt movements or shocks that are likely to make the RO nauseous. The RO now smoothly follows the OSO's movements instead of experiencing every little bump.

In an attempt to further reduce sickness, self-evaluated experiments have been done where the RO's movements were retained to some degree. This was done with the intent to keep a connection between perceived movement an actual movement. Fast movements by the RO were copied but any introduced deviation from the OSO's position would be smoothly corrected by pushing the RO's view back to the OSO's view again. It turned out that this made the problem worse, as this promoted to look around yourself but gets confusing when it doesn't work as expected. Therefore it was decided to turn this feature off and tell users to not move their head when they are using the system in this configuration.

6.5.2 Modifications

Besides the different configurations, some other modifications have been made to OpenIMPRESS specifically for this experiment as well. Certain elements have been disabled while others have been added compared to the system explained in chapter 4.

Hololens Alignment Borders Aligning the Hololens properly on a participant's head can sometimes be a difficult task, especially when nothing is being displayed in the Hololens's view. To help with this problem, a red border that touches the borders of the Hololens's field of view has been added. While helping the participant put on the glasses, the researcher can now verify whether the glasses have been installed properly by asking whether the participant can all the corners of the red rectangle. If the participant can't see all the corners, the rectangle serves as a guide for moving the glasses in the right direction. The border is only visible during the setup part of the experiment. When the experiment starts, the border disappears so it doesn't interfere with the rest of the experiment.

Radiation Beams For the first task of the experiment, the OSO has to navigate through a maze consisting of radiation beams that only the RO can see (Figure 6.3). Three dimensional beams have been designed and incorporated into the RO's virtual representation of the on-site environment. The beams are constructed out of two cylinders with different diameters, one inside the other, that rotate in opposite directions around their own axis. Noise textures have been applied to both cylinders in order to make them more visible.

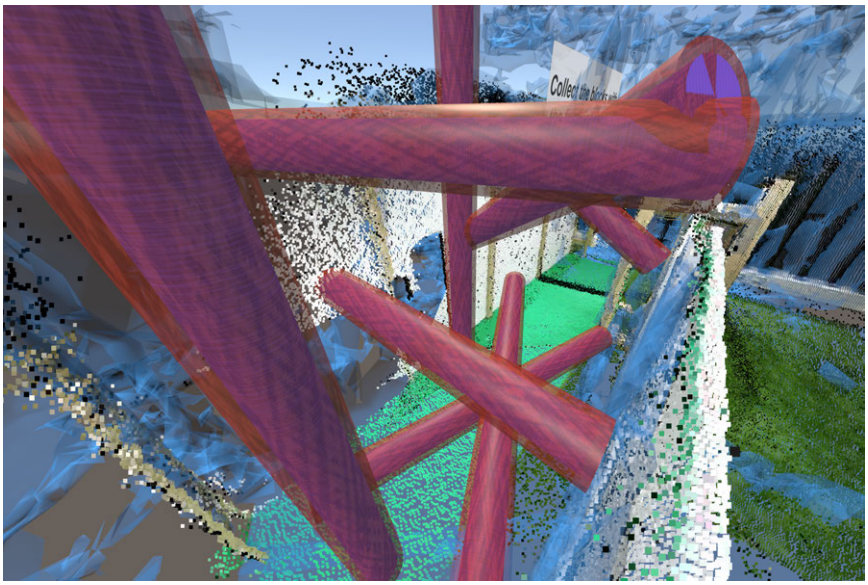


Figure 6.3: One of the two radiation beam maze layouts.

A collider has been added so that collisions with the OSO's Hololens can be detected. Six of those beams have been put together in different orientations to form a maze layout. Two layouts have been made in total; one for the first round and one for the second round in order to reduce the learning effect.

OptiTrack An Optitrack system is used to track the position and orientation of the blocks in the escape room. The blocks are covered with unique patterns of OptiTrack's precision sphere markers (Figure 6.4), so that the OptiTrack system can identify and track the individual blocks. The infrared light from the OptiTrack camera's was found to interfere with the Kinect V2, which would be unable to detect any depth in parts of the scene that are covered by a OptiTrack camera. This was solved by turning down the OptiTrack cameras' infrared LED light intensity[2].

[2]LED value was set to "2" using the "Devices pane" in the Motive software (`https://v110.wiki.optitrack.com/index.php?title=Devices_pane`)

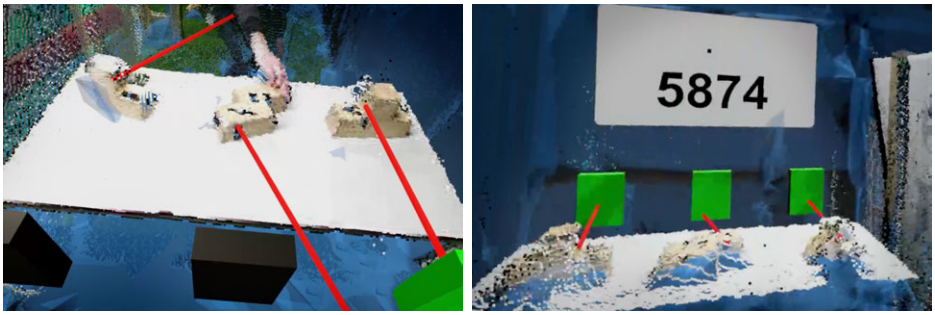
Figure 6.4: Blocks with OptiTrack markers attached to them.

The OptiTrack Unity plugin[3] was used to receive the tracking data from OptiTrack's Motive software in Unity.

Block Lasers and Markers The tracking data from the OptiTrack system is aligned with the coordinate system that is used in the rest of OpenIMPRESS by manually offsetting the coordinates that are received from the OptiTrack system in Unity. This way, the locations of the blocks as tracked by the OptiTrack system align with the locations where the RO sees the blocks. Virtual laser beam objects are created and linked to the OptiTrack tracking data so that each block emits one laser beam (Figure 6.5). The laser beams check whether they hit one of the three marker objects using Unity's raycast functionality[4]. A marker object is a black rectangle that participants are told to point the laser towards. A soon as a laser is pointed towards one, it will gradually change color until it has become green. As soon as all three markers are green, the safe code is revealed. In Figure 6.5b the three marker objects are shown while being lit by the lasers.

Code Revealer Next to displaying the code and the current state of the marker objects to the RO, the OSO is also shown those things. To do this, a *Code Revealer* component has been added, which receives data from the RO's computer and displays the current state of the laser markers and the

[3] `https://v20.wiki.optitrack.com/index.php?title=OptiTrack_Unity_Plugin`
[4] `https://docs.unity3d.com/ScriptReference/Physics.Raycast.html`



(a) The RO is adjusting one of the blocks' orientation.

(b) All laser markers are lit and the correct code is displayed to the OSO.

Figure 6.5: The blocks, lasers and marker objects as seen by the RO.

safe code on a monitor in the escape room (Figure 6.6). This component is implemented in Unity and runs independently on a separate computer. The output is displayed on a monitor that is positioned behind the table of the block alignment task. This way, the OSO gets direct visual feedback when a block has been aligned properly and can see the code once all three have been aligned.

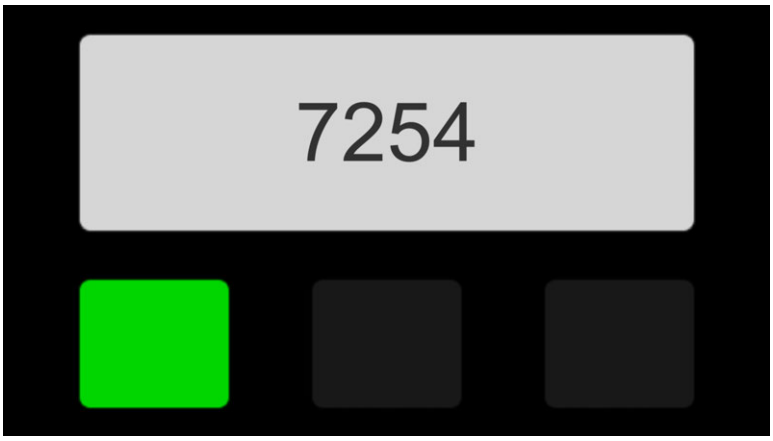


Figure 6.6: The code revealer interface as seen by the OSO on a monitor.

Experiment Management System The *Experiment Management System* keeps track of the current state of the experiment and makes sure all components are set to the right state when necessary. It is integrated into the software that runs at the RO and communicates with the other remote components using the *Networking System*. It manages the following tasks: It sends a message to the HoloLens to disable the alignment borders when

the experimenter presses the special key combination to start the experiment and a message to enable them when the stop combination is pressed. It sends the safe code to the code revealer as soon as all three laser markers are lit and a message to hide the code when the system starts, as well as messages about whether each laser marker is currently lit or not. It also sends a message to the HoloLens whenever the HoloLens' transform collides with one of the beams from the radiation beam avoidance task. When this happens, a red flash is displayed on top of both the RO's and the OSO's views in combination with a *zapping* sound to warn them that the OSO hit a radiation beam.

6.5.3 Data Logger

Logging functionality was added to OpenIMPRESS which captured all data that got sent and received at the RO, which encompasses all the streaming data in the system. This is useful because it provides a standardized way for determining the duration of an experiment by taking the difference between the timestamp of the start command and the end command. This is also useful for future research because the data contains a lot of information about the interactions participants had, like head positions, hand gestures, verbal communication and 3D scans of the room.

6.5.4 Layout

The layout of the physical components at the on-site location is shown in Figure 6.7. The dimensions of the escape room area are 3,0m X 5,6m.

Three Kinect V2 depth cameras are installed at the borders of the room to capture the room from above. The first is installed at the end of the maze, pointing to the start location. This way, when the OSO is walking through the maze, he/she will be facing the camera which cleared image for the RO. The second camera is pointing towards the blocks that are lying on the ground. Because the first camera also partially has the blocks in its view, will be visible to the RO even when one of the cameras gets covered by the OSO. The third camera is positioned behind the table of the alignment task to point down to it from the top. It captures what is positioned on top of the table as well as who is standing in front of it. This allows the RO to get a good overview of the blocks and how the OSO has to manipulate them.

Six OptiTrack cameras are installed around above the table in order to track the location and orientation of the blocks using the reflective markers that are attached to the blocks.

There are two PC's located outside of the escape room. *PC 1* processes the data from the first two Kinect cameras and streams them to the RO using two instances of the *Depth Camera Streamer* (subsection 4.1.1).

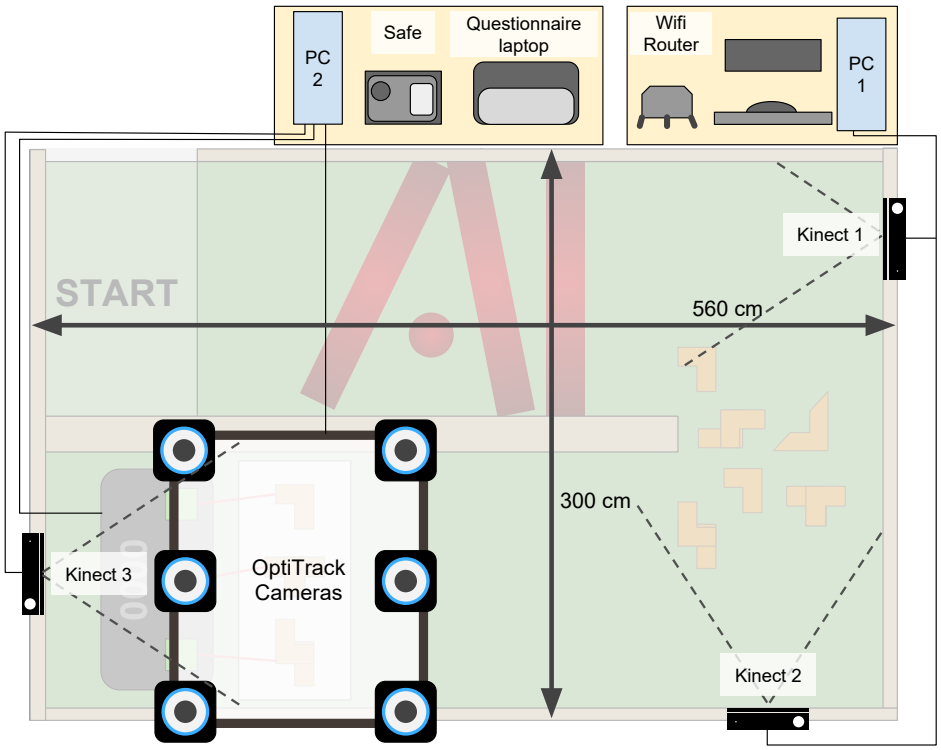


Figure 6.7: A map of the system's technical components in the escape room.

PC 2 processes and streams the data from the third Kinect camera and runs an instance of OptiTrack Motive which receives and processes the data from the OptiTrack cameras and calculates the transforms of the blocks based on that. It also runs the *Code Revealer* software component, which is shown on the connected display in the escape room.

Before a experiment run begins, the OSO sits behind the first table and fill in the questions on the questionnaire laptop. Next on the same table, the safe is located which the participants can open at the end of the complete experiment.

