UNIVERSITY OF THE WITWATERSRAND

MASTER'S DISSERTATION

Mixed Reality Simulators

Author:
Natalie AUSMEIER

Supervisor:
Prof. Turgay CELIK

A Dissertation submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science

Pretoria, May 2017

Declaration of Authorship

I, Natalie AUSMEIER, declare that this Dissertation is my own, unaided work. It is being submitted for the Degree of Master of Science at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University.

_____

(Signature of candidate)

_____day of_____20_____in_____

“It is often said that before you die your life passes before your eyes. It is in fact true. It’s called living.”

Terry Pratchett

Abstract

Virtual Reality (VR) is widely used in training simulators of dangerous or expensive vehicles such as aircraft or heavy mining machinery. The vehicles often have very complicated controls that users need to master before attempting to operate a real world version of the machine. VR allows users to safely train in a simulated environment without the risk of injury or damaging expensive equipment in the field. VR however visually cuts off the user from the real environment, which may obtain obstructions. Users are unable to safely move or gesture while wearing a VR headset. Additionally users are unable to use standard input devices such as mice and keyboards. By mixing in a live view of the the real world, the user can still see and interact with the physical environment. The contribution of this research is presenting ways of using *Mixed Reality* to enhance the user experience of traditional VR based simulators. *Mixed Reality* improves on traditional VR simulators by allowing the user the safety and freedom of not being cut off from the real world, allowing interaction and the tactile feedback of interacting with complex physical controls, while still allowing simultaneous use of virtual controls and by adding a real world reference point to aid in diminishing simulator sickness caused by visual motion.

Acknowledgements

I would like to acknowledge the inputs and encouragement from Turgay Celik, Jaco Cronje, Jason De Villiers, Bernardt Duvenhage, Asheer Bachoo and David Baxter.

Contents

List of Figures

List of Tables

Listings

Abbreviations

2D	**(2)**Two **D**imensional
3D	**(3)**Three **D**imensional
AR	**A**ugmented **R**eality
CMOS	**C**omplementary **M**etal **O**xide **S**emiconductor
FOV	**F**ield **O**f **V**iew
GPU	**G**raphics **P**rocessing **U**nit
ICD	**I**nter **C**amera **D**istance
IPD	**I**nter **P**upillary **D**istance
IR	**I**nfra **R**ed
LD	**L**abyrinthine **D**efectives
LED	**L**ight **E**mitting **D**iode
MR	**M**ixed **R**eality
NASA	**N**ational **A**eronautics and **S**pace **A**dministration
OST	**O**ptical **S**ee **T**hrough
PSO	**P**article **S**warm **O**ptimisation
QTE	**Q**uick **T**ime **E**vent
VR	**V**irtual **R**eality

Chapter 1

Introduction

1.1 Background

Virtual Reality (VR) places a user in a computer simulated environment, aimed to mimic real world scenarios. VR is widely used in training simulators of dangerous or expensive vehicles such as aircraft or heavy mining machinery. The vehicles often have very complicated instruments and controls that users need to master before attempting to operate a real world version of the machine. VR allows users to safely train in a simulated environment without the risk of damaging expensive equipment in the field.

Figure 1.1 demonstrates the concept of entering a virtual world by means of computer simulation, the user is using a computer running simulation software to drive a virtual car.

FIGURE 1.1: The Concept of Virtual Reality.

There are two ways in which traditional simulators approach VR, *Hardware Console Based Simulators* and *Virtual Console Based Simulators*.

1.1.1 Hardware Console Based Simulators

A scale replica of the vehicle console is built containing the exact instruments and controls as its real world counter part. These simulators often operate on a base capable of simulating the motion of the vehicle as well.

FIGURE 1.2: Example of a Hardware Console Based Simulator from Thales [1].

Figure 1.2 shows an example of a Thales [1] aircraft simulator with physical cockpits representing the exact model of each aircraft.

The hardware console perfectly matches and functions like that of the real aircraft or vehicle and the motion bases matches the movement of the real vehicle. Projectors or multiple monitors usually provide the user with an out the window view of a virtual world. The hardware console, motion base and surround view makes for a very realistic simulator experience.

This approach is however very costly and requires a fixed setup in a simulator room or transportable container. If the simulator is completely software based, it can easily be setup anywhere for mobile training solutions and at a lower cost.

Hardware console based simulators are also very specific to the model of the vehicle it was designed for; the entire console needs to be replaced for a different model of vehicle being trained. In a virtual or software based simulator, the same virtual world could be used with various vehicle models.

As discussed in section 2, a hardware console based simulator is more likely to cause simulator sickness than simulators with no motion base. This is partly due to discrepancies between the physical movement of the motion base and the visuals of the simulator.

Table 1.1 summarises the pros and cons of a hardware console based simulator.

Pros	**Cons**
The real instruments and controls of the vehicle are used	The system can be costly
The physical controls give tactile feedback	The system can't easily be moved and used at different locations
The vehicle can physically move and rotate on top of a motion base	Motion base movement does not always perfectly match the visuals. Motion bases are more likely to cause simulator sickness during use
There is often a 360° surround view of the world	The instruments and controls are for a specific vehicle model, and can't easily be swapped out

TABLE 1.1: Pros and Cons of a Hardware Console Based Simulator

1.1.2 Virtual Console Based Simulators

A virtual representation of the vehicle console is represented in the simulated environment. This approach is completely software based, the vehicle instruments, controls and out the window view are all virtual. The virtual controls and instruments are usually not interactive and the simulator is controlled using generic input devices.

FIGURE 1.3: Example of a Virtual Console Based Simulator from MS Flight [2].

Figure 1.3 shows an example of Microsoft Flight [2], a commercially available flight simulator with virtual instruments and controls. This allows different models of aircraft to be easily swapped in. The virtual controls can be combined with physical controls such as flight yokes or steering wheels for a more realistic and tactile experience. Furthermore additional displays can be added for a larger field of view.

As with hardware console based simulators, adding projectors or monitors makes the system more costly. Adding hardware also decreases the mobility of the system.

Virtual console based simulators do not provide any tactile feedback with the actual instruments and controls of the vehicle, making it a much less realistic simulator experience.

Table 1.2 summarises the pros and cons of a virtual console based simulator.

Pros	Cons
Virtual consoles for different vehicle models can easily be swapped in	The virtual console can't be interacted with physically
Generic input devices can still provide tactile feedback	The system does not usually simulate physical movement via a motion base
A larger field of view can be added through additional display support	While more displays can be added, the system does not provide a 360° surround view of the world
The system relatively cheap	More displays means more cost and less mobility
The system is relatively easy to move and use at different locations	

TABLE 1.2: Pros and Cons of a Virtual Console Based Simulator

1.2 Motivation

To increase the immersion of virtual console based simulators, *Virtual Reality* (VR) headsets are often used, removing the need for costly and often bulky displays such as monitors and projectors. An example of an VR headset is shown in Figure 1.4.

FIGURE 1.4: Example of a Virtual Reality Headset.

VR headsets used in current simulators, consist of a helmet with a small display in front of the wearer's eyes. The wearer's head is tracked and the display correctly reflects the direction the wearer is looking in. The participant is effectively taken out of the real world and placed within the simulated environment with a complete 360° view of the world. Data gloves are often used to track wearer's hands within the simulated environment.

There are however some limitations when using traditional VR headset in virtual console based simulators:

Users are visually cut off from the real environment. Users are fully immersed in a virtual world, not seeing any real obstacles or people around them. This could lead to physical injury to the user or other people. The user could for example walk into obstructions, knock over items or make contact with people while gesturing. The real environment might change without the user knowing.

Users can't interact with complex physical controls. While simple input devices can be used by touch alone, more complicated levers, switches and dials as would be found in heavy mining machinery would not be usable without seeing it. Users are unable to see their own hands and interact with physical controls such as steering wheels and flight yokes.

Virtual instruments and controls do not provide any tactile feedback. Complicated levers, switches and dials could be interacted with on a virtual console, but this would be missing the tactile feedback of physical controls.

Latency between real and virtual hands. Some VR simulators would track and render virtual representations of the user's hands. Depending on the hand tracking used, there is a varying delay between the user's hand movement and the update of the virtual hand visuals. There is not a one to one relationship between the user's hands in the real and virtual environment, making it difficult for the user to operate delicate controls and instruments.

Data gloves are usually tethered, uncomfortable and restrictive. To track the user's hands, tethered gloves are sometimes used. Data gloves restricts the user's movement and is uncomfortable to wear.

Users of closed off headsets sometimes experience motion sickness. Section 2 discusses usability studies showing that headsets are more likely to cause disorientation, nausea and other symptoms of simulator sickness [3] [4] [5] under simulator users.

Optical-See-Through(OST) headsets used for Augmented Reality (AR) applications, use displays that are see through or transparent, allowing the wearer to see the real environment as if they were wearing common reading glasses. These displays are however also capable of displaying rendered virtual objects within the real environment.

The Epson Moverio BT-200 [6] smart glasses shown in Figure 1.5 is an example of AR glasses that connects to a supplied Android control unit, and provides true augmented reality through its two screens. The Moverio consists of two displays that cast a 960x540 pixel image over each of your eyes.

FIGURE 1.5: Example of a Augmented Reality Headset [6].

Similar to VR headsets, OST headsets also make use of head tracking and is capable of displaying virtual objects relative to the wearer's gaze. However, OST headsets also have some limitations for use with virtual console based simulators.

A complete 360° **view is not possible**. The displays of current OST headsets produce a fairly narrow field of view. For example a single display of the Moverio BT-200 has a field of view of only 23°. This is only useful for AR applications where virtual objects are overlayed such as text or small virtual objects. When viewing a virtual world through these displays a very narrow view of the virtual world would be visible. The approximate field of view of a human

eye is 200°, the combined field of view of the BT-200 is 46°, meaning that OST headsets like the Epson Moverio BT-200 can only displays virtual object in the very center of the user's view, greatly decreasing immersion.

An opaque view is not possible. The semi-transparent displays of OST headsets always allow the real world to show through. Only when a completely black screen is rendered to the display, will it appear opaque to the wearer. These displays are more suited for displaying overlays like reticles or text to augment reality. For a simulator application it would be preferable to only see the virtual world, but being able to switch to a real world view on demand.

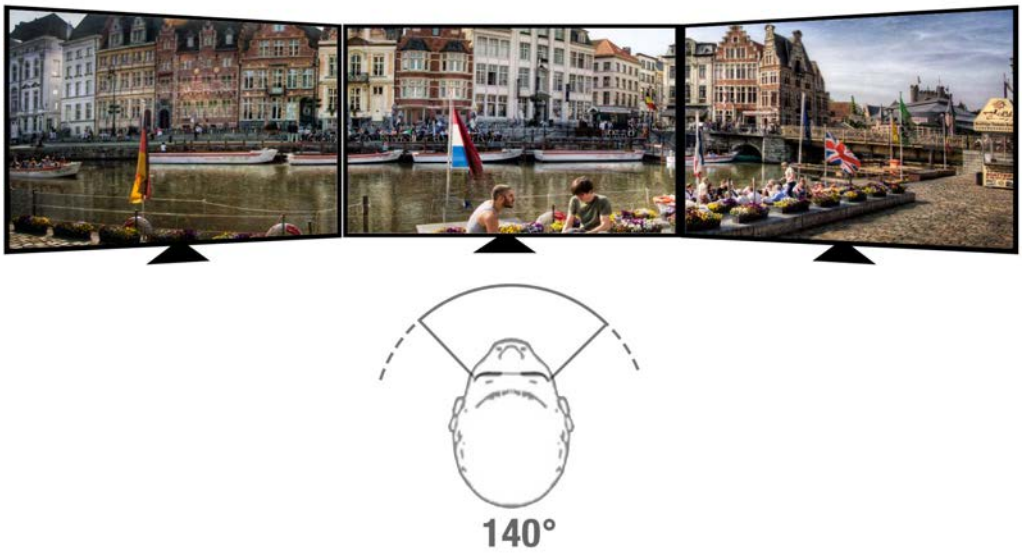
1.3 Aim of the Research

This research aims to create a *Mixed Reality* [7][8] prototypes for virtual or hardware console based simulators. *Mixed Reality* refers to the merging of the real and virtual world in such a way that real and virtual objects co-exist and interact with each other. For example, the user's real hands could interact with a hardware console video feed that is superimposed over the virtual world.

One approach is that virtual object take the form of configurable virtual screens. These screens can be placed anywhere and be any size, distance or angle relative to the user. They can surround the user 360° and also be placed above or below the user. Because these screens are virtual they can be interacted with, allowing for virtual control panels and instruments to be implemented. Additionally these virtual screens can be see-through or opaque. The user can still see and interact with the real environment and thus use physical input devices, such as mice, keyboards, flight yokes or custom simulator consoles.

The concept is effectively a combination of a physical multi-monitor setup, and a 360° VR headset. Figure 1.6 shows a physical multi-monitor view in (a) and a VR headset view in (b). By combining the two, we can have the immersion of the VR headset combined with infinitely configurable simulator setups, still allowing a view of the real environment.

This research is aimed specifically at immersive training simulators. The goal for this research is to be surrounded by a virtual scene, but to be able to look down and see your hands, take a sip of coffee or use physical controls.



(A) Multi-Monitor View

(B) VR Headset View

FIGURE 1.6: A Multi-Monitor Setup vs. a VR Headset.