A wireless router is located on top of the table with *PC 1*. The router is connected to both PC's and the internet with a gigabit Ethernet connection and to the HoloLens with a wireless 802.11ac connection.

At the remote location the layout is relatively simple compared to the on-site location. A desktop PC is set up in on corner of the room on a table with a monitor, mouse and keyboard and the rest of the room is used as the play area for the VR setup. The play area is positioned in such a way that there is enough physical space to allow the RO to freely walk inside the virtual representation of the on-site environment without bumping into any physical obstacles.

6.6 Study design

The experiment is split into three sub experiments; an *independent view*, an *immersive display* and a *virtual embodiment* experiment.

Each experiment is designed with a single within subject variable "configuration" at one of two levels: *baseline* and *test*. This means that each pair completes the escape room twice. Once in the *baseline* configuration, and once in the *test* configuration of in the respective experiment. The order of "configuration" is counterbalanced.

In the *independent view* experiment, the two levels of "configuration" thus are the *baseline* (independent view) configuration and the *dependent view* configuration. In the *immersive display* experiment, the two levels of "configuration" thus are the *baseline* (immersive display) configuration and the *non-immersive* configuration. In the *virtual embodiment* experiment, the two levels of "configuration" thus are the *baseline* (virtual embodiment) configuration and the *no embodiment* configuration (subsection 6.5.1).

Ten pairs participated in each experiment, which results in 60 participants who participated in total. Participants were recruited using messages on social media and mouth to mouth communication. Most of the participants are students or employees at the University of Twente. Participants were asked to sign up, preferably in pairs, using an online form. If a participant signed up alone, he or she would be paired with another participant that signed up alone.

The group consisted of 31 males and 29 females, the mean age is 25.9 years (sd: 9.0, max: 57). 8 pairs are male-male, 7 pairs are female-female and 15 pairs are male-female. 83.3% of the participants report to have tried VR once or twice before and 8.3% says to use it frequently where 66.7% report to have tried AR once or twice before and 3.3% reporting to use it frequently.

One pilot test for each of the three experiments has been conducted before the actual experiments started. Based on those pilot tests it was found that there was still confusion about some of the tasks' details. This is why next to written task instructions, the RO and OSO also receive an explanation about the tasks from the experimenter directly.

6.7 Measures

For getting an indication of the performance we are looking at the duration and the amount of errors. The duration is determined automatically by subtracting the timecode of the "start data packet" which indicates the start of the run from the timecode of the "code reveal data packet" which was sent when the participants finished the last puzzle thus indicating the end of the run. The amount of errors are described by the amount of times the

OSO hits one of the radiation beams in the radiation beam avoidance task. This is measured by counting the amount of "radiation hit data packets" that are sent during a particular run.

For getting an indication of the users' experience we look at usability, spatial presence, co-presence, perceived message understanding and perceived behavioral interdependence.

We measure usability using the *System Usability Scale* (SUS) (Brooke et al., 1996). It consists of 10 questions on how the user perceived the system; whether he/she would have needed support or training, and the overall complexity of the system. The answers are used to calculate a single score from 0 to 100 that rates the overall usability of the system using the scoring system detailed by Brooke et al. (1996).

We use the *igroup presence questionnaire* (IPQ) (Schubert, Friedmann, and Regenbrecht, 2001) to measure the participant's spatial presence. This questionnaire consists of 14 questions on a 5-point Likert scale, of which 6 relate to spatial presence. To compute the spatial presence score, the scores from negative questions are inverted after which the average of those 6 questions' scores is calculated.

We use the *Networked Minds Measure of Social Presence* (Harms and Biocca, 2004), a questionnaire focusing on social presence, to measure co-presence, perceived message understanding and perceived behavioral interdependence. It consists out of 36 questions in total, providing the following six subscales: co-presence, attentional allocation, perceived message understanding, perceived affective understanding, perceived emotional interdependence and perceived behavioral interdependence[5].

Before starting the experiment, participants were also asked to fill in the *Immersive Tendencies Questionnaire* (Witmer and Singer, 1998) which was used to "measure differences in the tendencies of individuals to experience presence". Specifically, a version of this questionnaire that was revised by the UQO Cyberpsychology Lab (2004) was used. It consists out of 18 questions and provides scores on the following four subscales: tendency to become involved in activities, tendency to maintain focus on current activities, tendency to play video games and tendency to become emotionally affected.

[5]We take the average of the item scores associated to the respective subscales. It should be noted that in Harms and Biocca (2004), they only report positive factor loadings for the respective items, although some items are clearly phrased as the inverse. We thus have inverted the items (7, 8, 11, 12, 17, 18, 21 and 22) before computing the scores for the purposes of this study

6.8 Experiment procedure

Before the experiment begins, the experimenter starts with selecting one of the three experiments, selecting whether the baseline or the altered condition is presented first and whether the first or second maze layout it presented first. This is done using a predefined sequence where, with maze 1 presented first, the three experiments are first done with the "baseline" configuration first and then with the "test" configuration first after which everything is repeated with maze 2 presented first.

A table is made to help the experimenter keep track of where in this sequence the user tests are currently positioned, this table can be seen in section A.1. This appendix also contains a random sequence of 1's and 0's that is generated to ensure the participants get the RO and OSO task assigned randomly. Before every experiment the next number in the sequence is selected. If the number is 0, the first person stepping through the door would be assigned the role of OSO. If the number is a 1, the first person is assigned the RO role.

A checklist is made that the experimented can use during the experiment (section A.2). It contains fields to write down the names of the participants, and fields to check off each step that the experimenter has to take to complete an experiment. For every step it contains information about the location where it should take place and the text the experimenter should say or the action the experimenter should take.

When the participants arrive, they are welcomed and led into the remote location room. After a basic introduction about what is going to happen, the OSO is asked to follow the experimenter to the on-site location room. Both participants are asked to read and fill in the consent form (section A.5) and fill in the pre-experiment questionnaire on the computers in their own rooms. After they are done filling in the questionnaire, the experimenter presents them the RO and OSO instructions (section A.4 & section A.3) accordingly and does a walk-through of the steps together with each participant to make sure they understand what has to happen.

Then, the OSO is helped to put on the HoloLens in the escape room area. The HoloLens is aligned by making sure the participant can see all edges of the boundary rectangle displayed in the HoloLens. The experimenter explains that the participant should try not to cover the front of the HoloLens with anything, to prevent the sensors loosing track of the environment. If they do loose track during an experiment and the "Trying to map your surroundings" message appears, the participant is asked to inform the experimenter immediately so the HoloLens can be re-calibrated and the experiment either restarted or continued. The OSO is asked to wait in a corner of the escape room and the experimenter turns on a Skype connection with their phone so they can communicate from the other room. Then, the experimenter leaves the room to set up the RO.

Depending on which condition is selected, the RO is asked to either put on the VR headset or sit behind the desk. The software is then started by the experimenter, which starts the connection between the RO and OSO. The experimenter confirms whether the participants can hear each other and asks the RO to look for the OSO within the virtual representation of the escape room. In a condition where the RO has an embodiment at the OSO, the OSO is asked to follow the arrow that is displayed in the HoloLens to find the RO. In order to give the participants an idea of how the gesturing system works, the RO is asked to place his or her hand on top of one of the poles in the middle of the room and the OSO is asked whether he or she can see the hand on op of the pole and to shake hands with each other.

Then, the OSO is instructed to walk to the start position and if the participants don't have any further questions, the experimenter starts the experiment by pressing the right key combination on the keyboard. After the participants have finished, they are helped to take of their HMD's and asked to fill in the post-experiment questionnaire on the computers in their rooms. When the questionnaires are filled in, the participants are set up for the second run with either the baseline or altered condition depending on which one was done in the first run. A tutorial will be done again explaining only the parts that are required for that particular condition that hasn't already been explained during the tutorial of the first run. The experiment is started and after the participants found the code, they will fill in the post-experiment questionnaire for the second time as well. After both participants filled in the questionnaire, the RO is asked to follow the experimenter to the on-site location room, where the participants can open the safe together with the code they found during the experiment. The safe contains candy, which the participants can take as a reward for participating.

The experimenter will answer any remaining questions the participants may still have and (if there is enough time left) offers the participants to try the experiment from the other room as well.

Due to a bug in the point-cloud renderer, the RO system could freeze at random moments in time. The RO software then had to be restarted by the experimenter, after which it would work again. Restarting the software could take up to 30 seconds, during which the RO would see for the most part a black image. This freeze occurred during six different experiment runs. In those cases a decision was made whether to restart the experiment or to continue at the same point in the experiment at which the system froze depending on how far the run had already progressed.

6.9 Data Processing

This section will explain how the data was processed after it was captured during the experiments.

6.9.1 Duration Extraction

During some experiments the system had to be restarted mid-run. This meant that the automatically generated log data files of those runs were split into two. This made the method of measuring the difference between the timestamps of the start and end commands to determine the duration of a run invalid. In order to measure the duration in those cases, camera footage was used as a reference to manually determine the total length of a run.

Depending on how far the participants were into the experiment, the experiment was either resumed at the point where the participants were at the moment the system restarted or the experiment was restarted from the beginning. If the experiment was restarted from the beginning and task 1 was already completed, the time is counted during the first attempt until task 1 is finished, after which the remaining time is counted in the second attempt after task 1 is finished, otherwise only the time of the second attempt is used.

If the experiment was resumed with the participants staying in position, the extra time that was lost to restarting the system is cut out, so that only the true time that was needed to solve the tasks remains.

6.9.2 Learning Effect

To determine if there was a learning effect influencing the performance, a test was done in which the total task durations measured in the baseline condition of first runs are compared to those of second runs across all experiments. The dataset is split up into two groups and one outlier is removed from the second run group (Figure 6.8) which resulted in a sample size of 15 for the first run group and 14 for the second run group.

Shapiro-Wilk normality test for both groups resulted in p-values greater than 0.05, which implies we can assume normality. An F-test resulted in a p-value of 0.021, which implies we cannot assume equal variances. A one-tailed, independent-samples t-test was conducted to compare the total task duration in the baseline configuration during the first run and the second run. There was a significant positive difference in the scores for first runs (M=305.73, SD=91.66) and second runs (M=214.56, SD=47.00); $t(21.187) = 3.403$, $p = 0.0013$. This suggests that the total task durations in the first run are indeed longer than in the second run, which means there was a learning effect that influenced the performance of the participants. The

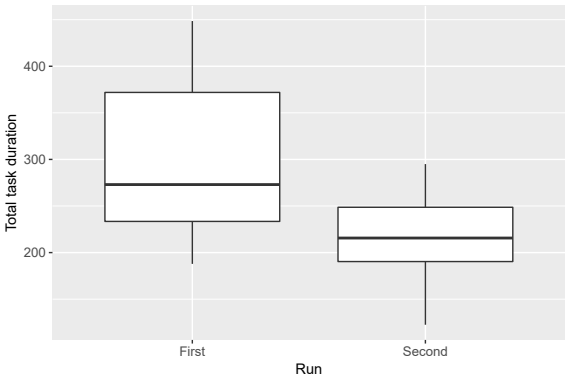


Figure 6.8: Total task durations in the baseline condition, between first and second runs.

average decrease in the required time due to the learning effect is 91,17 seconds, which is determined by taking the difference between the average durations of the first and seconds runs.

To compensate for this learning effect, those 91,17 seconds are added to all durations measured in a second run. This is not skewing the data in favor of a particular condition, because every condition contains the same amount of samples that are collected in a first and in a second run and therefore same amount of time is added to all conditions.

6.9.3 Data Analysis

Because this was an experiment with a within-subject design, we look at the difference between the paired scores from the two runs by the same pair of participants. The scores of the condition with the design aspect disabled (dependent view, non-immersive & no embodiment) are subtracted from the scores of the condition with the same design aspect enabled (independent view, immersive display & virtual embodiment). This creates a positive number when a particular design aspect resulted in a higher score and a negative number when that design aspect resulted in a lower score.

Score differences for all measures were considered outliers when they fell outside 1.5 times the interquartile range above the upper quartile and bellow the lower quartile. This is shown in the accompanying box plot figures in section 6.10 by marking outliers with a point above or below a box plot. Scores that produced an outlier were removed from the dataset and thus not further used during the analysis of that particular measure.

By default, a one-tailed paired-samples t-test is performed on the scores to determine whether there is a statistically significant difference between the scores from the condition with a particular design aspect enabled and the condition where it is disabled.

A paired t-test assumes that the difference in paired scores is normally distributed. Therefore, before conducting a t-test, a Shapiro-Wilk test is conducted on the differences to test this. In case the test shows that the distribution of the differences is significantly different from normal distribution, a paired samples Wilcoxon test is used instead of a paired t-test. In section 6.10 the result of the Shapiro-Wilk test is not shown unless it is significant.

All tests are performed using R version 3.4.3 (2017-11-30) x86_64-w64-mingw32 and the plots are generated using ggplot2 version 2.2.1.

6.10 Results

The results of the statistical tests are presented separately under the corresponding hypothesis.

6.10.1 View independence

✓ **H1.1** ***View independence decreases the total task duration.*** A one-tailed, paired-samples t-test was conducted to compare the total task duration in independent view and dependent view conditions (Figure 6.9). There was a significant negative difference in the scores for independent view (M=301.040, SD=74.400) and dependent view (M=348.644, SD=66.133) conditions; t(9) = -2.614, p = 0.014, mean of the differences = -47.604. This suggests that view independence does indeed decrease the total task duration compared to a fixed view.

✗ **H1.2** ***View independence decreases the amount of errors.*** A one-tailed, paired-samples t-test was conducted to compare the amount of beam hits in independent view and dependent view conditions (Figure 6.10). There was not a significant negative difference in the scores for independent view (M=1.111, SD=1.269) and dependent view (M=0.889, SD=1.269) conditions; t(8) = 0.686, p = 0.744, mean of the differences = 0.222. This shows no evidence to support that view independence decreases the amount of errors compared to a fixed view.

✓ **H1.3** ***View independence increases usability for the RO.*** A one-tailed, paired-samples t-test was conducted to compare the SUS score of the RO in independent view and dependent view conditions (Figure 6.11). There was a significant positive difference in the scores for independent view (M=77.778, SD=10.266) and dependent view (M=66.944, SD=14.185) conditions; t(8) = 3.506, p = 0.004, mean of the differences = 10.833. This suggest that view independence does indeed increase the usability for the RO compared to a fixed view.

✓ **H1.4** ***View independence increases spatial presence of the RO.*** A one-tailed, paired-samples t-test was conducted to compare the perceived spatial presence score of the RO in independent view and dependent view conditions (Figure 6.12). There was a significant positive difference in the scores for independent view (M=6.233, SD=0.610) and dependent view (M=5.150, SD=1.101) conditions; t(9) = 4.079, p = 0.001, mean of the differences = 1.083. This suggests that view independence does indeed increase spatial presence of the RO compared to a fixed view.

✓ **H1.5** ***View independence increases co-presence for the RO.*** A one-tailed, paired-samples t-test was conducted to compare the co-presence score of the RO in independent view and dependent view conditions (Figure 6.14). There was a significant positive difference in the scores for independent view (M=4.407, SD=0.501) and dependent view (M=4.111, SD=0.577) conditions; t(8) = 3.411, p = 0.005, mean of the differences = 0.296. This suggests that view independence does indeed increase co-presence of the RO compared to a fixed view.

✗ **H1.6** ***View independence decreases perceived behavioral interdependence of the RO.*** A one-tailed, paired-samples t-test was conducted to compare the perceived behavioral interdependence score of the RO in independent view and dependent view conditions (Figure 6.15). There was not a significant negative difference in the scores for independent view (M=4.367, SD=0.576) and dependent view (M=4.050, SD=0.604) conditions; t(9) = 1.934, p = 0.957, mean of the differences = 0.317. This shows no evidence to support that view independence decreases perceived behavioral interdependence of the RO compared to a fixed view.

✗ **H1.7** ***View independence decreases perceived behavioral interdependence of the OSO.*** A one-tailed, paired-samples t-test was conducted to compare the perceived behavioral interdependence score of the OSO in independent view and dependent view conditions (Figure 6.15). There was not a significant negative difference in the scores for independent view (M=3.967, SD=0.399) and dependent view (M=4.167, SD=0.423) conditions; t(9) = -1.450, p = 0.090, mean of the differences = -0.200. This shows no evidence to support that view independence decreases perceived behavioral interdependence of the OSO compared to a fixed view.

6.10.2 Immersion

✓ **H2.1** ***Immersion decreases the total task duration.*** A one-tailed, paired-samples t-test was conducted to compare the total task duration in immersive and non-immersive conditions (Figure 6.9). There was a significant negative difference in the scores for immersive (M=298.029, SD=81.956) and non immersive (M=358.348, SD=119.836) conditions; t(8) = -2.105, p = 0.034, mean of the differences = -60.319. This suggests that immersion does indeed decrease the total task duration.

✗ **H2.2** ***Immersion decreases the amount of errors.*** A one-tailed, paired-samples t-test was conducted to compare the amount of beam hits in immersive and non-immersive conditions (Figure 6.10). There was not a significant negative difference in the scores for immersive (M=0.750, SD=1.389) and non-immersive (M=1.375, SD=1.188) conditions; t(7) = -0.886, p = 0.203, mean of the differences = -0.625. This shows no evidence to support that immersion decreases the amount of errors.

✓ **H2.3** ***Immersion increases usability for the RO.*** A one-tailed, paired-samples t-test was conducted to compare the SUS score of the RO in immersive and non-immersive conditions (Figure 6.11). There was a significant positive difference in the scores for immersive (M=77.250, SD=15.831) and non-immersive (M=59.000, SD=17.288) conditions; t(9) = 3.222, p = 0.005, mean of the differences = 18.250. This suggests that immersion does indeed increase usability for the RO.

✓ **H2.4** ***Immersion increases spatial presence for the RO.*** A one-tailed, paired-samples t-test was conducted to compare the perceived spatial presence score of the RO in immersive and non-immersive conditions (Figure 6.12). There was a significant positive difference in the scores for immersive (M=5.633, SD=1.165) and non-immersive (M=4.017, SD=1.738) conditions; t(9) = 2.735, p = 0.012, mean of the differences = 1.617. This suggests that immersion does indeed increase the spatial presence for the RO.

✗ **H2.5** ***Immersion increases co-presence for the RO.*** A one-tailed, paired-samples t-test was conducted to compare the co-presence score of the RO in immersive and non-immersive conditions (Figure 6.14). There was not a significant positive difference in the scores for immersive (M=4.367, SD=0.520) and non-immersive (M=4.150, SD=0.506) conditions; t(9) = 1.361, p = 0.103, mean of the differences = 0.217. This shows no evidence to support that immersion increases co-presence for the RO.

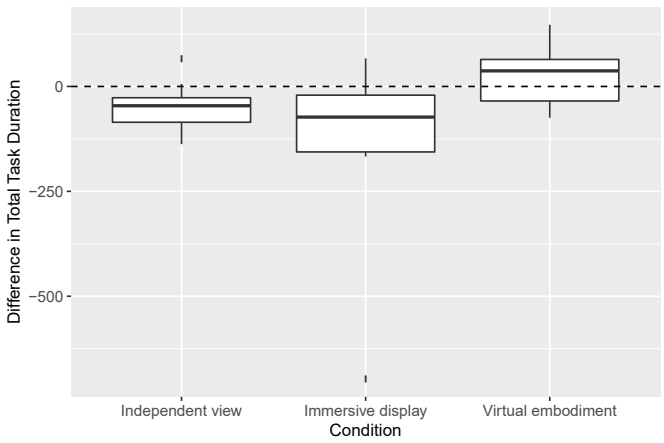


Figure 6.9: Total Task Duration Differences

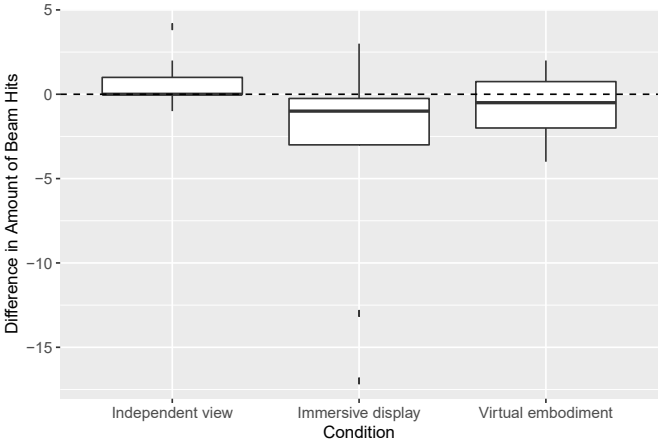


Figure 6.10: Amount of Beam Hits Differences

✓ **H2.6** ***Immersion increases perceived message understanding of the RO.*** A one-tailed, paired-samples t-test was conducted to compare the perceived message understanding score of the RO in immersive and non-immersive conditions (Figure 6.13). There was a significant positive difference in the scores for immersive (M=3.833, SD=0.692) and non-immersive (M=3.278, SD=0.607) conditions; $t(8) = 3.780$, $p = 0.003$, mean of the differences = 0.556. This suggests that immersion does indeed increase perceived message understanding for the RO.

✓ **H2.7** ***Immersion increases perceived message understanding of the OSO.*** A one-tailed, paired-samples t-test was conducted to compare the perceived message understanding score of the OSO in immersive and non-immersive conditions (Figure 6.13). There was a significant positive difference in the scores for immersive (M=4.250, SD=0.517) and non-immersive (M=3.733, SD=0.903) conditions; $t(9) = 1.935$, $p = 0.042$, mean of the differences = 0.517. This suggests that immersion does indeed increase perceived message understanding for the OSO.

6.10.3 Virtual embodiment

✗ **H3.1** ***Giving the visitor a virtual embodiment decreases the total task duration.*** A one-tailed, paired-samples t-test was conducted to compare the total task duration in virtual embodiment and no embodiment conditions (Figure 6.9). There was not a significant negative difference in the scores for virtual embodiment (M=331.437, SD=93.044) and no embodiment (M=301.716, SD=103.732) conditions; $t(9) = 1.229$, $p = 0.875$, mean of the differences = 29.721. This shows no evidence to support that Giving the visitor a virtual embodiment decreases the total task duration.

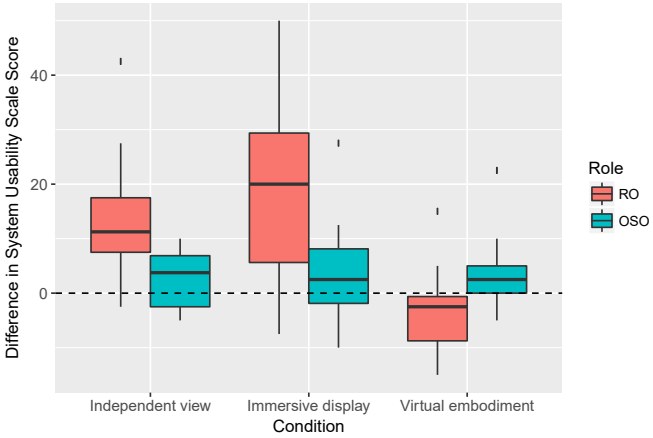


Figure 6.11: System Usability Scale Score Differences

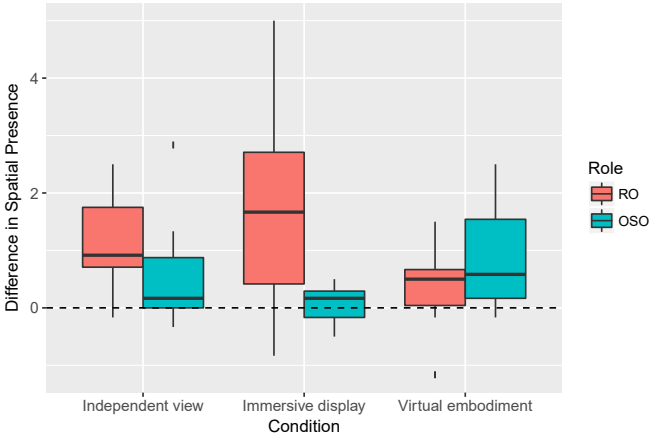


Figure 6.12: Spatial Presence Score Differences

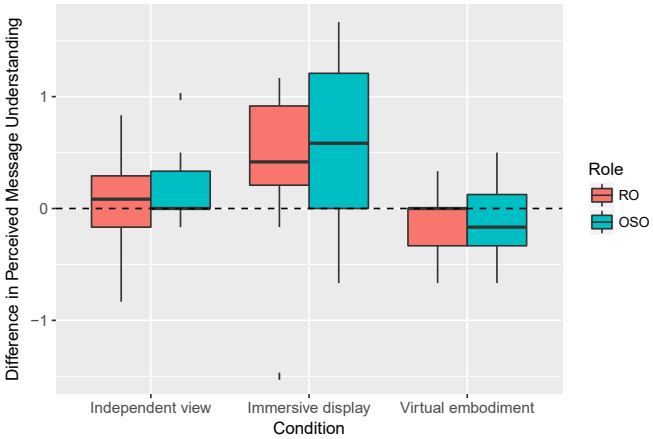


Figure 6.13: Perceived Message Understanding Score Differences

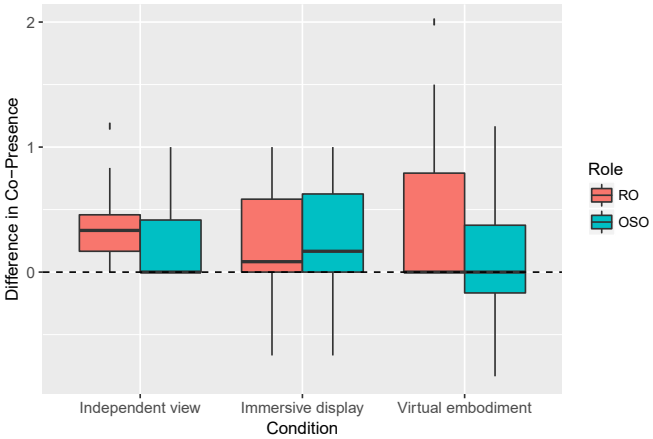


Figure 6.14: Co-Presence Score Differences

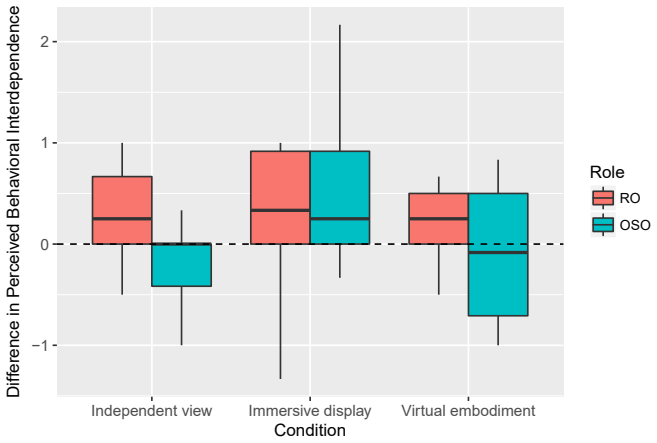


Figure 6.15: Perceived Behavioral Interdependence Score Differences

✗ **H3.2** ***Giving the visitor a virtual embodiment decreases the amount of errors.*** A one-tailed, paired-samples t-test was conducted to compare the amount of beam hits in virtual embodiment and no embodiment conditions (Figure 6.10). There was not a significant negative difference in the scores for virtual embodiment (M=0.600, SD=0.699) and no embodiment (M=1.400, SD=1.776) conditions; t(9) = -1.309, p = 0.111, mean of the differences = -0.800. This shows no evidence to support that Giving the visitor a virtual embodiment decreases the amount of errors.