In order to mix the real and virtual worlds, a combination of a VR headset and a OST headset is needed. Figure 1.7 shows a closed off VR headset next to a OST headset. By combining these two, we can have the immersion of VR, while still allowing real world interaction.

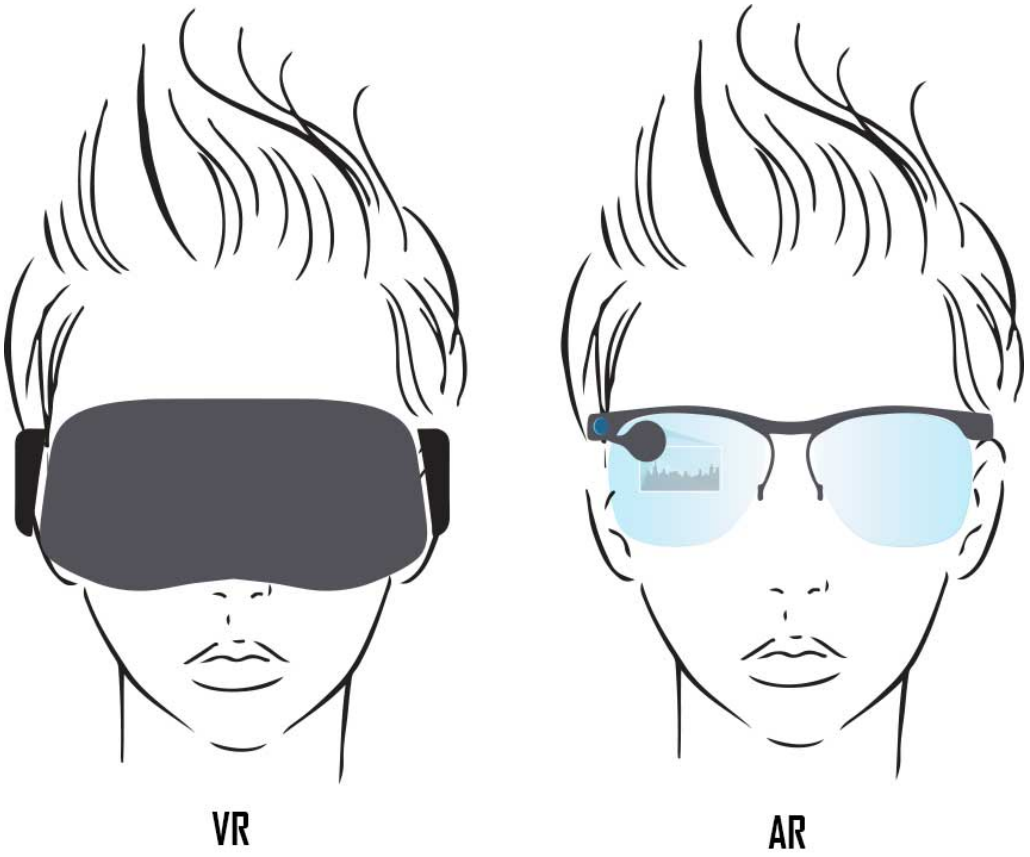


FIGURE 1.7: A VR Headset and an Optical-See-Through Device.

By making these virtual screens interactive, custom virtual instruments and control can be manipulated along with generic input devices such as keyboards and off-the-shelf joysticks. Tethered, glove based or marker based systems would interfere with the use of physical controls, so a bare-hands input mechanism is required.

The ultimate aim of the proposed research is to create an immersive yet flexible simulator that should be cheap and lightweight. Traditional simulators require bulky, custom hardware, while VR simulators do not allow the use of physical controls.

1.4 Hypotheses

By combining real and virtual worlds in a mixed reality simulator the research attempts to answer the following questions.

> Would the **immersion** of a VR headset be maintained, but with the added **usability** of allowing interaction with real and virtual objects? Additionally, would keeping track of real world objects add to the **comfort** of the simulator and prevent or mitigate simulator sickness?

1.4.1 Immersion

In order to maintain the 360° wide field of view of the virtual world, a VR headset is needed. VR headsets allowing video see-through is however not readily available at the time of writing. A custom video see-through solution would need to be built. The see-through video should match the user's field of view so that physical objects appear where they really are relative to the user. The virtual and real world systems would need to be aligned for a seamless mixed reality experience.

1.4.2 Usability

A benefit of a mixed reality simulator is the ability to interact with virtual and physical instruments and controls. Physical controls would need to be tracked and separated from the background to be viewed amongst virtual objects. A sense of depth would be needed to be able to comfortably interact with real word objects. The real objects would need to appear exactly where they are relative to the user for easy interaction. To interact with virtual controls, a user's hands and fingers would need to be tracked.

1.4.3 Comfort

Users suffering from discomfort or simulator sickness should have the ability to add real world cues to the simulator. The video latency should be low enough to not cause any additional discomfort or visual cue discrepancies.

1.5 Limitations of the Research

This research does not set out to design new hardware or a novel headset specifically for *Mixed Reality* simulators. VR headsets allowing video see-through is however not readily available at the time of writing. Because of this constraint a custom hardware solution is needed. Certain hardware limitations regarding video latency and resolutions is assumed, as off the shelf cameras not matching the specifications of the headset would have to be used. These limitations may impact the usability of the system. Furthermore, computationally expensive alignment steps would need to take place to match up the visual and IR cameras needed for interacting with virtual objects.

1.6 Outline of the Dissertation

The dissertation is structured as follow. Chapter 2 explores the current state of the art and related work. Chapter 3 presents the methodology used for implementing simulator prototypes, demonstrating the novel use of *Mixed Reality* with custom video see-through hardware and a more mobile hardware platform. Chapter 4 explores objective and subjective test results to evaluate the system performance and usability. Chapter 5 presents conclusions on the test result findings and recommendations for future work.

Chapter 2

Related Work

Looking at current technology and research, the three main focus areas are:

- **Mixed Reality**. The user should be able to see and interact with virtual and real objects interchangeably.
- **Virtual Input Devices**. The user should be able to interact with virtual controls.
- **Simulator Sickness**. The experience should mitigate the chances of the user experiencing simulator sickness.

2.1 Mixed Reality

Ohta and Tamura [7] provide an in-depth look at the current state of Mixed Reality and some of its applications. One example use of Mixed Reality is combining live and virtual performance art, as explored by Benford and Giannachi [8]. It has however not been used to combine real world inputs with virtual simulations.

Mixed reality applications are usually developed for Optical-See-Through (OST) headsets. While current Optical-See-Through (OST) headsets allows users to stay aware of their surroundings, OST headsets are not suited for immersive virtual reality experiences. This is due to two factors, the narrow field of view (FOV) of the displays, and the inherent see through nature of the projected glass displays, not allowing for a completely opaque view.

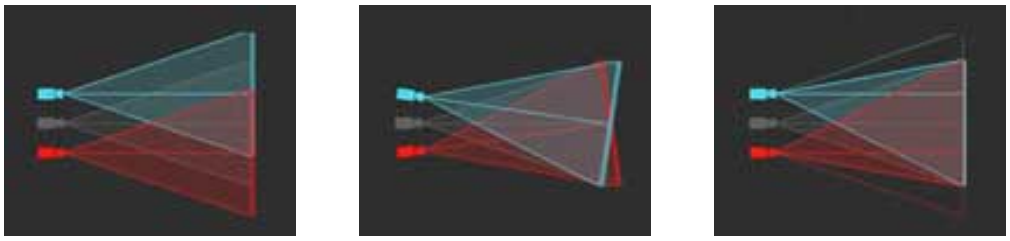
Virtual reality headsets, on the other hand, do not provide the real world view necessary. Another option referred to as a video see-through headset, essentially combines a VR and OST headsets. Instead of semi-transparent glass, a live feed from a video cameras is used. In fact a stereo pair of cameras are used, one for the left and one for the right eye. Rolland et al. [9] compares the optical and video see-through devices with respects to design, build process and usage. Many of the issues regarding occlusion and registration described by Rolland et al. has since been addressed in current headsets and software libraries.

Figure 2.1 shows an example of a stereo camera attachment from OVRVision [10] combined with a VR headset.

FIGURE 2.1: Example of a Video See-Through Headset [10].

Steptoe [11] in his article on building a video see-through headset showcase, describes the camera considerations with regards to resolution, refresh rate, FOV as well as mounting. Steptoe notes that there are three options for mounting the stereo camera pair. Figure 2.2 shows the each mounting option as illustrated in the 3D rendering software, Lightwave [12].

Mounting the cameras parallel would mean that the optical axis of the two cameras overlap at infinity. This is not desirable as objects closer than infinity appears at a negative parallax, in



(A) Parallel (B) Toe-in (C) Off-axis

FIGURE 2.2: Mounting Options for Stereo Camera Pair [12]

front of the stereo plane. In Figure 2.3 from Miriam Ruthross' [13] 3D cinema tutorial, the objects at infinity would appear at the stereo plane and anything closer at negative parallax.

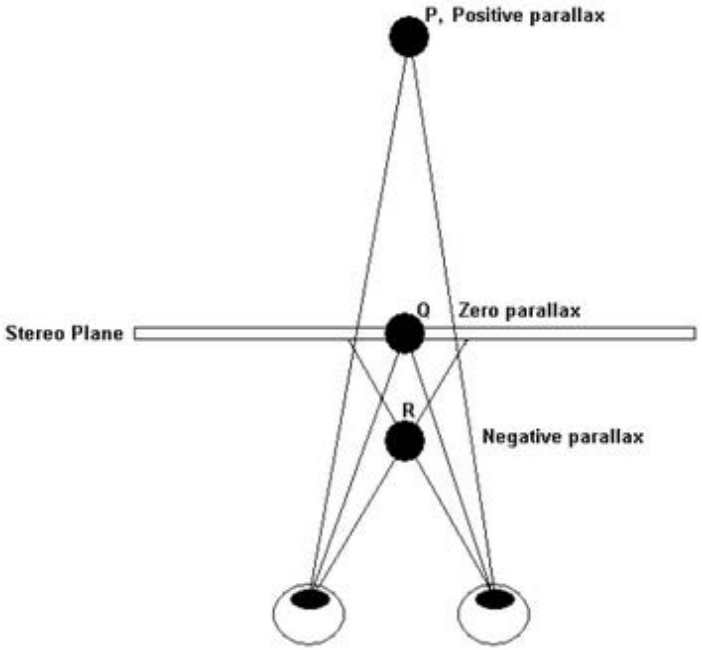


FIGURE 2.3: Stereo Camera Parallax [13].

Mounting them toed-in as shown in Figure 2.2 means that the cameras are rotated inwards so that their optical axis intersect. This method correctly displays some object in front and some behind the stereo plane. In Figure 2.3 some objects would thus also appear at positive parallax. Steptoe however warns that toe-in mounting produces vertical parallax because of the rotation angles.

Steptoe used toed-in mounting for his AR showcase but notes that lens shift or off-axis mounting as shown in Figure 2.2 should be used instead. The lens is physically shifted horizontally relative

to the sensor, creating the desired overlap and positive and negative parallax. This method however requires a custom built camera.

To be able to render virtual objects in the stereo camera feed the video and virtual spaces needs to be aligned. Steptoe calculates the FOV and angular distribution of the stereo cameras and determines that it is generally in alignment with the virtual camera. Some undistortion needs to be done to compensate for the radial distortion of the consumer cameras used. Steptoe notes that camera distortion for AR applications is very important as it makes it difficult for users to estimate distances and sizes.

Figure 2.4 shows the AR showcase from Steptoe [11] with the stereo camera feed and some virtual objects, such as the rendered character, shadow, capsules, blocks and web pages.

FIGURE 2.4: Stereo Camera Feed with Virtual Objects [11].

2.2 Virtual Input Devices

The *Mixed Reality* (MR) prototypes should allow the user to see and interact with their hands, allowing the use of any physical input devices. A secondary goal however is to be able to interact with the virtual screens, allowing interactive, custom instrument panels to be developed.

The goal is to implement a non-restrictive method of interacting with virtual objects. Tethered, glove based or marker based systems would interfere with the use of physical controls, so a bare-hands input mechanism is required.

Current research presents two main methods for three-dimensional (3D) interaction with virtual objects, namely stereo visual cameras or infra-red based depth sensors.

Stereo Cameras. Jennings [14] combines several finger segmentation techniques to fit a robust finger model. The paper explains the limitations involved with each hand segmentation technique found in prior literature and combines the following approaches:

- Colour Segmentation. By using predefined hue and colour intensity values for skin colour, any skin coloured regions can easily be segmented out. This approach is fast and robust but becomes problematic if the colour of any background objects is close to skin colour. Likely fingertips are found using finger convexity features in the segmented images.

FIGURE 2.5: Colour Segmentation with Background Interference [14].

- Edge Detection. By using a method based on the Canny edge detector, an image with traced edges is produced. Similarly to colour segmented images, the traced edges are used to find likely fingertips when combined with other segmentation data.

FIGURE 2.6: Edge Detection for Combination with Colour Segmentation [14].

- Depth Image. A stereo camera system computes a depth image using a sum of squared differences correlation algorithm.