✓ **H3.3** ***Giving the visitor a virtual embodiment increases usability for the RO.*** A one-tailed, paired-samples t-test was conducted to compare the SUS score of the RO in virtual embodiment and no embodiment conditions (Figure 6.11). There was not a significant positive difference in the scores for virtual embodiment (M=70.500, SD=11.714) and no embodiment (M=73.250, SD=13.126) conditions; t(9) = -1.029, p = 0.835, mean of the differences = -2.750. This suggests that Giving the visitor a virtual embodiment does indeed increase the usability for the RO.

✗ **H3.4** ***Giving the visitor a virtual embodiment increases usability for the OSO.*** A one-tailed, paired-samples t-test was conducted to compare the SUS score of the OSO in virtual embodiment and no embodiment conditions (Figure 6.11). There was not a significant positive difference in the scores for virtual embodiment (M=79.444, SD=8.640) and no embodiment (M=77.500, SD=7.289) conditions; t(8) = 1.306, p = 0.114, mean of the differences = 1.944. This shows no evidence to support that Giving the visitor a virtual embodiment increases the usability for the OSO.

✗ **H3.5** ***Giving the visitor a virtual embodiment increases perceived message understanding of the RO.*** A one-tailed, paired-samples t-test was conducted to compare the perceived message understanding score of the RO in virtual embodiment and no embodiment conditions (Figure 6.13). There was not a significant positive difference in the scores for virtual embodiment (M=4.067, SD=0.459) and no embodiment (M=4.200, SD=0.414) conditions; t(9) = -1.350, p = 0.895, mean of the differences = -0.133. This shows no evidence to support that Giving the visitor a virtual embodiment increases perceived message understanding for the RO.

✗ **H3.6** ***Giving the visitor a virtual embodiment increases perceived message understanding of the OSO.*** A one-tailed, paired-samples t-test was conducted to compare the perceived message understanding score of the OSO in virtual embodiment and no embodiment conditions (Figure 6.13). There was not a significant positive difference in the scores for virtual embodiment (M=4.017, SD=0.319) and no embodiment (M=4.133, SD=0.205) conditions; t(9) = -1.000, p = 0.828, mean of the differences = -0.117. This shows no evidence to support that Giving the visitor a virtual embodiment increases perceived message understanding for the OSO.

✓ **H3.7** ***Giving the visitor a virtual embodiment increases co-presence of the RO.*** A Shapiro-Wilk normality test for the differences between the co-presence score of the RO in virtual embodiment and no embodiment conditions was conducted which resulted in a p-value of p=0.00067. This implies that the distribution of the differences are significantly different from normal distribution. Therefore, normality can not be assumed and a paired-samples t-test cannot be used.

A paired samples Wilcoxon test was conducted to compare the co-presence score of the RO in virtual embodiment and no embodiment conditions (Figure 6.14). There was a significant positive difference in the scores for virtual embodiment (M=4.350, SD=0.412) and no embodiment (M=3.883, SD=0.798) conditions; V = 28, p = 0.022, mean of the differences = 0.467. This suggests that Giving the visitor a virtual embodiment does indeed increase co-presence for the RO.

✗ **H3.8** ***Giving the visitor a virtual embodiment increases co-presence of the OSO.*** A one-tailed, paired-samples t-test was conducted to compare the co-presence score of the OSO in virtual embodiment and no embodiment conditions (Figure 6.14). There was not a significant positive difference in the scores for virtual embodiment (M=4.167, SD=0.624) and no embodiment (M=4.050, SD=0.478) conditions; t(9) = 0.606, p = 0.280, mean of the differences = 0.117. This shows no evidence to support that Giving the visitor a virtual embodiment increases co-presence for the OSO.

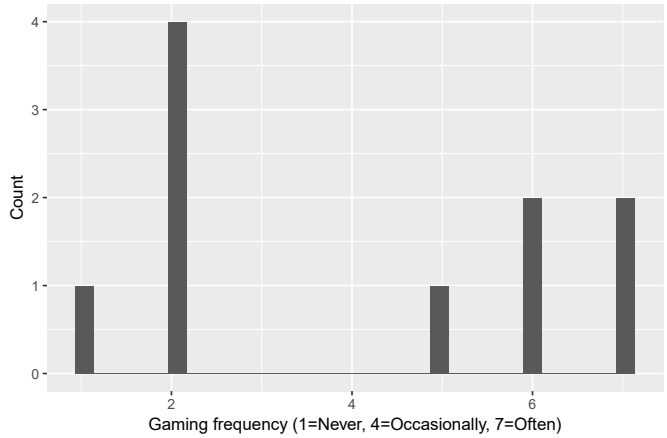


Figure 6.16: Gaming frequency of ROs in the non-immersive condition.

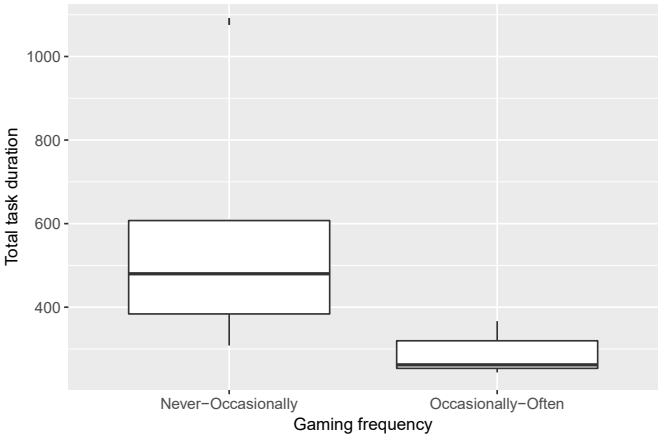


Figure 6.17: Total task duration in the non-immersive condition plotted against the RO's gaming frequency

6.11 Exploratory study

Besides using the captured data to test the hypotheses from section 6.3, we used the data as well to answer questions that arose during and after the experiment.

Gaming Frequency We observed a distinction between participants who were having difficulties navigating in the non-immersive configuration using the 2D desktop interface and participants who didn't seem to have difficulties with it. For participants who had difficulties navigating, the total time needed to complete the tasks seemed to be longer as well. Navigating in the non-immersive condition is done by using keyboard and mouse controls that are often found in popular video games as well. Therefore we hypothesize that participants who have more experience playing video games will have less difficulties navigating the virtual representation of the on-site environment in the non-immersive condition which results in a lower total task duration.

We used question 14 of the ITQ (*How often do you play arcade or video games?*) to make a distinction between participants who never to occasionally play video games and participants who play occasionally to often. Figure 6.16 shows the distribution of the answers given on this question by the RO's who used the system in the non-immersive condition. Those RO's have been split up into two equal populations; 5 who play video games never to occasionally and 5 who play occasionally to often.

We can look at Figure 6.17 to compare the total task duration of RO's who never to occasionally play video games (M=433, SD=130) and those

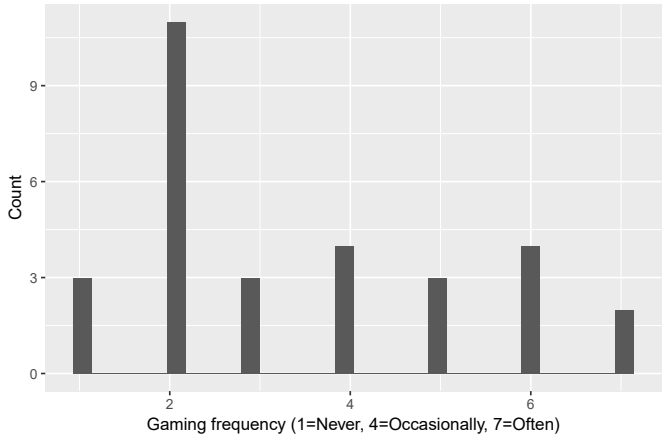


Figure 6.18: Gaming frequency of ROs in the baseline condition.

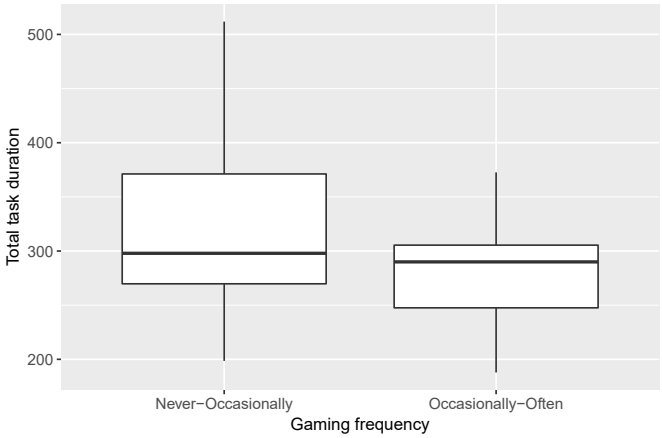


Figure 6.19: Total task duration in the baseline condition plotted against the RO's gaming frequency.

who play occasionally to often (M=283, SD=54) in the non-immersive configuration. We see a clear increase in the mean time for RO's who play never to occasionally. This suggests that RO's who game more often were indeed faster in completing the tasks in the non-immersive configuration.

Based on the effect that gaming experience seems to have on the performance in the non-immersive condition, we looked at whether gaming experience also affects the performance in other conditions. This can give an insight in whether the keyboard and mouse controls are the cause of the better performance for RO's that game more often or whether gaming frequency is an overall indicator for better performance in remote collaborative work scenarios. We use the baseline condition for this comparison, as all participants have completed this condition which allows us to work with more samples compared to using the no embodiment or dependent view conditions.

As can be seen in Figure 6.18, four participants selected the middle option, those will be disregarded. This results in a group of 17 RO's who play never to occasionally and a group of 9 RO's who play occasionally to often.

We can look at Figure 6.18 to compare the total task duration of RO's who never to occasionally play video games (M=330, SD=92) and those who play occasionally to often (M=277, SD=54) in the baseline configuration. We see that the means of the two groups are closer together than they were with the non-immersive configuration but still show a difference. This suggests that "gamers" may be more motivated to perform in collaborative tasks compared to "non-gamers" but that this advantage becomes especially

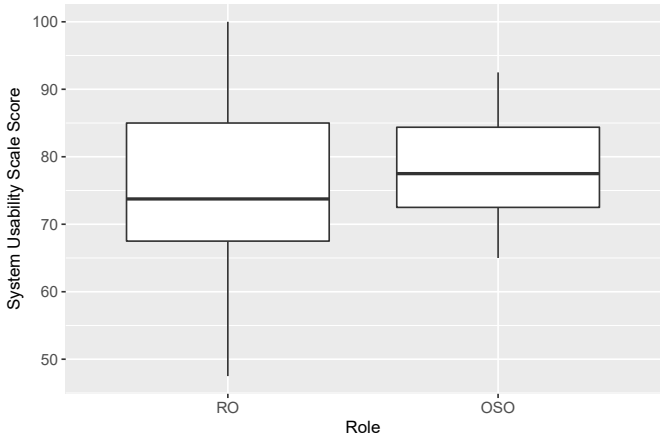


Figure 6.20: System Usability Scale Scores between the RO and OSO in the baseline condition.

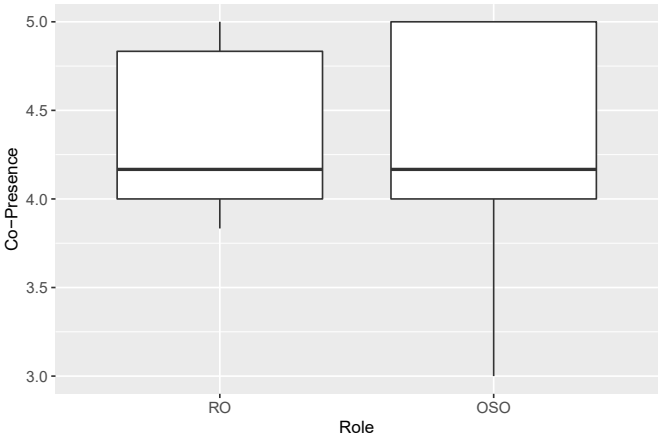


Figure 6.21: Co-presence between the RO and OSO in the baseline condition.

noticeable when they use a desktop interface that is similar to what they use when playing video-games.

OSO vs RO The OpenIMPRESS system is asymmetrical, this means that the part of the system that the RO uses is different from the part the OSO uses. Where the RO is presented a completely virtual representation of the on-site environment, the OSO is physically there and the system only overlays a virtual embodiment of the RO on top of the OSO's view of the real environment. This means that there is a difference in the experience that the system provides to both users.

Because the users have to interact with the system in different ways, we may expect this to lead to a difference in usability. Also, because there is a difference in how both users perceive each other (the RO can see a visor and a complete point cloud of the OSO's body where the OSO can only see the RO's hands and visor), we expect a difference in co-presence. We expect that the asymmetrical nature of the system results in a difference in usability and co-presence between the RO and OSO. For this comparison we use the SUS and co-presence scores from ROs and OSOs in the baseline condition.

First, we will look at the SUS scores between the RO (M=75.42, SD=12.80) and OSO (M=78.08, SD=7.924) using Figure 6.20. We see that the means are relatively close to each other but that the scores from the RO are more spread out than the scores from the OSO are. This suggests that on average there is not a difference in usability between the RO and OSO, but that from person to person there is more variance in how usable the system is perceived by the RO.

Next, we will look at the co-presence scores between the RO (M=4.40, SD=0.468) and OSO (M=4.31, SD=0.603) using Figure 6.21. Again we see that the means are relatively close to each other but that the scores from this time the OSO are a little bit more spread out. This suggests that on average the users feel each other's presence with the same strength, but that there is more variance in the OSO's feeling of co-presence.

6.12 Discussion

Three design aspects of our mixed reality telepresence system have been tested to understand how they affect the performance and experience of the users in a collaborative context. Each aspect has been tested in a study in which the participants had to collaborate once in the system configured with all aspects implemented and once configured with the aspect not implemented. Because the configurations had the other two aspects implemented in both cases, this resulted in scenarios that give new insights into how those aspects influence performance and experience in a telepresence system compared to previous research.

View independence Allowing the RO to freely walk around in the virtual representation of the on-site environment has a positive influence on the overall performance, usability for the RO and the feeling of spatial and co-presence of the RO. This is in line with the results found by Tait and Billinghurst (2015), which showed the positive effects of view independence on the quality of the collaboration using a desktop interface. This research builds on top of that by showing that with an immersive display and a virtual embodiment of the RO, positive effects are measurable as well.

We didn't find evidence for view independence decreasing the perceived behavioral interdependence for either user as was hypothesized in H1.6 and H1.7. The idea was that by disconnecting the RO's view from the OSO's view, their behaviors would become more independent as the RO can now independently navigate through the virtual representation of the on-site environment. When looking at Figure 6.15 we have a strong indication that view independence actually increases the perceived behavioral interdependence for the RO instead of decreasing it. This could be a result of the increased efficiency of the system which allows the RO to respond to the OSO's actions more directly.

We assumed that view independence makes the system become more usable because of the RO being able to look at what he/she needs without the need to ask the OSO. After observing the participants, this didn't always seem to be the case. With the block collection task, the RO often asked the OSO to hold up a block so it would be better visible, even when the RO could freely look at objects from different perspectives.

This is probably a side effect of the limited resolution of the depth cameras and the fact that they only scan objects from one side, thus giving no information about the back of an object without it physically being rotated in front of the camera. This effect has been minimized by placing multiple depth cameras at the on-site location so objects would be scanned from multiple sides, but because of alignment errors it could sometimes still be confusing as to how an object actually looked like. Higher resolution cameras and filter algorithms that correct for alignment errors and gaps in the point cloud data could be a good solution for this issue. Also, converting the point cloud data into a mesh could help, as it gives more certainty about contours of an object compared to using a direct point-cloud representation.

Immersive display Visualizing the virtual representation of the on-site location to the RO using an immersive VR display did decrease the total task duration, increase the usability and perceived spatial presence for the RO and increase the perceived message understanding for both users.

As expected, the feeling of presence in the virtual environment was indeed higher compared to using a desktop set-up, which is one of the main selling points for using VR in the first place. The system is rated more usable by the RO but not by the OSO (Figure 6.11). This can be explained by the fact that the system appeared the same to the OSO in both conditions; whether the RO is using a VR display or a desktop monitor, in both conditions the RO has a virtual embodiment that can freely navigate the environment and looks the same. The increased usability that was perceived by the RO apparently did not translate into a higher usability for the OSO as well.

By observing the participants we could see that a lot of RO's had trouble navigating through the virtual representation of the environment in the non-immersive condition using the 2D desktop interface. Those participants often tried to give instructions to the OSO from a perspective in which it was difficult to see important parts of the task, by parts for example being obscured by other parts or just being far away. We observed for example some RO's trying to give instructions about how to navigate through the maze while looking at the maze from the end of the maze while the participant is still standing at the beginning of the maze. Because the navigation in the non-immersive condition used controls that are relatively common in video games, we expect this behavior to be correlated with the amount of experience the RO has with playing video games. In the exploratory study we indeed found that RO's who reported to play video-games regularly needed a lower time on average to complete the tasks.

Observations of the RO's in this condition also seem to suggest that moving around a lot improves performance. RO's that completed the tasks relatively fast often seem to move around a lot, especially in the maze task.

We hypothesize this to be a technique that is used to compensate for the lack of stereoscopic depth perception when using a 2D monitor; by moving around a part of the scene the motion parallax can create a impression of the relative depth when other cues are absent (Rogers and Graham, 1979). The lack of stereoscopic depth perception can also be a reason for the hand gesture system not being used relatively often. It looked like people had trouble understanding how far away their hands exactly were from their position in the virtual representation.

The higher usability score from the RO seems to be easily explained when looking at the reaction of some of the participants that switched from the desktop condition to the VR condition, announcing how much better and easier the system was to use in VR.

Virtual embodiment Giving the RO a virtual embodiment did increase the usability and co-presence for the RO. This seems to suggest that, at least for the RO, having a virtual body in the virtual representation of the on-site environment and in the on-site environment itself has a positive effect on the experience.

However, we didn't find an increase in any of the performance measures or in both users' perceived message understanding. This is hypothesized to be because of the users still needing to learn how to properly make use of the advantages of having an embodiment. By observing the participants we saw that the RO often navigated the OSO through the maze by only using their voice while the ability to use hand gestures was available. Some participants didn't seem to realize they could use them only until the last task, while others tried to use them for navigation but didn't persist and fell back to verbal commands after it didn't work for them as expected. A reason could be the limited field of view of the Hololens and the OSO not being able to see the hands, especially when they are held close the the OSO's head. The RO would sometimes ask whether the OSO could see his or her hands and just fall back to voice if there was any confusion.

Another reason for not finding an increase in both users' perceived message understanding could be the fact that voice communication was available in all configurations. This could have made the participants feel like they were able to understand each other well enough independently from whether there was a virtual embodiment of the RO or not. This could also have lead to a ceiling effect; that the voice communication caused the measurement scales to be maxed out, prohibiting to measure a further increase that a virtual embodiment would cause. If we look at Figure 6.22 we see that the scores in the configuration where only voice communication was possible are indeed close to 5, which is the maximum value that the scale allows.

No increase in co-presence was found for the OSO. We expected that the visualization of the RO would make the OSO more aware of the RO's

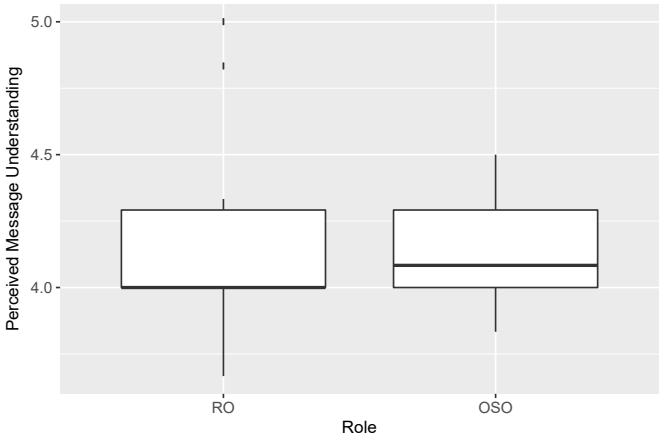


Figure 6.22: Perceived Message Understanding between the RO and OSO in the no embodiment configuration.

presence compared to not being able to see the RO at all and therefore increase the feeling of co-presence. The limited field of view of the HoloLens and the reduced visibility by the absence of a torso in the RO's embodiment can be reasons for the OSO not noticing the RO enough to become significantly more aware of him/her. Interestingly, the co-presence of the RO *did* increase when giving him/her a virtual embodiment. This could be a sign that, although the OSO didn't feel any difference, the OSO behaved differently towards the RO when a virtual embodiment was available. The OSO could for example have directed their communication more towards the RO, which made the RO therefore feel more addressed and more co-present. Another explanation could be that because the RO could now see his/her own hands, the RO just assumed that the OSO would feel more co-present with him/her. An additional user test could be performed in which the the virtual embodiment only gets disabled for the OSO. In this case, the RO would be able to see their hands in both conditions, limiting the cause of a potential increase in co-presence to only the OSO's behavior towards the RO.

Different types of techniques were used to navigate through the maze: navigating the OSO using voice commands like "up, left, forward", making the OSO understand where the beams are by pointing out their locations using hands, often in combination with afterwards holding a hand where the OSO should go, the RO going through the maze him or herself and telling the OSO to follow while not looking at the OSO or the RO moving his/her hand through the maze while walking backwards and looking at the OSO. This last method seemed the most effective, the fastest time recorded used this method. This shows that having an embodiment could indeed be more effective, but only in combination with a proper technique. Because in this

experiment it was left to the participants to come up with a way to use the available tools, they may not have been used in the most effective way possible. Although this gives a better impression of how intuitive the tools are to use, in most use cases users would first have a bit more training and an explanation of how the tools can be used in the most optimal way for their scenario. Doing this before running the experiment could have arguably lead to more fair results.

If we look at Figure 6.12, we see that giving the RO a virtual embodiment did seem to have a positive effect on the feeling of spatial presence of the RO. It is hypothesized that this is because of the RO's hands now being visible in the virtual scene which give the RO a visual cue that helps to feel more present in the virtual representation. By looking at Figure 6.12 we also see that giving the RO a virtual embodiment seems to increase the feeling of spatial presence for the OSO. We try to explain this by looking at the questions of the IPQ (Schubert, Friedmann, and Regenbrecht, 2001). The questions about spatial presence focused on the feeling of being present in the virtual environment. Because the OSO's visualization of the virtual environment is limited to the embodiment of the RO, it is logical that the OSO scores higher on spatial presence when there is an embodiment at the OSO compared to when there isn't an embodiment visualized.

In this evaluation we looked at how the design aspects of our telepresence system, which are described in section 1.1, influence the performance and experience of the users in a collaborative setting. This was done by conducting a user study in which 60 people participated. The user study was split up into three sub-studies that each evaluated the effects of either view independence, an immersive display or a virtual embodiment. Pairs of two participants collaborated remotely on a series of tasks in an escape room setting, once with all of the three design aspects enabled and once with one of the aspects disabled. By comparing the performance and scores of questionnaires about usability, spatial presence and social presence between the two runs, the effects of the design aspects were determined.

In the next chapter we will use the results of the user study to answer each of the research questions separately.

Chapter 7

Conclusion

We presented OpenIMPRESS, an open source immersive telepresence system for remote collaboration applications. The system has been designed with a focus on giving the users an immersive experience and making it accessible. The first research question was as follows: *How to create a state of the art end-to-end mixed reality telepresence system that is immersive and accessible?* It has been split up into four subquestions, which are answered individually below.

The main evaluation focused on answering the second research question: *How do the three telepresence system design aspects influence the performance and experience of the users in a collaborative setting?* This question was split up into three sub questions; one for each design aspect, which are answered individually below as well.

7.1 Research Questions

RQ1.1. How to create an immersive telepresence experience? To ensure an immersive experienced we based the design on three aspects that have been understood to be important in telepresence systems; mobility at the remote location, an immersive view on the environment and the ability to affect the remote environment. The mobility aspect is covered by making use of free viewpoint video techniques to scan and visualize the on-site environment to give the visitor complete freedom to navigate trough a virtual representation of this environment. By using a VR headset, the visitor has an immersive view on the on-site environment. A basic model of the visitor including detailed scans of the hands is made and displayed at the visitee, which gives the visitor the ability to have an effect on the people in the remote environment.

RQ1.2. What components can be used, while keeping it accessible? To make the system accessible, only commodity hardware compo-

nents are used in OpenIMPRESS. The visitor part uses a generally available VR system that is currently becoming popular among the gaming community. This is extended with a consumer grade Leap Motion hand sensor to provide hand tracking, allowing the visitor to use detailed hand gestures in the shared space.