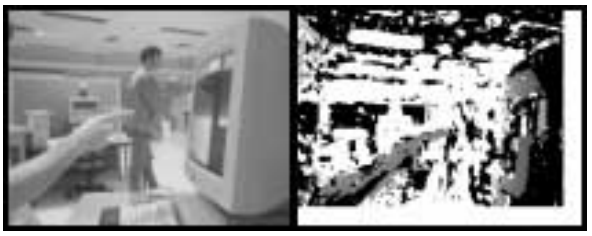


FIGURE 2.7: Depth Image from Stereo [14].

- Background Subtraction and Motion Segmentation. The background model can be easily trained and subtracted to segment out a moving hand. This approach was however found to be to sensitive to changing light conditions and not used in the final implementation.

Jennings [14] uses a model fitting technique based on Bayes Theorem to combine the various measurements. The resulting finger tracking is more robust than each of techniques on its own, coping with a vibrating camera, dynamic background objects, flesh coloured background objects as well as varying lighting conditions. Self occlusion is however not handled in this approach.

Malik [15] developed a similar stereo vision based hand tracking system using two downward facing web cameras. The system is able to track the three-dimensional (3D) position and two-dimensional (2D) orientation of the thumb and index finger of each hand. Malik adds gesture recognition for interaction purposes.

Because of the static downward facing cameras, hands are easily segmented using background subtraction. Skin colour segmentation is used to further segment out shirt sleeves. Next the contours of the segmented hands are detected and the contour with the smaller mean x coordinate is the left hand, and the contour with the larger mean x is the right hand. Peak and valley detection is then used to determine the hand features such as fingertips. Gestures can now be detected using these features, e.g. a pointing gesture would be a single peak and a pinching gesture two peaks.

The tracking system developed by Malik works well for simple pointing and pinching gestures and could be expanded to use a more sophisticated gesture recognition system.

Song, et al. [16] takes the stereo vision based system a step further, adding interaction with virtual physics based objects. Interaction involved two *Mixed Reality* games, finger fishing and Jenga, which require similar interactions to that proposed in this research. Song, et al. uses finger tracking based on Hardenberg's fingertip shape detection method [17], with improvements

to accuracy and robustness [18]. Fingertip shape detection is done by background subtraction on a static background followed by circle detection within search squares.

A usability study was conducted where participants used various forms of traditional inputs as well as the bare hands finger tracking method. The questionnaire results from 57 participants indicate that the majority of participants preferred the finger tracking to all other inputs but found it less accurate. This was also verified via objective based task performance tests, where the majority of users could complete tasks much faster, using traditional input methods. It was concluded that while most users prefer the more natural finger based interaction method, improvements to the tracking algorithm is still needed to reach the accuracy achieved with traditional input methods.

Depth Sensors. Depth sensors as found in the Microsoft Kinect [19], consists of an infra-red laser projector combined with a monochrome CMOS sensor. It captures video data in three dimensions under any lighting conditions.

(A) Kinect Depth

(B) Segmented Fingers

FIGURE 2.8: Finding Fingertips with Kinect Depth Sensor [20]

As Raheja, et al. [20] explains , this overcomes a lot of the limitations found with stereo camera based tracking. Hand segmentation methods used along with stereo or single camera systems perform poorly under certain lighting conditions, when hand motion is too rapid or when the background is cluttered or dynamic. Using depth information supplied by e.g. the Kinect was found to be more reliable and robust.

Using the Kinect depth information Raheja, et. al. used Bayesian Object Localization for hand detection and NiTE^{TM} [21] modules for hand tracking and points detection. Once the hands points are detected the depth image can be segmented using a depth threshold and the blob containing the points is then labelled as the hand. Fingertip detection is performed next by removing the palms using circle filters and then using the depth information on the remaining fingers. The finger tips were accurately found as the values with minimum depth, in other words the point on the finger closest to the camera. Similarly the centres of palms could be found by applying the distance transform on the inverted binary images of the hand. Figure 2.8 shows an example of segmented fingertips from a Kinect depth image.

Lab results indicated that Raheja, et. al were able to identify fingertips and centre of palm very accurately and efficiently, even when the fingers were bent. A 100% detection rate was obtained on detecting fingertips with open fingers and a 90% rate was obtained on detecting the centre of palms.

Oikonomidis, et al. [22] uses a variant of Particle Swarm Optimisation (PSO) to minimize the variance between parameters of a three dimensional(3D) hand model and actual hand observations as obtained by the Microsoft Kinect for 3D hand pose recovery. Both the colour and depth images from the Kinect sensor is used to perform skin colour and depth segmentation of the hand. A 3D hand model pose is generated, with a pose described by a vector of 27 parameters. These parameters are estimated in such a way to minimize the discrepancy between the hand hypotheses and the actual observations. A graphics rendering technique is used to produce skin and depth maps for the 3D hand model that can be compared to the depth maps produced by the Kinect. An appropriate objective function is thus formulated and a variant of PSO is employed to search for the optimal hand configuration.

Evaluation of this model based fitting on varied and complex hand poses resulted in 74% of poses deviating 4cm or less from the tracked poses. Oikonomidis, et al. [22] demonstrated a accurate and robust 3D hand tracking algorithm running at 15Hz, which through further GPU optimization could be sped up for true real-time usage.

From the research it is clear that when it comes to bare-hands tracking and pose estimation, better results are obtained using depth sensors combined with colour cameras. A dedicated hand and finger depth tracking device consisting of two monochromatic IR cameras and three infra-red LEDs, the Leap Motion, has become available. A second version of the Microsoft Kinect supporting higher resolution depth maps, greater depth precision and support for closer distances has also been released.

The Leap Motion [23] hand and finger tracker, shown in Figure 2.9, was chosen for prototyping. It's small size makes it easy to combine with a VR headset. The Leap Motion controller is a small USB peripheral device. Using two monochromatic IR cameras and three infrared LEDs, the device observes a roughly hemispherical area, to a distance of about 1 meter. The LEDs generate a 3D pattern of dots of IR light and the cameras generate almost 300 frames per second of reflected data, which is then sent through a USB cable to the host computer, where it is analysed by the Leap Motion controller software. The smaller observation area and higher resolution of the device differentiates the product from the Microsoft Kinect.

FIGURE 2.9: Leap Motion Hand and Finger Tracker [23].

2.3 Simulator Sickness

Simulator Sickness is a particular form of motion sickness. Motion sickness is characterised by symptoms like sweating, nausea, pallor and vomiting. Motion sickness may occur during physical motion, but can also be induced by viewing visual motion, in other words motion of the scene alone. In healthy people self, propelled motion in a natural environment, does not generally lead to motion sickness.

We differentiate between simulator sickness in a motion base system and simulator in a fixed base system.

Simulator sickness occurs when there is a discrepancy between the visual cues of position and movement and the perceived position and movement the body's proprioceptive system transmits to the brain [24][25]. In a fixed base simulator, the eyes tell the brain that the user is moving through the environment whereas the body tells the brain that it is not moving. It would stand to reason that adding a motion base would alleviate the symptoms, as the motion should more closely match the visual cues, Stern, et al. [24] however finds that the opposite is in fact true. The motion base system is not precise or fast enough to match the expected motion, and does not perfectly match the visual cues. Sterns findings show that expensive hardware console based simulators using motion bases is much more likely to cause motion sickness due to the wider field of view, and inaccurate artificial motion of the motion base.

2.3.1 Physical Motion Simulator Sickness

Motion sickness induced by physical motion for example carsickness, airsickness or seasickness, occur in artificial conditions, like a moving platform. This kind of motion sickness is also experienced in hardware console based simulators making use of motion bases. Irwin [26] has shown that people without functioning organs of balance in the inner ears, never get motion sickness. A group of deaf mute co-passengers were proven to be immune to motion sickness during a sea voyage. These people are said to be labyrinthine defectives (LD).

To minimize simulator sickness on motion basis, Sharkey, et al [27] explores the role of the motion base. The NASA Vertical Motion Simulator (VMS) was used with high-fidelity motion cues. Their aim was to reduce the discrepancy between visually implied motion and actual motion to see how this affects sickness symptoms. Pilots flew test sorties with and without a

motion base and with the motion base set at a different fidelity. The motion base condition is shown to be practically irrelevant with respect to the incidence and severity of motion sickness. The authors note that the data collection procedure could not detect differences in sickness levels, only that sickness still occurred.

2.3.2 Visual Motion Simulator Sickness

For VR based simulators we are more interested in simulator sickness that occurs for fixed base simulators. This type of visually induced motion sickness also has no effect on LD patients [28], which implies that the same discrepancies between visual cues of motion and perceived motion is at work. Bos, et al. [25] calls this the subjective vertical mismatch theory. The theory states that the mismatch between the senses, provided by artificial visual cues in the case of a closed off headset, and the subjective vertical expected by the user from previous experience, results in sickness.

Sharples, et al. [4] conducted research on the sickness symptoms experienced when comparing HMD, desktop and projection display systems. Seventy one participants took part in the experiments and were asked to fill in the Simulator Sickness Questionnaire (SSQ) before and after each period of VR exposure. Significantly higher SSQ scores were obtained for post-exposure nausea and disorientation when comparing the HMDs to the other display systems, with 68.4% of HMD users experiencing a large increase of symptoms while using the HMD.

Similarly Howarth and Costello [5] compared a HMD system with a desktop display system. Out of twenty participants, four withdrew due to headache and nausea while using the HMD system. None withdrew while using the desktop display system. Participants reported a significantly greater number of increases in general discomfort, fatigue, headache, nausea, dizziness and stomach awareness using the HMD compared to the desktop display system. General discomfort was reported by 80% of participants using the HMD system, compared to 20% using the desktop display system.

It is clear from the research that simulator sickness is much more prevalent in a closed off, fully immersive system, such as a VR headset with non see-through displays. This can be attributed to the fact that the visual cues in a VR system completely encompasses the user. Users are left with no real world reference with which to consolidate signals from the body's proprioceptive system.

The use of mixed reality allows for the flexibility to tweak how much of the real world is visible. This allows users who are sensitive to *Simulator Sickness* to configure the system to see more real world visual cues to match their subjective vertical.

Chapter 3

Methodology

3.1 Method 1: Virtual Screens

FIGURE 3.1: Virtual Screen Concept

Virtual objects take the form of configurable virtual screens. These screens can be placed anywhere and be any size, distance or angle relative to the user. Figure 3.1 shows the concept of virtual screens. Virtual screens can surround the user 360° and also be placed above the user.

Because these screens are virtual they can be interacted with, allowing for virtual control panels and instruments to be implemented. Additionally these virtual screens can be see-through or opaque. The user can still see and interact with the real environment and thus use physical input devices, such as mice, keyboards, flight yokes or custom simulator consoles.

3.1.1 Hardware Implementation

The first prototype was deployed on a Oculus Rift DK2 VR headset [29]. In order to see the real world through the VR headset, live footage from two wide lens action cameras is used. For the purposes of interacting with virtual controls, the Leap Motion [23], hand, finger and gesture tracker is used.

FIGURE 3.2: Hardware Components - An Oculus Rift, a Leap Motion and Two Visual Cameras From Top to Bottom

3.1.1.1 Camera and Leap Mounting

Figure 3.2 shows the hardware components involved. Table 3.1 shows the specification of each hardware component.

Hardware Components		
Oculus Rift DK2	Resolution Refresh Rate Field of View	960 × 1080 per eye 60 Hz 100°
Leap Motion Tracker	IR Resolution Field of View	240 × 640 × 2 cameras 135°
Visual Cameras	Resolution Field of View (Vertical)	720 × 1280 @60fps 1080 × 1920 @30fps 65°

TABLE 3.1: Specifications for Hardware Components

There are many considerations to take into account when mounting the cameras. Ideally we want to match the camera placement as closely as possible to the position of an average person's eyes. The average inter pupillary distance (IPD) for adults is around 54-68mm [30]. The Leap cameras have a fixed inter camera distance (ICD) of 40mm, slightly closer than the average adult. We cannot place the visual cameras at a different ICD as the Leap cameras, as this causes alignment issues between the virtual and the real world objects.

All the cameras need to be placed as close to the centre of the Rift headset as possible; this is where the centre of the headset's displays and wearer's eyes are located. It was decided to place the Leap beneath the visual cameras as the wearer's hand will more often than not be tracked below the wearer's line of sight. Figure 3.3 shows the proposed optimal mounting positions. Note that the visual cameras are both turned 90° this is to match the ratio of the Rift screens. i.e. 960x1080.

FIGURE 3.3: Camera and Leap Mounting Positions

Figure 3.4 shows the actual field of view (FOV) of each of the components. The Leap Motion has a slightly larger FOV than the Rift, allowing for hand tracking outside of the wearer's vision. The visual camera pair however has a much smaller FOV than the Rift. To make sure that the entire FOV of the Rift is utilized a combination of IR and visual images were used, as will be shown later.