The visitee part is using a HoloLens, which at the point of writing is available as a developer edition, but should become available for consumers soon as well. To scan the on-site environment we use commodity depth cameras like the Microsoft Kinect and the Intel RealSense.

RQ1.3. How to cover a large working area? To cover a large working area we make use of the HoloLens' automatically generated spatial model as a basis of the virtual representation of the remote environment. This ensures that wherever the visitee goes, there will be a representation available for the visitor to navigate in. By making use of the Hololens' tracking system, we also allow on the fly recalibration of the depth cameras so the working area can be reconfigured as necessary.

RQ1.4. How to make the system easy to set-up and use? There are multiple aspects that make thee system easy to set-up. First of all, the networking system allows connecting the different components with each other with relative ease, by using a web-interface on which the user can create sessions and assign components to those sessions. It also circumvents the need of manually managing port forwarding settings of routers through which components connect to the internet, which greatly decreases the threshold for setting this system up in different environments. Secondly, because of the focus on usability of the HoloLens, the steps that are required to get it to work are similar to what people are used to with current mobile devices like smartphones.

The system has been made easy to use by relying on natural interactions as much as possible. The visitor can navigate by physically walking around and use hand gestures as one would do in the real world as well. At the visitee, a virtual representation of the visitor is visualized as a three dimensional hologram that integrates with the visitee's environment as one would expect, which promotes natural interactions from the visitee.

RQ2.1. ***How does view independence influence the performance and experience of the users in a collaborative setting?*** View independence has a positive influence on the performance. On average, participants finished the complete experiment 14% (48 seconds) faster when the visitor was free to navigate independently trough the virtual representation of the on-site environment.

It also has a positive influence on the experience the visitor has with the system. It was rated more usable on the system usability scale and it resulted in a higher feeling of spatial- and co-presence for the visitor. We didn't see a decrease of the perceived behavioral interdependence when switching from the dependent view condition to the independent view condition; users don't feel their behaviors become more independent when the visitor's view isn't locked to the visitee's.

RQ2.2. ***How does using an immersive display influence the performance and experience of the users in a collaborative setting?*** Immersive displays seem to have a positive influence on the performance. The time participants needed to complete the experiment dropped by 16.8% (59 seconds) on average when the visitor used an immersive VR display compared to when the visitor used a 2D desktop display. As shown in section 6.11, people that game regularly did seem to have less problems using the non-immersive display than people who don't.

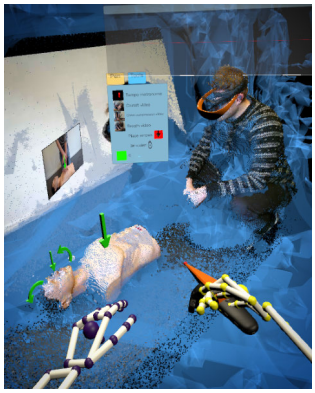
Using an immersive display also has a positive effect on the experience. By using an immersive display the system received a higher usability score from the visitor and gave the visitor a higher feeling of spatial presence. Co-presence did not increase; the visitor doesn't become more aware of the visitee when using an immersive display. Message understanding does increase for the visitor but not for the visitee, which shows that immersion helps with making communication easier for the person that is immersed.

RQ2.3. ***How does giving the visitor a virtual embodiment influence the performance and experience of the users in a collaborative setting?*** Giving the visitor a virtual embodiment does not seem to have an effect on the performance but does have a positive effect on the visitor's experience. The visitor rated the system more usable when he/she had a virtual embodiment, but there was no difference in the visitee's usability rating when a virtual representation of the visitor was available.

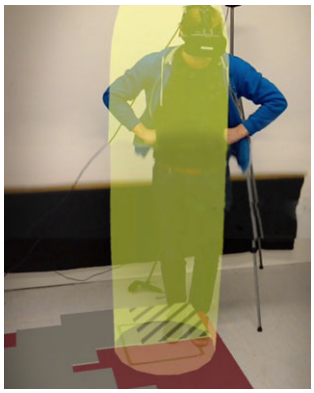
Both users don't perceive a higher sense of message understanding when a virtual embodiment of the visitor is displayed, this suggests that gestures and body language don't outweigh the other communication channels available to the participants

The co-presence of the visitor does increase when he/she is given a virtual embodiment, but the visitee's co-presence doesn't.

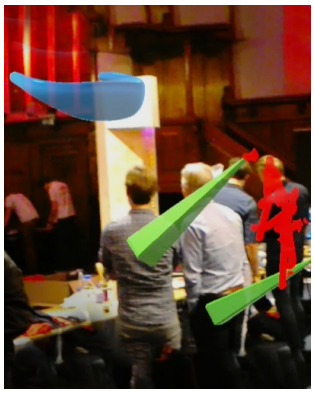
Overall we see that view independence and using an immersive display both contribute to a higher performance in the form of a decrease in the total task duration and make the visitor feel more spatially present in the virtual representation of the visitee's environment.



(a) The teleoperator's view when assisting the bystander in performing CPR.



(b) The OSO's view while navigating the RO through a maze in a trust building study.



(c) OpenIMPRESS being used at the Twente Science Night.

Figure 7.1: Cases of OpenIMPRESS being used in other settings.

View independence, using an immersive display and giving the visitor a virtual embodiment all make the system more usable for the visitor. View independence and giving the visitor an embodiment both make the visitor feel more co-present with the visitee. Equipping the visitor with an immersive VR display also helps the visitor with communication by increasing the perceived message understanding.

All three design aspects either have a positive influence on the performance, the experience or both.

7.2 Additional Findings

Besides the evaluations done in chapter 5 and chapter 6, OpenIMPRESS has also been used in contexts other than remote collaboration and more informal settings. During those sessions some additional findings were made which are discussed in this section.

Other Uses OpenIMPRESS has already been used in a variety of different scenarios; from other research to public demos.

The system has been adapted to serve in a study into how mixed reality telepresence can assist in emergency first response scenarios where CPR has to be performed. In this case, a expert teleoperator gives instructions to a bystander that has to perform CPR on a patient. The bystander is in this case the OSO wearing a HoloLens, and the teleoperator is the RO who is virtually present using VR (Figure 7.1a). OpenIMPRESS has been adapted to include useful tools that the teleoperator can use to guide the bystander

through the necessary steps to perform successful CPR, like instructional videos, arrows and rhythm bars.

The system has also been adapted and used for a study into building trust between people. In this case a physically co-located configurations was used, where the OSO had to guide the RO to follow a path that was only partially visible for the RO (Figure 7.1b).

Next to being used in other research, OpenIMPRESS has also been demoed at a public science event (Figure 7.1c) and at an AR/VR workshop[1].

Mobility A main feature of our system is the mobility that the RO has in the remote environment's representation. When leaving the area that is covered by the depth cameras, the RO loses a detailed view of the on-site location. The only visual cues that are left are the HoloLens' spatial model and a minimal representation of the OSO based on the location of the HoloLens. We expected that in those conditions the system would not provide any advantages compared to rudimentary telecommunication techniques like phone calls. We found out however, that it still provides an unique experience to both the RO and OSO that can be used in various scenarios, with the advantage of not having to set up anything; the OSO only needs to wear a HoloLens with an internet connection. This setup allows the OSO to walk anywhere while still having the RO virtually present. Because the HoloLens is continuously updating the spatial model while the OSO is moving, the virtual representation of the on-site environment at the RO is updated where needed as well. This allows the RO to follow the OSO wherever they go. The RO can follow by physically walking or, when the RO hits the limits of their own environment, by using the "dragging" method described in subsection 4.5.2.

The aspect of mobility is similar to what is done with *Jack-in Head* (Kasahara and Rekimoto, 2015). In this system the visitee is wearing a 360° camera that allows the visitor to look around wherever they go. The difference with OpenIMPRESS is that in OpenIMPRESS the visitor can freely navigate the on-site location and isn't locked to the OSO's perspective. Because the RO is also visualized as its own entity in the view of the OSO using the holographic display capabilities of the HoloLens, the OSO is also inclined to treat the RO as such. This completely changes the dynamics of the user's relationship while using the system compared to a phone or video call, because among others, things like personal space and gaze suddenly start playing a role. It is a refreshing realization when using this system that a remote partner can just follow you through a door, walk next to you in a hallway and enter a completely different room while you can talk to them and see their hand gestures as if they were actually right there with you, without any interruptions or the need to set anything up in the spaces beforehand.

[1] `https://www.4tu.nl/ht/en/events/workshop_virtual_augmented_reality/`

The combination of the spatial model and the annotation system allows the RO to create drawings that take the OSO's space in account.

Games Various games emerged while people were trying out the system. For example a game of tag, where the RO has to try to find and capture the OSO. As we learned from *The Matrix* (The Wachowskis, 1999), the laws of physics don't have to apply in virtual reality. By using the "dragging" gestures, the RO can fly above the surface as if there is no gravity. This creates new and interesting dynamic to the classic game of tag, that allows for scenarios that would otherwise be impossible.

Another example made use of the annotation capabilities to create virtual three dimensional artwork in at the on-site location. This is similar to how people use Tilt Brush (Google, 2016a), with the difference of now having a real environment as the canvas. Some people used this functionality to play a game where the RO has to draw something and the OSO has to recognize what is being drawn as fast as possible. The fact that the drawings can be made in 3D literally adds a new dimension to this game, which is normally played on a piece of paper or on a (touch)screen.

Detail The experience for the RO is relatively limited in such a completely mobile configuration because of the lack of detailed scans from dedicated depth cameras. Ideally, we would use high resolution real-time point clouds captured with depth cameras that are built into the HoloLens. This would allow for a detailed representation with a fast refresh rate of the area the OSO is currently facing. Technically, the HoloLens actually has depth cameras built in already, although the raw data is not accessible and only used to create and maintain the spatial model in the background. Creative solutions could be applied to achieve the same effect. On of those solutions is demonstrated with the setup build by Garon et al. (2016), which mounted an additional depth camera on top of the HoloLens including a stick-PC and battery pack to process and stream the captured data while still retaining a mobile system.

7.3 Future Work

Now that the system has been implemented, tested and found to work, we identified some areas for improvement.

Platform Independence At the moment the functionality of the system is fixed depending on the role of the user. The RO uses VR to view point clouds of the OSO and it's environment and the OSO uses a HoloLens to see a rudimentary visualization of the RO. A next step could be to make the system more platform independent to allow point clouds to be displayed in the HoloLens as well, for which the work of Kowalski, Naruniec, and Daniluk (2015)[2] can be used as a basis.

We see different use cases for displaying point clouds in the HoloLens. First of all, it can be used to increase the level of detail of the RO's virtual representation. By pointing one or more depth cameras to the RO in the RO's physical environment a full body 3D scan of the RO can be made. By using the body index frames that are provided by the Microsoft Kinect SDK[3], we can easily filter out background objects so that only the RO's body is displayed.

Another use case for point clouds in the HoloLens is to display previews to the OSO of what the depth camera's are capturing in the OSO's environment before it gets streamed to the RO. We noticed that there isn't a user friendly way for the OSO to determine what the cameras are seeing, which can be challenging during set-up.

Making the system more platform independent also encompasses allowing the RO use AR glasses instead of VR glasses. This blurs the line between RO and OSO, resulting in a more symmetrical system similar to Holoportation (Orts-Escolano et al., 2016). In a temporally dislocated but physically co-located scenario (section 3.2), where previously recorded material is played back at the same location to be used for training purposes for example, using AR glasses like the HoloLens could make more sense than using VR glasses.

Modularity By making the system more modular, the system could support a wide range of new use cases. With modularity we mean, making it easy to add or remove components like extra HoloLenses or extra viewing locations. Adding extra HoloLenses would for example allow multiple people at the same location to interact with the same remote visitor. This can also work the other way around; by allowing multiple RO's to join a session, people from multiple remote locations can be virtually present at the on-site location.

[2] `https://github.com/MarekKowalski/LiveScan3D-Hololens`

[3] `https://msdn.microsoft.com/en-us/library/windowspreview.kinect.bodyindexframe.aspx`

The main challenge when making the system more modular is updating the *Networking System*. Right now, data is always sent from one component to one other component. When for example adding multiple RO's, data from the on-site depth cameras and HoloLens(es) need to be sent to multiple locations. Depending on the implementation, this would require a fundamental change in the design of the *Matchmaking Server* or in both the *Matchmaking Server* and the *UDP Connector*.

One option would be to change the role of the *Matchmaking Server* from solely keeping track and matching IP addresses and port numbers to a central data hub. In this case, all data that needs to be shared will first be sent to the server, where it will be forwarded to one or multiple destinations. The advantage here is that, the *UDP Connector* always has to sent the data to only the server, even when it has to be sent to multiple other components, which reduces the bandwidth requirements at the clients.

Another option would be to allow the individual *UDP Connector* components to connect to multiple other *UDP Connectors*. In this case the *Matchmaking Server* keeps its current role, but needs to get extended to support keeping track of multiple connections per component. This makes the system slightly more complex but is more scalable, as the entire system doesn't get limited by the bandwidth at the *Matchmaking Server*.

Robotics Right now only virtual elements that the RO can manipulate are added to the OSO's environment through the HoloLens. This allows the RO to have an effect on the OSO's environment but doesn't allow the RO to directly physically manipulate objects. In many use cases, the OSO can be seen as an extension of the RO; physically manipulating the environment based on the RO's instructions.

Robotics could be a solution to allow the RO to directly physically manipulate the on-site environment, without the need for the OSO to act as a middle-man. Robotic arms can be programmed to follow the RO's hand movements, which can already be successfully captured with the current system. Because of safety and speed limitations that have to be taking into account when using robotics, we envision the RO's robotic physical embodiment to be used in conjunction with the virtual embodiment. For example, adding specific robotic zones in the on-site environment, in which the robotic arms lock to and start tracking the RO's hands when they are moved into that zone. This gives the RO the freedom to navigate the on-site environment as is currently possible with the system but also allows the RO to physically manipulate the on-site environment at specific places where it is deemed necessary.

7.4 Final Thoughts

The designed system is implemented, tested and has been found to work. The system's accessibility has been proven by other researchers who successfully adapted and used the system for their own research. We have also shown that the implemented design features do indeed improve the experience and usability of the system, which are useful insights for continuing research in the area of telepresence.

We hope to see that this work is used to spark a trend into making telepresence systems more accessible and available to the general public. Like advances in the current communication systems have made it a commodity to be able to communicate with everybody though text and voice, the same can become true for telepresence. A telepresence system that is both immersive and accessible could easily disrupt the way we travel, work and communicate with people that are far away from us.

We are convinced that our contribution will help to further the research and development of telepresence systems and are eager to see what other advances lie ahead.

Bibliography

Amores, Judith, Xavier Benavides, and Pattie Maes (2015). "Showme: A remote collaboration system that supports immersive gestural communication". In: *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, pp. 1343–1348.

Baños, Rosa María, Cristina Botella, Mariano Alcañiz, Víctor Liaño, Belén Guerrero, and Beatriz Rey (2004). "Immersion and emotion: their impact on the sense of presence". In: *CyberPsychology & Behavior* 7.6, pp. 734–741.

Benavides, Xavier, Judith Amores, and Pattie Maes (2015). "Remot-IO: a System for Reaching into the Environment of a Remote Collaborator". In: *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, pp. 99–100.

Bourges-Sévenier, Mikaël and Euee S Jang (2004). "An introduction to the MPEG-4 animation framework extension". In: *IEEE transactions on Circuits and Systems for Video Technology* 14.7, pp. 928–936.

Brooke, John et al. (1996). "SUS-A quick and dirty usability scale". In: *Usability evaluation in industry* 189.194, pp. 4–7.

Butler, D Alex, Shahram Izadi, Otmar Hilliges, David Molyneaux, Steve Hodges, and David Kim (2012). "Shake'n'sense: reducing interference for overlapping structured light depth cameras". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pp. 1933–1936.

Carranza, Joel, Christian Theobalt, Marcus A Magnor, and Hans-Peter Seidel (2003). "Free-viewpoint video of human actors". In: *ACM transactions on graphics (TOG)*. Vol. 22. 3. ACM, pp. 569–577.

Cazamias, Jordan and Abhilash Sunder Raj (2016). "Virtualized Reality Using Depth Camera Point Clouds".

Collet, Alvaro, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan (2015). "High-quality streamable free-viewpoint video". In: *ACM Transactions on Graphics (TOG)* 34.4, p. 69.

Doc-Ok (2014). *3D Video Capture with Three Kinects*. URL: `http://doc-ok.org/?p=965` (visited on 05/26/2017).

Epic Games (1998). *Unreal Engine®*. URL: `https://www.unrealengine.com`.

Feldmann, Ingo, Wolfgang Waizenegger, Nicole Atzpadin, and Oliver Schreer (2010). "Real-time depth estimation for immersive 3D videoconferencing". In: *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*. IEEE, pp. 1–4.

Ford, Bryan, Pyda Srisuresh, and Dan Kegel (2005). "Peer-to-Peer Communication Across Network Address Translators." In: *USENIX Annual Technical Conference, General Track*, pp. 179–192.

Fussell, Susan R, Leslie D Setlock, and Robert E Kraut (2003). "Effects of head-mounted and scene-oriented video systems on remote collaboration on physical tasks". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pp. 513–520.

Fussell, Susan R, Leslie D Setlock, Jie Yang, Jiazhi Ou, Elizabeth Mauer, and Adam DI Kramer (2004). "Gestures over video streams to support remote collaboration on physical tasks". In: *Human-Computer Interaction* 19.3, pp. 273–309.

Gao, Lei, Huidong Bai, Gun Lee, and Mark Billinghurst (2016). "An oriented point-cloud view for MR remote collaboration". In: *SIGGRAPH ASIA 2016 Mobile Graphics and Interactive Applications*. ACM, p. 8.

Garon, Mathieu, Pierre-Olivier Boulet, Jean-Philippe Doironz, Luc Beaulieu, and Jean-François Lalonde (2016). "Real-Time High Resolution 3D Data on the HoloLens". In: *Mixed and Augmented Reality (ISMAR-Adjunct), 2016 IEEE International Symposium on*. IEEE, pp. 189–191.

Google (2016a). *Tilt Brush*. URL: `https://www.tiltbrush.com/` (visited on 02/10/2018).

— (2016b). *Virtual Art Sessions*. URL: `https://developers.google.com/web/showcase/2016/art-sessions`.

Groen, Eric L and Jelte E Bos (2008). "Simulator sickness depends on frequency of the simulator motion mismatch: An observation". In: *Presence* 17.6, pp. 584–593.

Harms, Chad and Frank Biocca (2004). "Internal consistency and reliability of the networked minds measure of social presence". In:

Harmsen, Emiel (2017). "IMPRESS Immersive Presence System using an Universal Volumetric Scene Pipeline".

Hartmann, Tilo, Werner Wirth, Peter Vorderer, Christoph Klimmt, Holger Schramm, and Saskia Böcking (2015). "Spatial presence theory: State of the art and challenges ahead". In: *Immersed in Media*. Springer, pp. 115–135.

Hauswiesner, Stefan, Matthias Straka, and Gerhard Reitmayr (2011). "Free viewpoint virtual try-on with commodity depth cameras". In: *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*. ACM, pp. 23–30.

Henry, Peter, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox (2012). "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments". In: *The International Journal of Robotics Research* 31.5, pp. 647–663.

Huang, Weidong and Leila Alem (2011). "Supporting hand gestures in mobile remote collaboration: a usability evaluation". In: *Proceedings of the 25th BCS Conference on Human-Computer Interaction*. British Computer Society, pp. 211–216.

Izadi, Shahram, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. (2011). "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera". In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, pp. 559–568.

Kainz, Bernhard, Stefan Hauswiesner, Gerhard Reitmayr, Markus Steinberger, Raphael Grasset, Lukas Gruber, Eduardo Veas, Denis Kalkofen, Hartmut Seichter, and Dieter Schmalstieg (2012). "OmniKinect: real-time dense volumetric data acquisition and applications". In: *Proceedings of the 18th ACM symposium on Virtual reality software and technology*. ACM, pp. 25–32.

Kammerl, Julius, Nico Blodow, Radu Bogdan Rusu, Suat Gedikli, Michael Beetz, and Eckehard Steinbach (2012). "Real-time compression of point cloud streams". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, pp. 778–785.

Kasahara, Shunichi and Jun Rekimoto (2014). "JackIn: integrating first-person view with out-of-body vision generation for human-human augmentation". In: *Proceedings of the 5th Augmented Human International Conference*. ACM, p. 46.

— (2015). "JackIn head: immersive visual telepresence system with omnidirectional wearable camera for remote collaboration". In: *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*. ACM, pp. 217–225.

Khoshelham, Kourosh and Sander Oude Elberink (2012). "Accuracy and resolution of kinect depth data for indoor mapping applications". In: *Sensors* 12.2, pp. 1437–1454.

Komiyama, Ryohei, Takashi Miyaki, and Jun Rekimoto (2017). "JackIn space: designing a seamless transition between first and third person view for effective telepresence collaborations". In: *Proceedings of the 8th Augmented Human International Conference*. ACM, p. 14.

Kowalski, Marek, Jacek Naruniec, and Michal Daniluk (2015). "Live Scan3D: A Fast and Inexpensive 3D Data Acquisition System for Multiple Kinect v2 Sensors". In: *3D Vision (3DV), 2015 International Conference on*. IEEE, pp. 318–325.

Kristoffersson, Annica, Silvia Coradeschi, and Amy Loutfi (2013). "A review of mobile robotic telepresence". In: *Advances in Human-Computer Interaction* 2013, p. 3.

Kuster, Claudia, Tiberiu Popa, Christopher Zach, Craig Gotsman, Markus H Gross, Peter Eisert, Joachim Hornegger, and Konrad Polthier (2011). "FreeCam: A Hybrid Camera System for Interactive Free-Viewpoint Video." In: *Vision, Modeling, and Visualization*, pp. 17–24.

Langmann, Benjamin (2014). "Depth Camera Assessment". In: *Wide Area 2D/3D Imaging: Development, Analysis and Applications*. Wiesbaden: Springer Fachmedien Wiesbaden, pp. 5–19. ISBN: 978-3-658-06457-0. DOI: 10.1007/978-3-658-06457-0_2. URL: http://dx.doi.org/10.1007/978-3-658-06457-0_2.

Lee, Chuen-Chien, Ali Tabatabai, and Kenji Tashiro (2015). "Free viewpoint video (FVV) survey and future research direction". In: *APSIPA Transactions on Signal and Information Processing* 4, e15.

Maimone, Andrew and Henry Fuchs (2011). "Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras". In: *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, pp. 137–146.

— (2012a). "Real-time volumetric 3D capture of room-sized scenes for telepresence". In: *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2012*. IEEE, pp. 1–4.

— (2012b). "Reducing interference between multiple structured light depth sensors using motion". In: *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*. IEEE, pp. 51–54.

Mehrotra, Sanjeev, Zhengyou Zhang, Qin Cai, Cha Zhang, and Philip A Chou (2011). "Low-complexity, near-lossless coding of depth maps from kinect-like depth cameras". In: *Multimedia Signal Processing (MMSP), 2011 IEEE 13th International Workshop on*. IEEE, pp. 1–6.

Microsoft (2016). *Hololens*. URL: https://www.microsoft.com/en-us/hololens.

Moreno, Carlos, Yilin Chen, and Ming Li (2017). "A dynamic compression technique for streaming kinect-based Point Cloud data". In: *Computing, Networking and Communications (ICNC), 2017 International Conference on*. IEEE, pp. 550–555.

Moreno, Carlos and Ming Li (2016). "A Comparative Study of Filtering Methods for Point Clouds in Real-Time Video Streaming". In: *Proceedings of the World Congress on Engineering and Computer Science*. Vol. 1.

Nicholson, Scott (2016). "The State of Escape: Escape Room Design and Facilities". In: *Meaningful Play 2016. Lansing, Michigan.*

Niehorster, Diederick C, Li Li, and Markus Lappe (2017). "The accuracy and precision of position and orientation tracking in the HTC vive virtual reality system for scientific research". In: *i-Perception* 8.3, p. 2041669517708205.

Obermeier, Christian, Thomas Dolk, and Thomas C Gunter (2012). "The benefit of gestures during communication: evidence from hearing and hearing-impaired individuals". In: *cortex* 48.7, pp. 857–870.

Orts-Escolano, Sergio, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. (2016). "Holoportation: Virtual 3d teleportation in real-time". In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, pp. 741–754.

Paulos, Eric and John Canny (2001). "Social tele-embodiment: Understanding presence". In: *Autonomous Robots* 11.1, pp. 87–95.

Pouliquen-Lardy, Lauriane, Isabelle Milleville-Pennel, François Guillaume, and Franck Mars (2016). "Remote collaboration in virtual reality: asymmetrical effects of task distribution on spatial processing and mental workload". In: *Virtual Reality* 20.4, pp. 213–220.

Rogers, Brian and Maureen Graham (1979). "Motion parallax as an independent cue for depth perception". In: *Perception* 8.2, pp. 125–134.

Rusu, Radu Bogdan and Steve Cousins (2011). "3d is here: Point cloud library (pcl)". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, pp. 1–4.

Sarbolandi, Hamed, Damien Lefloch, and Andreas Kolb (2015). "Kinect range sensing: Structured-light versus time-of-flight kinect". In: *Computer vision and image understanding* 139, pp. 1–20.

Schubert, Thomas, Frank Friedmann, and Holger Regenbrecht (2001). "The experience of presence: Factor analytic insights". In: *Presence: Teleoperators & Virtual Environments* 10.3, pp. 266–281.

Slater, Mel (2003). "A note on presence terminology". In: *Presence connect* 3.3, pp. 1–5.

Smolic, Aljoscha (2011). "3D video and free viewpoint video—From capture to display". In: *Pattern recognition* 44.9, pp. 1958–1968.

Smolic, Aljoscha, Karsten Mueller, Philipp Merkle, Christoph Fehn, Peter Kauff, Peter Eisert, and Thomas Wiegand (2006). "3D video and free viewpoint video-technologies, applications and MPEG standards". In: *Multimedia and Expo, 2006 IEEE International Conference on*. IEEE, pp. 2161–2164.