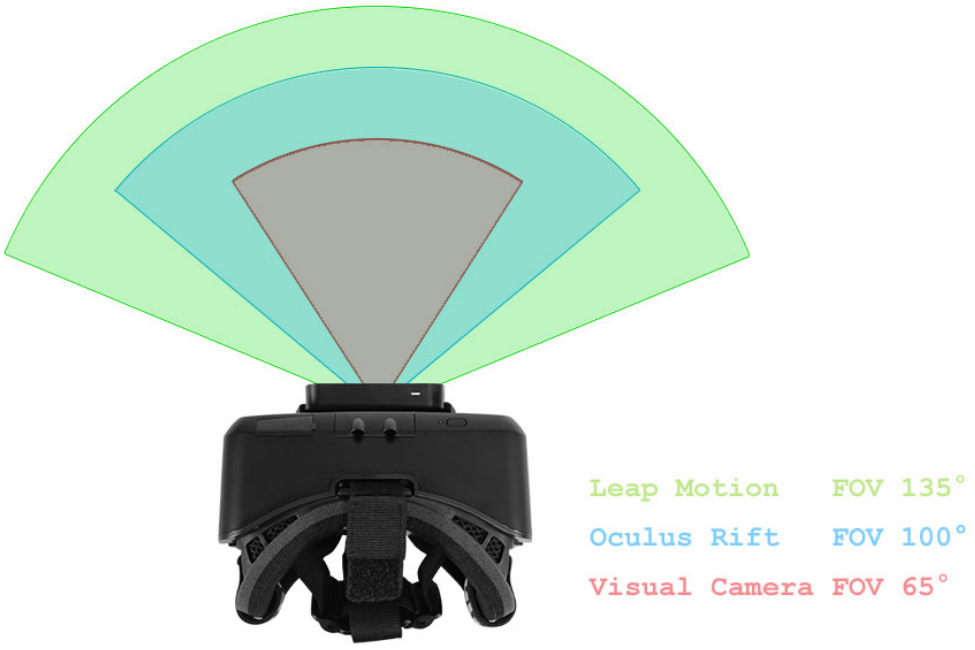
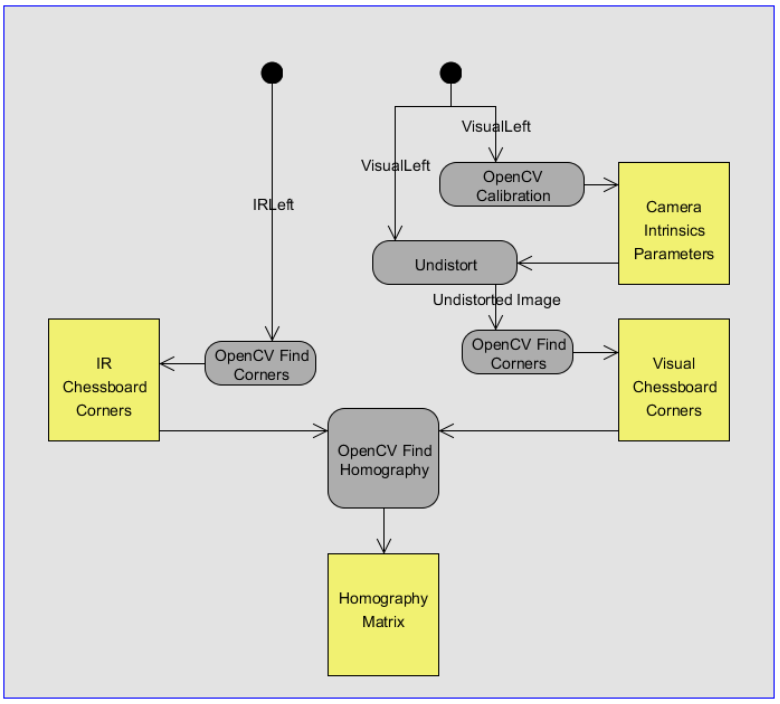


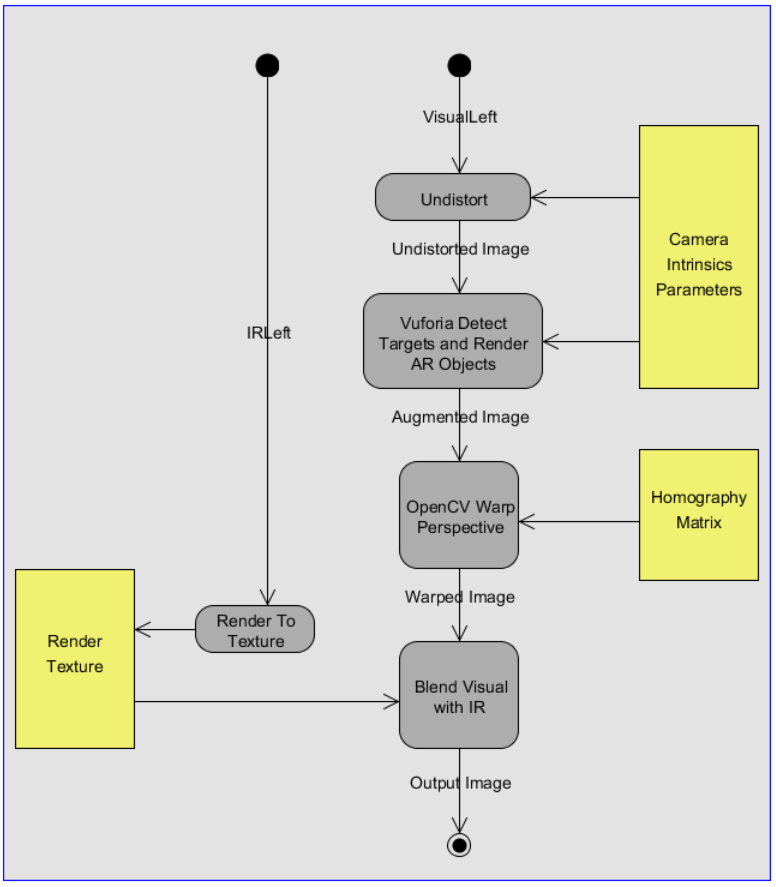
FIGURE 3.4: Field of View of the Leap, Rift and Visual Cameras

3.1.2 Software Implementation

The prototype was implemented using the Unity3D [31] game development tool and the Open Source Computer Vision (OpenCV) [32] library. The software consists of an off-line calibration step as shown in Figure 3.5a and the simulation loop as shown in Figure 3.5b. Note that calibration step and main loop is done twice, for both the left and right camera images. Each functional block is discussed in detail in the following sections.



(A) Calibration and Alignment Step



(B) Simulation Main Loop

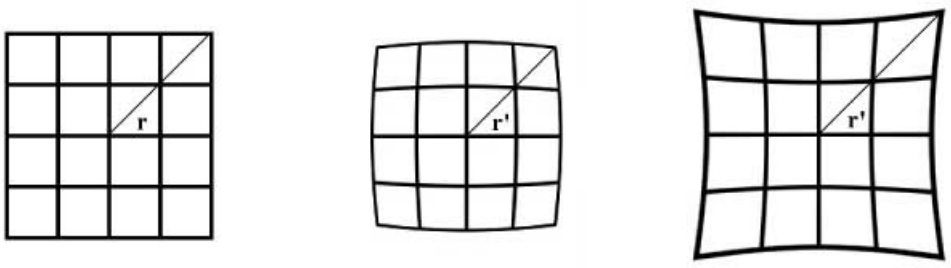
FIGURE 3.5: Flow Charts showing the Software Pipeline

3.1.2.1 Camera Calibration and Alignment

Because of mounting position offsets, as well as the different IR and visual lens characteristics, the IR image and visual images are completely misaligned. To ensure that the visuals we see match up with the hand tracking, two once-off calibration steps as well as a real-time perspective transformation step is needed.

Lens Distortion Compensation

The first calibration step is to correct for the lens distortion of the visual cameras. Although distortion can be irregular or follow many patterns, the most commonly encountered distortions are radially symmetric because of the symmetry of the lens. Figure 3.6 shows how negative (pincushion) and positive (barrel) radial distortion would deform a grid pattern [33].



(A) No Distortion (B) Barrel Distortion (C) Pincushion Distortion

FIGURE 3.6: Radial Distortion [34]

From the OpenCV documentation the radial distortion coefficients are calculated as follow [32]:

$$x_d = x_u(1 + k_1 r^2 + k_2 r^4)$$

$$y_d = y_u(1 + k_1 r^2 + k_2 r^4)$$

Where x_d and y_d are the distorted x and y coordinates and x_u and y_u are the undistorted x and y coordinates. k_1 and k_2 are the radial distortion symmetric parameters, used to measure the degree of radial distortion in an image.

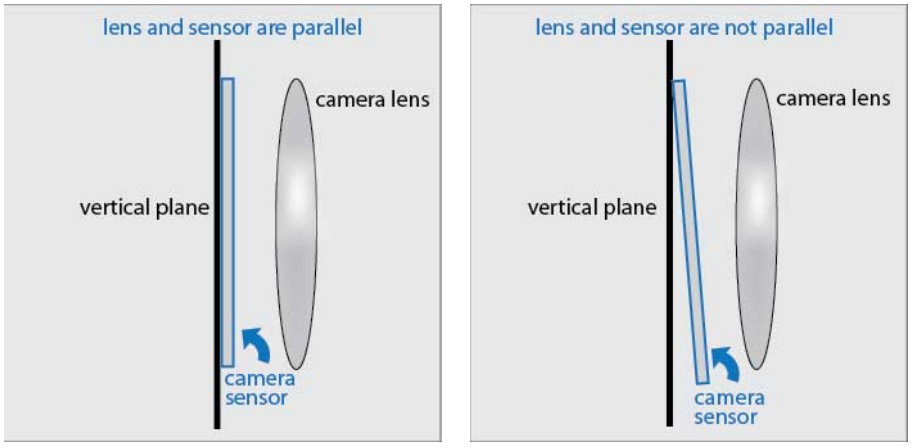
Tangential distortion [33] occurs when the lens and the sensor are not parallel. Tangential distortion coefficients are calculated by OpenCV[32] as follow:

$$x_d = x_u + [2p_1x_uy_u + p_2(r^2 + 2{x_u}^2)]$$

$$y_d = y_u + [p_1(r^2 + 2{y_u}^2) + 2p_2x_uy_u]$$

Where x_d and y_d are the distorted x and y coordinates and x_u and y_u are the undistorted x and y coordinates. p_1 and p_2 are the tangential distortion parameters.

Figure 3.7 shows the sensor and lens alignment leading to tangential distortion.



(A) No Distortion (B) Tangential Distortion

FIGURE 3.7: Tangential Distortion [35]

Using OpenCV, we calculate the *opticalCenter*, *focalLength* and radial and tangential factors of each lens, namely (k_1, k_2, p_1, p_2).

OpenCV calibration only has to be performed once per camera. A custom Unity3D Cg shader was written to correct for distortion in real time on the GPU. The shader fragment is shown in Listing 3.1, the code was derived from the OpenCV equations.

```
fixed4 frag(v2f_img i) : COLOR
{
    // UVc -> Optical Center in UV coords
    // UVf -> Focal Length in UV coords
    // UVu -> Undistorted pixel in UV coords
    float2 UVc = opticalCenter / imageSize;
    float2 UVf = focalLength / imageSize;
    float2 UVu = (i.uv - UVc) / UVfl;
```

```
    float r2 = dot(UVu, UVu);
    float r4 = r2 * r2;

    // Radial Distortion Coefficient
    float radialC = K1 * r2 + K2 * r4;

    // Tangential Distortion Coefficient
    float dx = P1*2.0*UVu.x*UVu.y + P2*(r2 + 2.0*UVu.x*UVu.x);
    float dy = P1*(r2 + 2.0*UVu.x*UVu.x) + P2*2.0*UVu.x*UVu.y;
    float2 tangentialC = float2(dx, dy);

    // UVd -> Distorted pixel in UV coords
    float2 UVd = ((UVu + UVu.xy*radialC + tangentialC)*UVf) + UVc;

    return tex2D(_MainTex, UVd);
}
```

LISTING 3.1: Lens Distortion Compensation

Figure 3.8 shows the image retrieved from the visual camera with clear barrel distortion due to the wide lens. Next to it is the shader corrected image.

(A) Original Distorted Image (B) Corrected Image

FIGURE 3.8: Lens Distortion Compensation

The Leap Motion IR camera images are already corrected for distortion and can be used as is.

Perspective Transform Between Cameras

The next calibration step is also only performed once. Using the distortion corrected images

from the IR and visual cameras we want to calculate the perspective transform needed to align the visual image with the IR image.

Homography estimation is used to find a transformation matrix between two planes. Figure 3.9 shows the homography matrix H that transforms a point in one view of a 3D geometry into the same point from a different view.

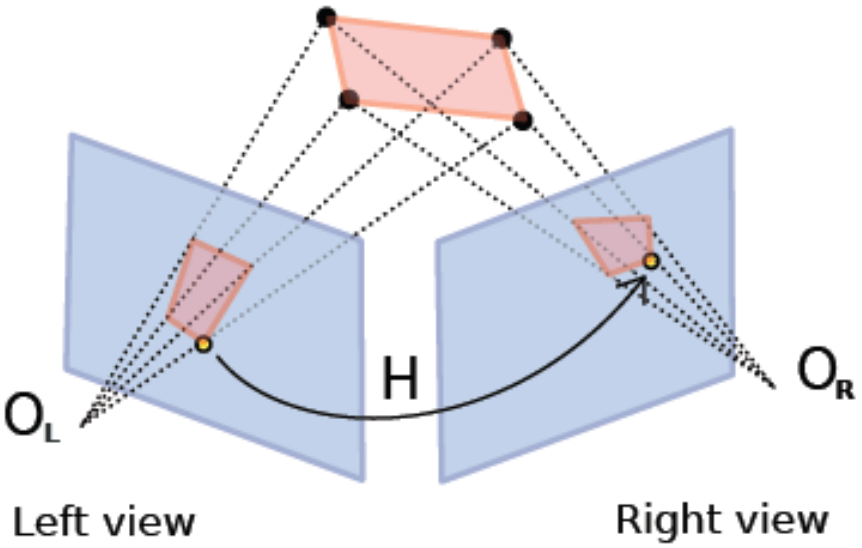


FIGURE 3.9: Perspective Transform using the Homography Matrix [36]

Once again OpenCV is utilized to detect a standard chessboard in each image and estimate the homography matrix to transform between the planes. OpenCV uses the Direct Linear Transform (DLT) algorithm described by Dubrofsky [37] to calculate a starting homography given the corresponding points found in the chessboard. Because the chessboard points are inexact, there will be some uncertainty. The problem then becomes to solve for a H that minimizes a suitable cost function. OpenCV iteratively minimizes H using a geometric [37] cost function.

The relationship between source (x_i, y_i) and destination (x_i', y_i'), given homography H

$$s_i \begin{bmatrix} x_i' \\ x_i \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

so that the cost function

$$\sum_i \left(x'_i - \frac{h_1 1 x_i + h_1 2 y_i + h_1 3}{h_3 1 x_i + h_3 2 y_i + h_3 3} \right)^2 + \left(y'_i - \frac{h_2 1 x_i + h_2 2 y_i + h_2 3}{h_3 1 x_i + h_3 2 y_i + h_3 3} \right)^2$$

is minimized.

Listing 3.2 shows the custom Unity3D C# code snippet that calculates the perspective transform using $findHomography$. The code assumes that the chessboard corners are populated for *cornersLeap* and *cornersRGB*.

```
void calibrate()
{
    CvMat mat1 = Cv.CreateMat(9*6, 2, MatrixType.F32C1);
    CvMat mat2 = Cv.CreateMat(9*6, 2, MatrixType.F32C1);

    for (int i=0; i<9*6; i++)
    {
        CvPoint2D32f p = cornersLeap[i];
        CvPoint2D32f p2 = cornersRGB[i];

        mat1[i, 0] = p.X;
        mat1[i, 1] = p.Y;

        mat2[i, 0] = p2.X;
        mat2[i, 1] = p2.Y;
    }

    CvMat H = new CvMat(3, 3, MatrixType.F32C1);
    Cv.FindHomography(mat1, mat2, H, HomographyMethod.Ransac);

    if (HInv == null)
    {
        HInv = new CvMat(3, 3, MatrixType.F32C1);
    }
```

```
    Cv.Inv(H, HInv, InvertMethod.LU);
    Debug.Log(HInvLeft);
}
```

LISTING 3.2: Perspective Transform Calculation

Figure 3.10 shows the Leap IR image, visual image and finally the transformed image to align the visual with the IR. Note that the images are upside down and that the red and blue channels of the colour image is swapped. This is because of image format difference between Unity3D and OpenCV, we do not do any format conversions until we convert back to Unity3D for the sake of performance. The final warped image's empty buffer is purposefully filled with a magenta colour, this is so that the visual and IR images can be blended using a shader. This will allow us to extend the small FOV provided by the visual image, by overlaying it on-top of the IR image.

(A) IR Image (B) Visual Image (C) Perspective Transform

FIGURE 3.10: Perspective Transform Between Cameras

Camera Alignment and Blending

Once the homography has been calculated, we can transform the visual image to be aligned with the IR image, in realtime. OpenCV's *warpPerspective* is used to align the images each frame.

WarpPerspective transforms the image using the following equation and the previously calculated homography matrix.

$$dst(x,y) = src(\frac{M_{11}x+M_{12}y+M_{13}}{M_{31}x+M_{32}y+M_{33}}, \frac{M_{21}x+M_{22}y+M_{23}}{M_{31}x+M_{32}y+M_{33}})$$

src is the original visual image after distortion correction, *dst* is the output image after a perspective transformation. In this case the transformation matrix is the inverse of the homography matrix, previously calculated $M = H^{-1}$ using M.

A custom Unity3D Cg shader was written to not render any magenta pixels, producing the aligned and blended image as shown in Figure 3.11. Note that the visual image is aligned with the IR image and that the position of the hand matches in both, meaning that where the user's hand is seen is also where the hand is being tracked by the Leap Motion.

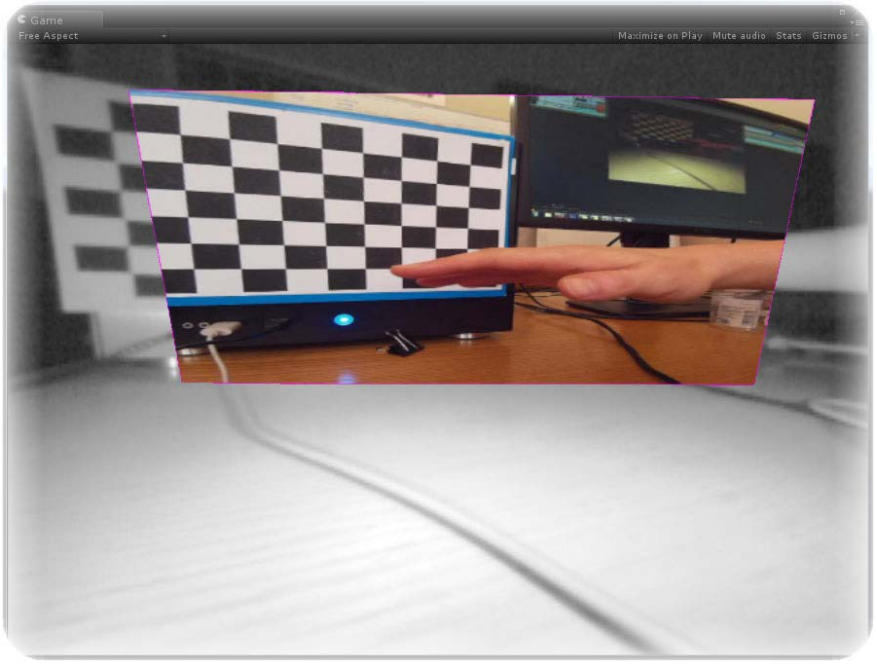


FIGURE 3.11: Physical Camera Alignment and Blending

3.1.2.2 Virtual Camera Placement

In addition to the physical cameras we also need virtual cameras, rendering the virtual objects for *Mixed Reality*. The virtual cameras are created within Unity3D and render the virtual screens, any virtual menus or controls as well as any augmented reality objects after the video from the physical cameras has been rendered.

As the virtual objects are rendered in stereo a stereo pair of virtual cameras are used. Unity3D allows this by simply setting the stereo separation property of the camera. It was found through experimentation, that it is better to match the separation with the inter camera distance (ICD) of the Leap Motion than to match the inter pupillary distance (IPD) of the headset user. This is because the latter would require scaling of the camera images for the interaction with virtual objects to align. Figure 3.12 shows the difference between the pupillary and camera distances [38].

FIGURE 3.12: Inter Camera and Inter Pupillary Distance [38]

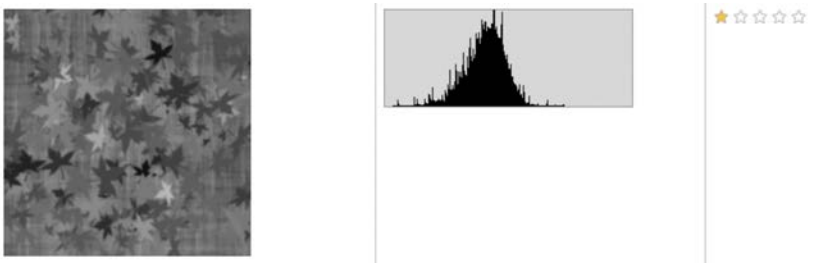
3.1.2.3 Augmented Reality Targets

Additionally to the virtual screens the Qualcomm Vuforia [39] SDK was used to recognize and track AR image targets. The Vuforia platform uses computer vision-based image recognition to allow augmentation of real world environments. Furthermore, Vuforia's computer vision library has been optimized to run on mobile devices and is easily integrated with Unity3D through a Unity plugin.

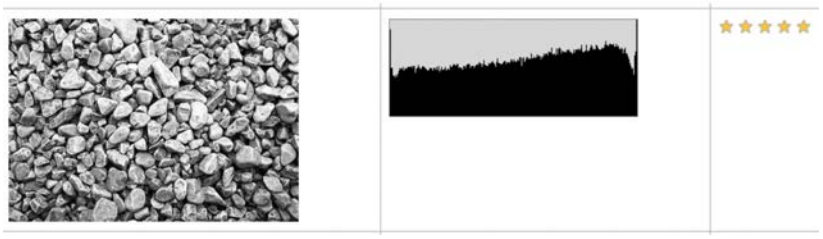
Vuforia can recognize various types of targets in a scene. For the purposes of this prototype, image based targets were chosen, as this allows any texture, added to the Vuforia database, to be

recognised. Unlike traditional markers, data matrix codes and QR codes, image targets do not need special black and white regions or codes to be recognized.

A good indicator to estimate target quality is to look at it as the grayscale representation. If the image has little overall contrast and the histogram of the image is narrow and spiky, there will likely not be good local contrast in the image and not many good features will be found. However, if the histogram is wide and flat, it is a good first indication that the image contains enough distinct areas with good features, meaning Vuforia can easily recognize the target. Figure 3.13 demonstrates two image targets from Vuforia [39] with bad and good feature ratings.



(A) Image Target with Bad Features



(B) Image Target with Good Features

FIGURE 3.13: Example AR Image Targets From [39]

Figure 3.14 shows the virtual cameras rendering a flower where a *Augmented Reality* marker is detected.

3.1.2.4 Virtual Screen and Instrument Placement

Finally the Rift's head tracking is used to place virtual objects relative to the virtual cameras. For example, a virtual screen in front of the user will always be in front of the initial position of the virtual cameras. This is achieved by attaching the virtual objects to the transform of the head tracking node in Unity3D. As the head transform is updated, all the child objects are also update. Figure 3.15 shows the camera in the center of the unity scene. The white plane directly

FIGURE 3.14: Virtual Camera Alignment

in front of the camera is a texture to which a camera feed is rendered. Two interactive menus. one containing text, and another buttons are placed at angles behind the camera.

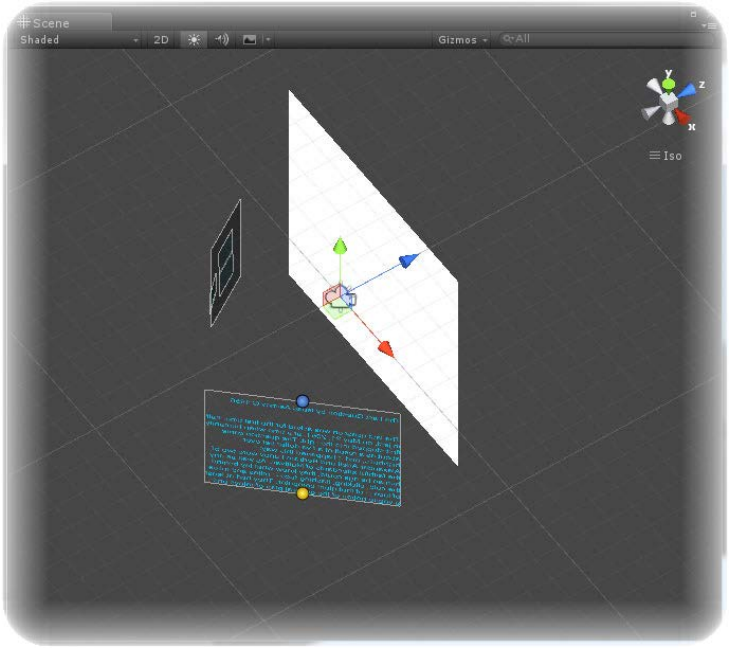


FIGURE 3.15: Virtual Screens Relative to Camera

The user is required to *reset* the initial virtual camera orientation as soon as he is facing what he considers to be the front view of the simulator. Figure 3.16 shows some example configurations of the prototype. The first example is a fully opaque view of a simulator cockpit, still allowing

use of a physical keyboard. The second example shows a semi-transparent dial that can be interacted with.

(A) Example of a Simulator Cockpit

(B) Example of a Interactive Menu

FIGURE 3.16: Virtual Screen Configurations Examples

3.2 Method 2: Stencil Cutouts

3.2.1 Stencils Instead of Virtual Screens

During implementation of the first prototype, it became evident that immersion is lost due to the lack of depth in the virtual screens. One way to fix the issue is to do the reverse and render the 3D world with stencils cutting out where we want the real world to show through. Thus instead of virtual screens we define stencils where the real world is shown.

The original and the revised concept is shown in Figure 3.17.

(A) Virtual Screens

(B) Stencil Cutouts

FIGURE 3.17: The Concept of Virtual Screens vs. Stencil Cutouts

Stencil cutouts define where the real world is visible. In Figure 3.17b stencil cutouts allow the user to still use the keyboard and mouse as well as get a peripheral view of his surroundings, as demonstrated by the stencil cutout to the right of the user's view.

Similar to the virtual screens, these stencils cutouts can be placed anywhere relative to the user with adjustable transparency. The advantage of using stencil cutouts is that the depth information is not lost by rendering a 3D world to 2D textures. This approach however maintains the benefits of being able to interact with physical controllers. It was decided to create new prototypes using stencil cutouts.