Sohn, Bong-Soo, Chandrajit Bajaj, and Vinay Siddavanahalli (2004). "Volumetric video compression for interactive playback". In: *Computer Vision and Image Understanding* 96.3, pp. 435–452.

Steed, Anthony, William Steptoe, Wole Oyekoya, Fabrizio Pece, Tim Weyrich, Jan Kautz, Doron Friedman, Angelika Peer, Massimiliano Solazzi, Franco Tecchia, et al. (2012). "Beaming: an asymmetric telepresence system". In: *IEEE computer graphics and applications* 32.6, pp. 10–17.

Tait, Matthew and Mark Billinghurst (2015). "The effect of view independence in a collaborative ar system". In: *Computer Supported Cooperative Work (CSCW)* 24.6, pp. 563–589.

Tecchia, Franco, Leila Alem, and Weidong Huang (2012). "3D helping hands: a gesture based MR system for remote collaboration". In: *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*. ACM, pp. 323–328.

The Wachowskis (1999). *The Matrix*. Warner Bros. Pictures.

Unity Technologies (2005). *Unity®*. URL: `https://unity3d.com/`.

Witmer, Bob G and Michael J Singer (1998). "Measuring presence in virtual environments: A presence questionnaire". In: *Presence: Teleoperators and virtual environments* 7.3, pp. 225–240.

Wu, Ying Choon and Seana Coulson (2007). "How iconic gestures enhance communication: An ERP study". In: *Brain and language* 101.3, pp. 234–245.

Zhao, Wenyi, David Nister, and Steve Hsu (2005). "Alignment of continuous video onto 3D point clouds". In: *IEEE transactions on pattern analysis and machine intelligence* 27.8, pp. 1305–1318.

Appendices

Appendix A

Main Evaluation Appendices

A.1 Experiment Sequence Tables

1	1	0	0	0	0
0	0	0	1	0	1
1	1	0	1	0	0
1	0	1	0	1	0
0	0	0	1	1	0
0	1	0	0	1	1

	Baseline first	**Variation first**
No Embodiment		
Desktop		
Fixed		

O = **FALSE** maze 1 first
X = **TRUE** maze 2 first

A.2 Experiment Checklist

	RO	**OSO**
Name		
Nr		

Check	Where	What
	INTERACT	• Reset blocks on the ground • Prepare questionnaire + randomizer tabs • Prepare consent form
	PLAY	• Prepare questionnaire tabs • Prepare consent form
	RECHARGE	*"Thanks for joining in this experiment. Together you'll go on a mission to retrieve a safe code while being physically at two different locations. You'll do two sessions in total and at the end of each session you receive one half of the code. During this whole experiment you may have to wait sometimes while I'm assisting the person in the other room."* To RO: *"please wait here for a moment"* To OSO: *"please follow me"*
	INTERACT	• let OSO sit and fill in consent form
	PLAY	• let RO sit and fill in consent form
	INTERACT	• Let OSO fill in questionnaire
	PLAY	• Let RO fill in questionnaire
	INTERACT	• Let OSO read instructions
	PLAY	• Let RO read instructions

		BEFORE RUN
	INTERACT	• Put on glasses, start OSOTool, scan marker
	INTERACT	*"You'll be getting help from your partner through the Hololens, I'll help you put it on. Do you see how the strap sits on my head?"* *"Please never cover the front of the glasses with your hands, or anything else"* • Help participant put them on. *"Can you see all edges of the red rectangle?* *It could happen that the glasses stop working for a moment and show an image with mountains and a message saying: **"Trying to map your surroundings"**, if this happens please say so at that moment.* *If you could stand here now, we will do a short training with your partner before we start. Afterwards I'll ask you to move to the start position."*
	INTERACT	• Start camera recording
	INTERACT	*"I will start a Skype call on the laptop so you can hear me and I can give further instructions from the other room"* • Start skype call • Disable mic on phone *"I'll go to setup your partner now, please wait here, we will make contact in a moment."*

	PLAY	• Open correct RO scene
		• Get random block order
		  ○ Select blocks in unity
		  ○ Select blocks in motive
		• Check and select maze version
		• Check show full code

	PLAY	• Start camera recording

		VR CONDITION
	PLAY	• Check webcam USB unplugged
	PLAY	• Help RO put on VR glasses, and headphones
	PLAY	• Start unity scene • Enable mic on phone
	PLAY	*"You should be able to see and hear each other now.* *OSO, can you look away from RO? You should see an arrow appear that points towards, RO's head.* *RO, Please hold your hands in front of you.* *OSO, can you see the hands?* *RO, now cross your hands, you'll notice they start behaving weirdly, sometimes you'll have to remove your hands from your view and re-enter them to fix this. Please lower them now.* *OSO, did you notice the hands disappear?* *RO, can you place your finger on top of the pole on the edge of the wall in the middle of the room?* *OSO, is that correct?* *Now, shake each other's hand.* *RO, please find the wall with the text about the keys, this is where instructions will appear."* *"Alright, we're ready to start.* *OSO, please go to your start position.* *I'll stop talking as soon as the experiment starts, that's also the moment when the beams will become active and you can start the mission.* *Three… Two… One…."* • Press 'U' 'I' keys • Mute phone mic

		DESKTOP CONDITION
	PLAY	• Check webcam USB plugged in and Vive unplugged
	PLAY	• Explain fps controls
	PLAY	• Start unity scene • Enable mic on phone
	PLAY	*"You should be able to see each other now. Please hold your hands in front of you.* *OSO, can you look away from RO? You should see an arrow appear that points towards, RO's head.* *OSO, can you see the hands?* *RO, now cross your hands, you'll notice they start behaving weirdly, sometimes you'll have to remove your hands from your view and re-enter them to fix this. Please lower them now.* *OSO, did you notice the hands disappear?* *RO, can you place your finger on top of the pole on the edge of the wall in the middle of the room?* *OSO, is that correct?* *Now, shake each other's hand.* *RO, please find the wall with the text about the keys, this is where instructions will appear."* *"Alright, we're ready to start.* *OSO, please go to your start position.* *I'll stop talking as soon as the experiment starts, that's also the moment when the beams will become active and you can start the mission.* *Three… Two… One…."* • Press 'U' 'I' keys • Mute phone mic

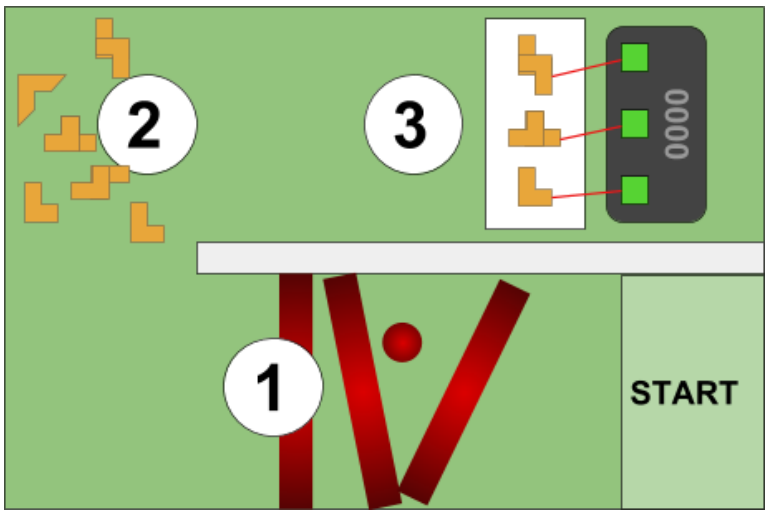
		AFTER RUN
	PLAY	• Press ‘O’ ‘P’ keys • Enable phone mic *“Well done! OSO, I’ll be there in a minute”* • Disable phone mic • Stop unity scene *“RO, you can take the headset off.* *Please take a seat and fill in this questionnaire”* • Turn off camera
	INTERACT	“*you can take the headset off.* *Please take a seat and fill in this questionnaire”* • Turn off camera

		END
	PLAY	If both are done with the questionnaire. “*Please walk with me to the other room”*
	INTERACT	*“You should be able to open the safe now”* • They find candy
	INTERACT	*“Do you still have any questions?”* • Show RO the room and maybe explain about how the system works.
	INTERACT	• Turn off camera mic • Put Hololens on charger

A.3 OSO Instructions

Instructions for On-site Operator

At the end of this room a safe code is stored which can open the safe in front of you.
Your assignment is to retrieve this code and open the safe.
The code is protected by three assignments that you'll have to overcome to crack the code, each of these assignments are explained below.
To successfully retrieve the code you'll need help from your remote partner, who has essential information on each assignment.



1. Radiation beams
Dangerous beams of radiation are blocking your path. One problem; they are invisible. Luckily your remote partner has the equipment to spot them and guide you safely around them.
While radiation is bad for your whole body, make sure that especially your head doesn't get into the beam!

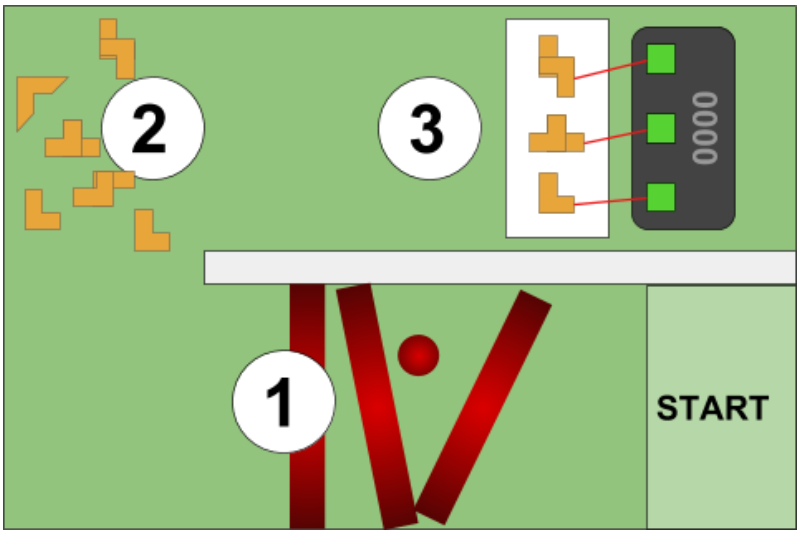
2. Keys
Six key blocks with different shapes are lying on the ground, you need to collect three specific ones to use in the next step. Your remote partner knows exactly which of the keys you'll need to take.

3. Key alignment
Each key, from the previous assignment, will emit an invisible laser beam in a random direction that only your remote partner can see. Position the three blocks correctly on the table to point the lasers on to the markers on the screen. Once each of the markers are lit by a laser pointer, part of the code will be displayed on the screen.

A.4 OSO Instructions

Instructions for Remote Operator

Your partner is on a mission to retrieve a code to open a safe. Your assignment is to help him or her accomplish that mission. By using openIMPRESS you can help your partner by being virtually present in the same room and guide him or her through the three assignments that protect the code, each of these assignments are explained below. Your support is essential for the success of the mission.



1. Radiation beams
Dangerous beams of radiation are blocking your partner's path. One problem: your partner can't see them. Luckily you have the equipment to spot them and guide your partner safely around them.
While radiation is bad for your partner's whole body, make sure that especially his or her head doesn't get into the beam!

2. Key shapes
Six key blocks with different shapes are lying on the ground, your partner needs to collect three specific ones to use in the next step. The keys that need to get collected are shown to you in VR.

3. Laser alignment
Each of the three keys will emit an invisible laser beam in a random direction that only you can see. Make sure the three keys get positioned correctly on the table. The lasers need to point to the markers behind the table. Once each of the markers are lit by a laser pointer, part of the code will be displayed on the screen.

A.5 Consent form

PP nr.

CONSENT FORM

Project title: openIMPRESS
Researcher: Emiel Harmsen

Data collection explanation: You have been invited to participate in an experiment in virtual and augmented reality. You and another participant will be located in different rooms. Depending on your role within this experiment you will be asked to either wear a Head Mounted Display (HTC Vive) or Augmented Reality Glasses (Microsoft Hololens). The person wearing the Head Mounted Display will be virtually present in the room with the other participant. Together you will go on a mission to retrieve a four digit code to unlock the safe. The mission consists of three assignments for which you'll have to collaborate in order to complete them.

The goal of this experiment is to evaluate openIMPRESS, an open source remote presence system.
Video recordings made during the test will be used to analyze the interactions, and responses of the system. The total duration of the experiment is approximately 60 minutes. The recorded material will not be shared and will be dealt with anonymously, with exception of the videos or snapshots of the videos that will be shared for publication and/or explanation of our research activities. We will blur your face if you prefer.

Please check the boxes.	
1. I have read the explanation and I understand that I can ask questions at any time.	☐
2. I understand that I can quit at any time, without having to give a reason, and that my test will not be part of any data analyses.	☐
3. I give permission for my data to be used for the goals of this test and for future research into VR-telepresence applications.	☐
4. I give permission for the use of my video recordings to be shared in (popular) scientific presentations and written publications.	☐

5. I prefer that any footage will be blurred to make me unidentifiable for the purposes mentioned in point 4.

○ **yes**, please blur ○ **no**, use images as is, I don't mind my face being visible

_____
Name participant

_____ _____
Signature participant Date

_____
Signature researcher