Hardware Components		
GearVR	Resolution Refresh Rate Field of View	1280 × 1440 per eye 60 Hz 96°
Samsung Galaxy S7	Resolution	2160 × 3840 @30fps 1080 × 1920 @60fps 720 × 1280 @240fps

TABLE 3.2: Specifications for Mobile Hardware Components

3.2.2 A More Portable Solution

During implementation the GearVR [40] headset from Oculus became available. The GearVR works with certain models of the Samsung Galaxy and Note range of smartphones. As this device is driven by the smartphone's processor, there is no need for tethering to a computer, making the headset more portable and less restrictive than the Oculus Rift. Additionally the built in phone camera can be used for mixed and augmented reality purposes, removing the need for an external camera attachment. Unfortunately the Leap Motion Controller is not compatible with the Android Operating System which runs on the Samsung Galaxy and Note smartphones. A bluetooth controller was incorporated for user inputs instead. Figure 3.18 shows the hardware components.

FIGURE 3.18: Mobile Hardware Components - A GearVR, Samsung Galaxy S7 and Bluetooth Controller From Left to Right

Table 3.2 shows the specification for the Gakaxy S7 and GearVR.

3.2.3 Implementing Stencil Cutouts

Shaders were used in Unity3D to modify the stencil buffer. For example, in Figure 3.19, a square cutout modified the stencil buffer to have 1's where the keyboard is visible in the background

video. Another shader, attached to the background video, only renders the pixels where the value in the stencil buffer is equal to 1.

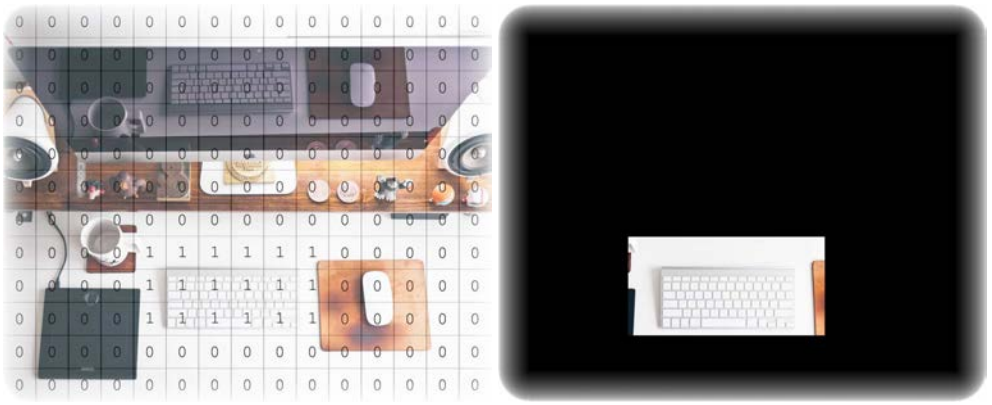


FIGURE 3.19: Stencil Cutout using the Stencil Buffer Applied to a Video Background

Listing 3.3 shows the *CutoutSetOne* shader, which writes a 1 into the stencil buffer.

```
Shader "Custom/Stencil/CutoutSetOne" {
SubShader {
    Tags { "RenderType"="Opague" "Queue"="overlay+9"
   "ForceNoShadowCasting" = "True"}
    ColorMask 0

    Stencil {
        Ref 1
        Comp always
        Pass Replace
    }

    Pass {
        ZTest Always
        ZWrite On
        Cull Off
        Lighting Off

        // vertex and pixel shaders
    }
    }
}
```

LISTING 3.3: Shader that Sets Stencil Buffer to 1

Listing 3.4 shows the *VideoBackgroundEqOne* shader that only renders the video background pixels where the stencil buffer equals 1

```
Shader "Custom/Stencil/VideoBackgroundEqOne" {
Properties {
    _MainTex ("Base (RGB)", 2D) = "white" {}
}
SubShader {
    Tags {"Queue"="overlay+10" "RenderType"="Opague" }
    Pass {
        ZTest Always
        ZWrite On
        Cull Off
        Lighting Off

    Stencil {
        Ref 1
        Comp Equal
    }

    SetTexture [_MainTex] {combine texture}
    }
    }
    FallBack "Diffuse"
}
```

LISTING 3.4: Shader that Renders when the Stencil Buffer Equals 1

Stencil cutouts shaders can be attached to any shape or geometry and can be placed relative to the user. For example a square or spherical stencil could be placed such that when the user looks down onto the table the mouse and keyboard are visible.

A more interesting application, is to have stencil cutouts that closely match the shape of a physical object, be tracked in real time. Figure 3.20 demonstrates a controller with an *Augmented Reality* marker being tracked and displayed in Unity3D.

The marker is tracked using the phone camera and the Vuforia Unity3D plugin. A square stencil, just big enough to reveal the controller, is attached to the marker's tracked position. Whenever the controller is "seen" by the camera, it becomes visible to the user. The user sees the controller

FIGURE 3.20: Controller Stencil Cutout using a Tracked Augmented Reality Marker

exactly where it would appear to him in the real word, without the headset on. The controller is not only visible to the user, but also appears where it would in the real world relative to the user, allowing for easier interaction.

Chapter 4

Evaluation and Usability

4.1 Test prototypes

In order to evaluate the stencil cutouts, driving and flying simulator prototypes were created, using some of the standard assets in Unity3D.

4.2 Objectively Evaluating the User Input

In order to objectively determine if the *Mixed Reality* user inputs are beneficial to the experience, a Quick Time Event (QTE) evaluation system was implemented. The QTE system requires the user to repeat a sequence of input events and measures how long each input event takes and whether it was successful or not.

A quick time event refers to a visual of a random required input shown to a participant. Successfully triggering the input causes a next event to trigger. The success or failure as well as delay of each input is measured and logged. This is an easy to use method that objectively measures how well the participants can interact with the system.

4.2.1 QTE Setup

The QTE system shows a sequence of inputs the user needs to carry out. The time between input actions is measured and saved for evaluation. To simplify the process for participants with

no experience using controllers, only the four front facing buttons of the controller were used. Figure 4.1 shows the colour coded buttons.

FIGURE 4.1: Coloured Buttons for Quick Time Events.

When a QTE is triggered the simulator is paused and one of the 4 coloured buttons is displayed at random. The user must then press the same button as shown on the screen. A sequence of four buttons is expected. When the wrong input is given the QTE sequence fails. If the correct input is given the time is recorded. Figure 4.2 shows a QTE sequence prompting the user to press the pink button. The stencil can optionally be switched off forcing the user to memorize the colours or to take off the headset.

FIGURE 4.2: Quick Time Sequence.

Results were saved as XML. Listing 4.1 shows excerpts from some captured results. The $<$ $stencilstate$ $>$ saves whether the stencil was on or off. The $< float >$ saves the time taken for

each input in *seconds*, while a value of -1 indicates failure. Upon failure the QTE sequence stops.

```
<!-- A complete saved QTE sequence with stencil on -->
<?xml version="1.0" encoding="utf-16"?>
<QTEData>
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>4.5178957</float>
    <float>3.51704335</float>
    <float>1.21341062</float>
    <float>1.3854059</float>
  </times>
</QTEData>

<!-- A failed QTE sequence with stencil off -->
<?xml version="1.0" encoding="utf-16"?>
<QTEData>
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>4.748553</float>
    <float>7.05744743</float>
    <float>-1</float>
  </times>
</QTEData>
```

LISTING 4.1: Saved QTE Results

4.2.2 Data Collection

Fourteen participants were chosen at random to take part in the objective evaluation test. A GearVR headset, with a Samsung Galaxy S7 running the driving simulator prototype was used. The driving simulator allows the user to drive around using the bluetooth controller as input. Each participant was instructed to look at the controller in their hands. Once the controller became visible via the stencil cutout, they were told which controller inputs to use to drive around in the world. Once they were comfortable, they were asked to trigger the QTE sequence using one of the trigger buttons of the controller. After completing the QTE sequence they were

Participant	Gender	Age Range	AR Experience	VR Experience
1	Male	30-35	No	Yes
2	Male	20-25	No	Yes
3	Female	30-35	No	No
4	Male	40-45	No	No
5	Male	25-30	No	No
6	Male	35-40	No	Yes
7	Female	30-35	No	No
8	Male	25-30	No	Yes
9	Male	25-30	No	No
10	Male	40-45	No	No
11	Male	25-30	No	Yes
12	Male	25-30	Yes	Yes
13	Male	35-40	No	Yes
14	Male	35-40	No	Yes

TABLE 4.1: Usability Study Participant List

asked to toggle the stencil mask off and perform the QTE sequence once again. QTE sequences were only performed once prevent participants from memorising the buttons.

Table 4.1 shows some information about the participants that took part.

4.2.3 QTE Results

Figure 4.3 shows the average time taken to provide the correct input for a QTE per participant. The column on the left shows the results while the participants were able to see the controller, the column on the right were the results while the stencil was turned off. Failed results indicate that the participant triggered the wrong input.

4.3 Subjective Usability Questionnaire

To evaluate the usability of the system and to determine if the *Mixed Reality* stencil cutouts have any effect on simulator sickness, a questionnaire was created.

4.3.1 Questionnaire Setup

The driving simulator prototype used for the QTE evaluation, as well as a second flight simulator prototype, was used for the questionnaire. The flight simulator was setup to fly on auto pilot with

Stencil On	Stencil Off
1.63	4.59
0.86	2.85
1.60	fail
fail	fail
1.48	fail
2.62	2.15
1.95	fail
2.23	2.66
1.43	fail
4.70	fail
fail	fail
2.61	fail
1.42	3.65
3.04	fail

FIGURE 4.3: Quick Time Sequence Results.

the controller triggers used to switch the stencils on or off. For this simulator the entire live feed from the Samsung Galaxy S7 camera was displayed. Figure 4.4 shows the flight simulator with the video feed displayed.

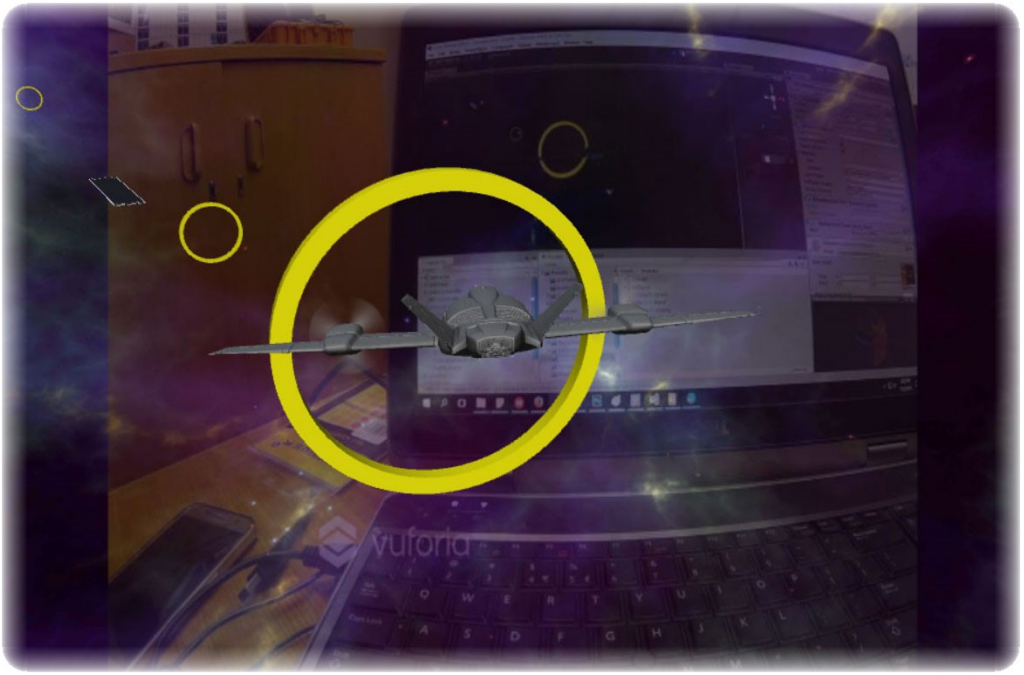


FIGURE 4.4: Flight Simulator with Video Background.

To not make the video stand out and detract from the virtual world too much, the video stencil was made semi transparent to blend with the skybox. The render order was also sorted in such

a way that the foreground simulator objects like the aircraft and obstacles render in front of the video background.

4.3.2 Data Collection

The same fourteen participants that took part in the objective evaluation test were asked to use the flight simulator prototype. While the video background was turned on, the participants were asked to perform tasks such as pick up their coffee mug or the telephone. Participants were then asked to fill in the questionnaire shown in Figure 4.5 pertaining to using both simulators. **Simulator 1** refers to the manually controlled driving simulator. **Simulator 2** refers to the auto pilot controlled flight simulator.

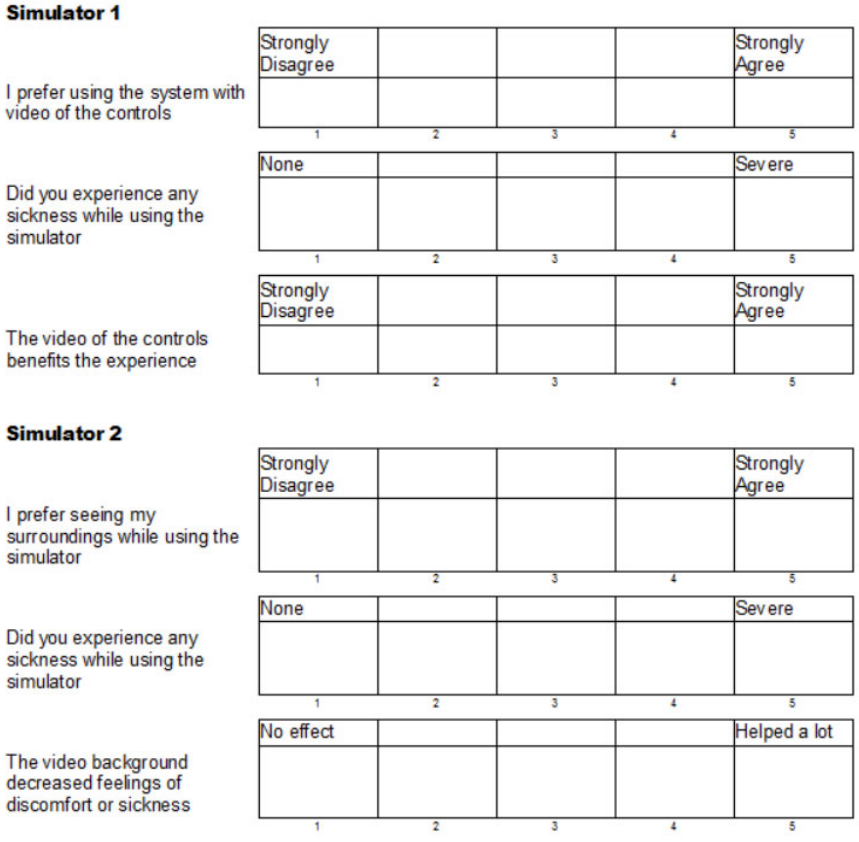


FIGURE 4.5: Subjective Usability Questionnaire.

4.3.3 Questionnaire Results

Repeated questions were combined and an average value was calculated for each question. The average scores are shown in Figure 4.6.

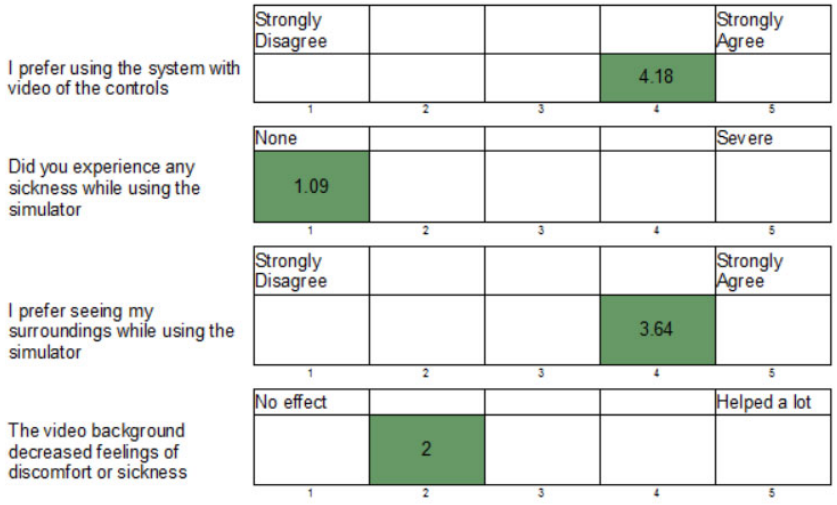


FIGURE 4.6: Average Usability Questionnaire Scores.

4.4 Findings Regarding Immersion

The GearVR headset allows for a 360° stereo view of a virtual environment. The headset tracks the user's head, allowing them to look around freely in the 3D virtual world. Participants were able to take part in a driving and flight simulator with complete immersion.

At the same time users were able to toggle video see-through on and off. This allowed participants to see and interact with physical objects such as phones or coffee mugs. From the questionnaire results we see that on average users have a 3.64 strong preference for being able to see their surroundings. Participants strongly felt that video see-through added to the experience.

One draw back of the video-see through implementation on the mobile hardware, is that only a single camera was used. The lack of stereo meant that participants lacked the depth perception to easily pick up and interact with objects. Participants did however get accustomed to this quite quickly.

4.5 Findings Regarding Usability

The first goal of the *Mixed Reality* prototypes was to allow the user to see and interact with their hands, allowing the use of any physical input devices. A prototype using an *Augmented Reality* marker on a controller was created. This allowed a user to see the real world controller while using the simulator. The controller was rendered exactly where it appeared relative to the user. Participants were given a Quick Time Event(QTE) based objective usability test.

The objective QTE results show an overwhelming advantage when using the stencil cutouts. Out of fourteen participants, eleven performed better while being able to see the controller. When considering more complex user controls, like the console in a drill rig or the flight yokes in an aircraft, the benefits would be even greater.

Participants were also asked to complete a questionnaire about their experience. A second prototype took control away from the user, showing a larger view of the real world, blended with the background of the simulator. This allowed users to still interact with their physical environment, such as phones, while the simulator was running.

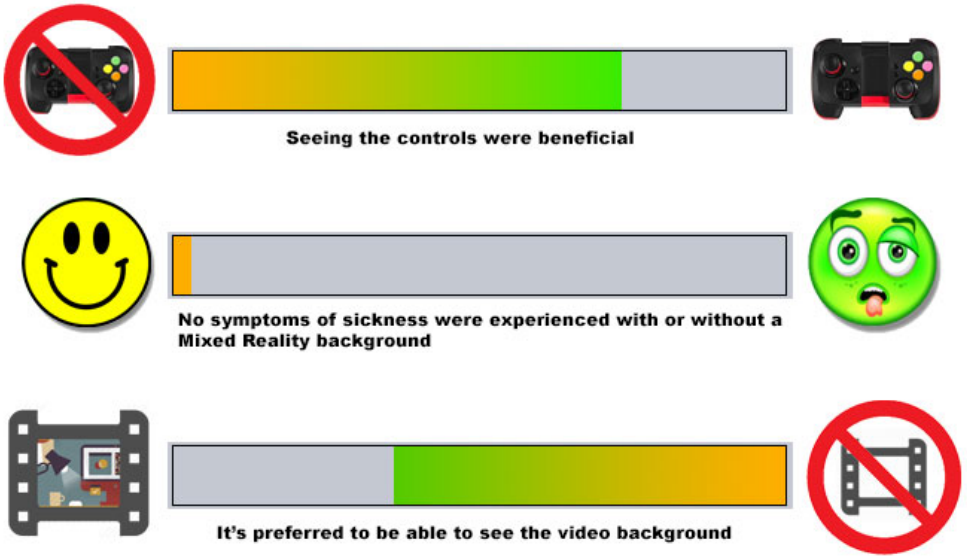


FIGURE 4.7: Subjective Usability Questionnaire Scale.

The results from the subjective usability questionnaire is represented on a scale in Figure 4.7. Most participants preferred being able to see the stencil cutout in the first prototype. Many participants commented that they would prefer to be able to toggle the background stencil on

and off as needed in the second prototype. All participants were able to interact with real world objects, such as coffee mugs, while using the simulator.

4.6 Findings Regarding Comfort and Simulator Sickness

The simulators prototypes are both fixed base simulators in that participants did not physically move. The driving simulator allowed participants to steer the vehicle. The flight simulator could not be controlled by the participant and flew by itself on auto pilot. As simulator sickness in fixed base systems are said to be caused by discrepancies between visual cues of motion and perceived motion, the flight simulator was meant to exasperate this discrepancy, with motion control taken away from the participant.

The hypothesis was that the mixed reality or video see-through nature of the simulator would negate or alleviate simulator sickness symptoms. This is because the user still has a reference point to the real world, matching his perceived motion.

In the entire set of participants only a single participant experienced mild symptoms related to simulator sickness. The participant felt ”unstable” while reversing the car in the simulator. The participant felt that the video background aided him, as he could focus on it while the virtual scene was moving. The participant did not experience any further symptoms after the video background was turned on.

Even though the lack of people experiencing simulator sickness, meant that the tests were inconclusive, the participant group strongly felt that the video see-through aided the experience.

Chapter 5

Conclusion

5.1 Summary of Contribution

The *Mixed Reality* prototypes presented in this research presents a novel use of augmented reality targets to mix in physical controls with *Virtual Reality* simulators. While existing hardware such as a OVRVision[10] stereo camera attachment, for the Oculus Rift[41] or a built in camera on for example a HTC Vive[42] does provide a live video feed to traditional *Virtual Reality* headsets, these systems are only currently used for *Augmented Reality* applications or to serve as proximity alerts for VR applications. The *Mixed Reality* built for this research shows the benefits of using these live feeds within VR applications, thus only cutouts, or blended views of the live feed is used as opposed to a full view of the live view in an AR application.

The *Mixed Reality* prototypes developed were not possible on Optical-See-Through (OST) displays such as the Microsoft HoloLens[43]. While see through displays can be used for *Mixed Reality* applications, they are currently not suited for immersive VR simulators, this is due to two factors, the narrow field of view (FOV) of the displays, and the inherent see through nature of the projected glass displays, not allowing for a completely opaque view of the virtual content.

5.2 Summary of Methodology

This aim of this research was to mix reality with *Virtual Reality* for simulator applications. This is different from merely using see-through *Augmented Reality* displays in that the virtual

world still needs to be an immersive 360° view. The limited field of view and transparency of see-through displays are not suited for these applications.

The approach was to use *Virtual Reality* headsets, combining them with cameras to create a *Mixed Reality* experience. Two different hardware approaches were tested, each with it's own pros and cons. Figure 5.1 shows a comparison of the two approaches.

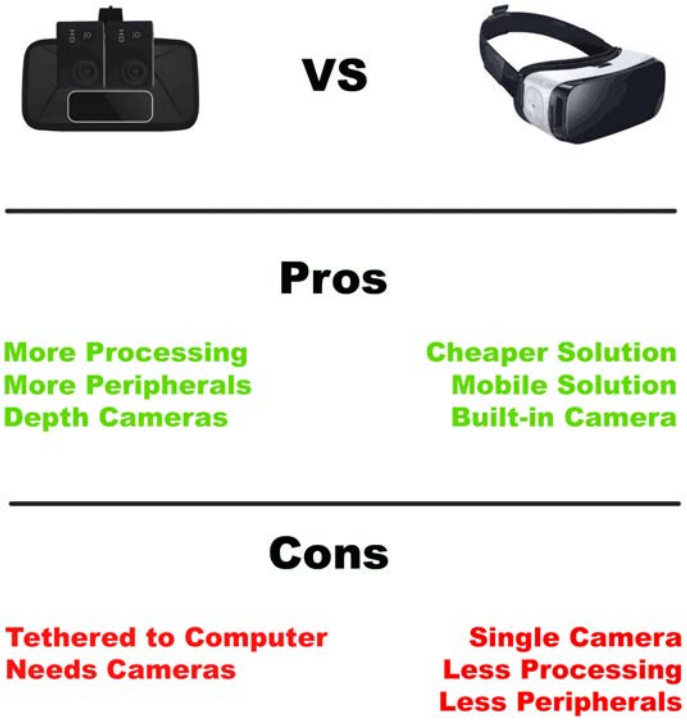


FIGURE 5.1: Comparison of the Two Hardware Approaches.

The first approach used an Oculus Rift headset combined with two cameras to add depth to the real world views. This approach allowed for the use of any computer peripherals such as the Leap Motion controller. The attached cameras and Leap Motion, lead to the system being bulky with a lot of additional wires leading to the computer. To be able to perform the usability tests it was decided to use a untethered system instead. The GearVR headset along with a mobile phone was used instead. This approach allowed for a mobile system that can be easily carried around and requires no setup. The mobile system is however limited in terms of available peripherals and processing power.

The initial software prototype used a view of the real world with virtual screens placed where desired. The virtual screens would contain views of the virtual simulator. This was implemented

as rendered textures, that rendered the correct view of the world relative to the user. The use of textures however meant that the depth information of the virtual scene was lost. To fix this the reverse approach was followed; the virtual world always surrounds the user with stencil cutouts letting the real world through where desired.

The mobile hardware along with two prototypes were given to fourteen participants for usability tests.

5.3 Conclusion of Usability Test Findings

Participants were required to perform QTEs which objectively measured how long it took to trigger required inputs. Around 80% of participants performed better while being able to see a video cutout of the tracked controller.

Participants were also asked to fill in a questionnaire regarding the usability of the system. Participants overwhelmingly preferred being able to toggle video see-through of the real world on or off. Participants were able to see and interact with physical objects, while using the simulator.

It is expected that video see-through negates or mitigates simulator sickness symptoms, due to the fact that users maintains a real world reference to match up with their perceived motion. The tests are however inconclusive, as only one participant experienced very mild symptoms of simulator sickness. Toggling on the video see-through did however alleviate the participant's symptoms.

The *Mixed Reality* simulators has the following observed benefits over traditional Virtual Reality simulators:

Users are not visually cut off from the real environment. Users are aware of any real obstacles or people around them increasing the safety of the simulator.

Users can interact with complex physical controls. While simple input devices can be used by touch alone, more complicated levers, switches and dials as would be found in heavy mining machinery would not be usable without seeing it. Users are now able to see their own hands and interact with physical controls such as steering wheels and flight yokes. Physical controls also provide tactile feedback over virtual controls. From the questionnaire and QTE results users prefer using and seeing physical controls, and perform better while doing so.

Mixed reality adds a real world reference to aid with motion sickness. Users now have a real world reference with which to consolidate signals from the body's proprioceptive system. While only a single participants experienced symptoms of simulator sickness, which disappeared with the addition of a real world video feed, most users indicated in the questionnaire that video cues reduces the discrepancy between the visual cues of position and movement and the perceived position and movement.

5.4 Recommendations and Future Work

From the usability tests it is evident that there is a lot of benefit in being able to see controllers and input devices. A next step would be to automatically classify and track input devices using for example a deep neural network, with automatic stencil fitting to only reveal the tracked object.

More tests are needed with regards to simulator sickness. A next step would be a prototypes designed with intentionally bad VR practices to try and induce more symptoms. The effect of the video background and whether it alleviates any of these symptoms can then be better studied.

Appendix A

Publications

The conference paper Stencil Cutouts for Virtual Reality Inputs, was accepted for peer reviewed publication in the International Conference Proceedings Series by ACM, (ISBN: 978-1-4503-4791-4), which will be archived in the ACM Digital Library, and indexed by Ei Compendex and Scopus and submitted to be reviewed by Thomson Reuters Conference Proceedings Citation Index (ISI Web of Science).

Stencil Cutouts for Virtual Reality Inputs

Natalie Ausmeier
Council for Industrial and Scientific Research
Pretoria
South Africa
+ 27 12 841 2911
nausmeier@csir.co.za

Turgay Celik
University of the Witwatersrand
Braamfontein
South Africa
+27 (0)11 717 1000
turgay.celik@wits.ac.za

ABSTRACT
Virtual Reality (VR) is widely used in training simulators of dangerous or expensive vehicles such as aircraft or heavy mining machinery. The vehicles often have very complicated controls that users need to master before attempting to operate a real world version of the machine. VR allows users to safely train in a simulated environment without the risk of injury or damaging expensive equipment in the field. VR however visually cuts off the user from the real environment, which may obtain obstructions. Users are unable to safely move or gesture while wearing a VR headset. Additionally, users are unable to use standard input devices such as mice and keyboards. By using *stencils* to *cutout* sections of the virtual world and insert a live video feed of the real world the user can still see and interact with the physical environment.

CCS Concepts
• Software and its engineering→Software creation and management • Software verification and validation→ Software prototyping.

Keywords
Mixed Reality; Virtual Reality; Augmented Reality; Training Simulators; Simulator Sickness.

1. INTRODUCTION
While VR headsets adds an immersive 360 ° view there are two major detractors that hinder the experience. The first is that while immersion is drastically increased the headset leaves the wearer completely cut off from the real environment. The user is unable to use any complex input devices such as keyboards or flight yokes, this is particularly important for training simulators of machinery with custom instruments and controls. Furthermore, this leaves the user unable to see any obstacles that occur in the real world and not in the virtual world, which may prove harmful. The second detractor is that users of closed off headsets may become disorientated or experience motion sickness. This is very rare experienced by users of multi-monitor configurations.

Conference'10, Month 1–2, 2010, City, State, Country.

DOI: http://dx.doi.org/10.1145/12345.67890

By using stencils, it is possible to "cut holes" into the view of the virtual world, to let glimpses of the real world come through. This creates a *Mixed Reality* [1][4] experience and how much or little of the real world can be controlled by the shape, size and transparency of the stencils.

Figure 1 demonstrates the concept, with stencil cutouts defining where the real world is visible. In this example stencil masks allow the user to still use the keyboard and mouse, revealed by the circular stencil mask as the user gazes downwards. A second stencil cutout allows a peripheral view of the user's surroundings, as demonstrated by the stencil cutout to the right of the user's view.

Figure 1. Glimpses of the real world using stencil cutouts.

2. BACKGROUND AND MOTIVATION
Looking at current technology and research, the two main focus areas are, being able to see and use physical controls or objects, and overcoming simulator sickness.

2.1 Physical Controls
While current Optical-See-Through (OST) displays allows users to stay aware of their surroundings, OSTs are not suited for immersive *Virtual Reality* experiences. This is mainly due to the semi-transparent projected glass displays of current OSTs. They are inherently see-through and more suited for *Augmented Reality* (AR) type applications. The horizontal and vertical field of view of available headsets are also much lower than that of the typical human eye, meaning that when a virtual worlds or large virtual objects are displayed the objects are cut off in the users' view. The goal for this research was to still be surrounded by a virtual scene, but to be able to look down and see your hands, take a sip of coffee or use physical controls.

To this end it was determined that a VR headset would be better suited, making use of a camera feed to implement the stencil cutouts.

2.2 Simulator Sickness

While there is no definitive research on the causes of visually induced motion sickness, it is clear from the research that *Simulator Sickness* is much more prevalent in a closed off, fully immersive system, such as a VR headset with non-see-through displays. Bos, et al. [2] presents the subjective vertical mismatch theory as a possible explanation of *Simulator Sickness*. The theory states that the mismatch between the senses, provided by artificial visual cues in the case of a closed off headset, and the subjective vertical expected by the user from previous experience, results in sickness.

Howarth and Costello [3] and later Sharples, et al. [5] compared the sickness symptoms experienced when comparing VR headsets, desktop and projection displays.

Participants reported a significantly greater number of increases in general discomfort, fatigue, headache, nausea, dizziness and stomach awareness using the headsets compared to alternative displays.

By using stencil cutouts, we have the flexibility to tweak the size, number and transparency of the stencils. This allows users who are sensitive to *Simulator Sickness* to configure the system to see more real world visual cues to match their subjective vertical.

3. IMPLEMENTATION

Ohta and Tamura [4] provide an in-depth look at the current state of *Mixed Reality*. One of the main uses of *Mixed Reality* is combining live and virtual performance art, as explored by Benford and Giannachi [1]. It has however not been used to combine real world inputs with virtual simulations. Stencil cutouts of the real world were used to implement *Mixed Reality* prototypes.

3.1 Hardware

For the VR headset the GearVR from Oculus, along with the Samsung Galaxy S7 smartphones, was used. A mobile solution was chosen as to avoid tethering to a computer as would have been necessary with for instance the Oculus Rift. Additionally, the built in phone camera could be used for the stencil cutouts, removing the need for an external camera attachment. Figure 2 shows the hardware components.

Figure 2. Hardware components.

3.2 Software

All software was developed using the Unity3D game development tool, allowing fast prototyping and deployment to Android.

3.2.1 Test Prototypes

In order to evaluate the stencil cutouts, driving and flying simulator prototypes were created, using some of the standard assets in Unity3D.

3.2.2 Implementing Stencil Masks

Shaders were used in Unity3D to modify the stencil buffer. For example, in Figure 3, a square cutout modified the stencil buffer to have 1's where the keyboard is visible in the background video. Another shader, attached to the background video, only renders the pixels where the value in the stencil buffer is equal to 1.

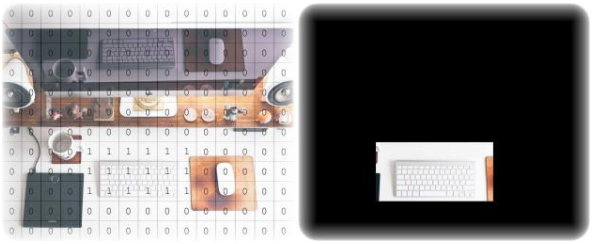


Figure 3. The stencil buffer.

The following Unity3D shaderlab snippet, shows the *CutoutSetOne* shader, which writes a 1 into the stencil buffer.

```
Shader "Custom/Stencil/CutoutSetOne" {
SubShader {
    Tags { "RenderType"="Opague" "Queue"="overlay+9"
                    "ForceNoShadowCasting" = "True"}
    ColorMask 0

    Stencil {
        Ref 1
        Comp always
        Pass Replace
    }

    Pass {
        ZTest Always
        ZWrite On
        Cull Off
        Lighting Off

        // vertex and pixel shaders
    }
    }
}
```

The *VideoBackgroundEqOne* shader snippet, only renders the video background pixels where the stencil buffer equals 1.

```
Shader "Custom/Stencil/VideoBackgroundEqOne" {
Properties {
    _MainTex ("Base (RGB)", 2D) = "white" {}
}
SubShader {
    Tags {"Queue"="overlay+10" "RenderType"="Opague" }
    Pass {
        ZTest Always
        ZWrite On
        Cull Off
        Lighting Off

    Stencil {
        Ref 1
        Comp Equal
    }

    SetTexture [_MainTex] {combine texture}
    }
    }
    FallBack "Diffuse"
}
```

Stencil cutouts shaders can be attached to any shape or geometry and can be placed relative to the user. For example, a square or spherical stencil could be placed such that when the user looks down onto the table the mouse and keyboard are visible. A more interesting application, is to have stencil cutouts that closely match the shape of a physical object, be tracked in real time. Figure 4 demonstrates a controller with an *Augmented Reality* marker being tracked and displayed in Unity3D. The marker is

tracked using the phone camera and the Vuforia Unity3D plugin. A square stencil, just big enough to reveal the controller, is attached to the marker's tracked position. Whenever the controller is "seen"

Figure 4. Tracked controller stencil cutout.

by the camera, it becomes visible to the user. The user sees the controller exactly where it would appear to him in the real word, without the headset on.

The controller is not only visible to the user, but also appears where it would in the real world relative to the user, allowing for easier interaction.

4. EVALUATION AND USABILITY

4.1 Objectively Evaluating the User Input

In order to objectively determine if the *Mixed Reality* user inputs are beneficial to the experience, a Quick Time Event (QTE) evaluation system was implemented. The QTE system requires the user to repeat a sequence of input events and measures how long each input event takes and whether it was successful or not.

4.1.1 QTE Setup

The QTE system shows a sequence of inputs the user needs to carry out. The time between input actions is measured and saved for evaluation. To simplify the process for participants with no experience using controllers, only the four front facing buttons of the controller were used. Figure 5 shows the color coded buttons.

Figure 5. Colored buttons for quick time events.

When a QTE is triggered the simulator is paused and one of the 4 colored buttons is displayed at random. The user must then press the same button as shown on the screen. A sequence of four buttons is expected. When the wrong input is given the QTE sequence fails. If the correct input is given the time is recorded. Figure 6 shows a QTE sequence prompting the user to press the pink button. The stencil can optionally be switched off forcing the user to memorize the colors or to take off the headset.

Figure 6. A Quick time event.

Results were saved as XML. Excerpts from some captured results are shown in the following snippet.

```
<!— A complete saved QTE sequence with stencil on —>
<?xml version="1.0" encoding="utf-16"?>
<QTEData>
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>4.5178957</float>
    <float>3.51704335</float>
    <float>1.21341062</float>
    <float>1.3854059</float>
  </times>
</QTEData>

<!— A failed QTE sequence with stencil off —>
<?xml version="1.0" encoding="utf-16"?>
<QTEData>
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>4.748553</float>
    <float>7.05744743</float>
    <float>-1</float>
  </times>
</QTEData>
```

The < *stencilstate* > saves whether the stencil was on or off. The < *float* > saves the time taken for each input in *seconds*, while a value of −1 indicates failure. Upon failure the QTE sequence stops.

4.1.2 Data Collection

Fifteen participants were chosen at random to take part in the objective evaluation test. A GearVR headset, with a Samsung Galaxy S7 running the driving simulator prototype was used. The driving simulator allows the user to drive around using the bluetooth controller as input. Each participant was instructed to look at the controller in their hands. Once the controller became visible via the stencil cutout, they were told which controller inputs to use to drive around in the world. Once they were comfortable, they were asked to trigger the QTE sequence using one of the trigger buttons of the controller. After completing the QTE sequence they were asked to toggle the stencil mask off and perform the QTE sequence once again. QTE sequences were only performed once prevent participants from memorizing the buttons.

4.1.3 QTE Results

Figure 7 shows the average time taken to provide the correct input for a QTE per participant. The column on the left shows the results while the participants were able to see the controller, the column on the right were the results while the stencil was turned off. Failed results indicate that the participant triggered the wrong input.

Stencil On	Stencil Off
1.63	4.59
0.86	2.85
1.60	fail
fail	fail
1.48	fail
2.62	2.15
1.95	fail
2.23	2.66
1.43	fail
4.70	fail
fail	fail
2.61	fail
1.42	3.65
3.04	fail

Figure 7. Quick time event results.

4.2 Subjective Usability Questionnaire

To evaluate the usability of the system and to determine if the *Mixed Reality* stencil cutouts have any effect on simulator sickness, a questionnaire was created.

4.2.1 Questionnaire Setup

The driving simulator prototype used for the QTE evaluation, as well as a second flight simulator prototype, was used for the questionnaire. The flight simulator was setup to fly on auto pilot with the controller triggers used to switch the stencils on or off. For this simulator the entire live feed from the Samsung Galaxy S7 camera was displayed. Figure 8 shows the flight simulator with the video feed displayed.

Figure 8. Flight simulator with video background.

To not make the video stand out and detract from the virtual world too much, the video stencil was made semi transparent to blend with the skybox. The render order was also sorted in such a way that the foreground simulator objects like the aircraft and obstacles render in front of the video background.

4.2.2 Data Collection

The same fifteen participants that took part in the objective evaluation test were asked to use the flight simulator prototype. While the video background was turned on, the participants were asked to perform tasks such as pick up their coffee mug or the telephone. Participants were then asked to fill in the questionnaire shown in Figure 9 pertaining to using both simulators.

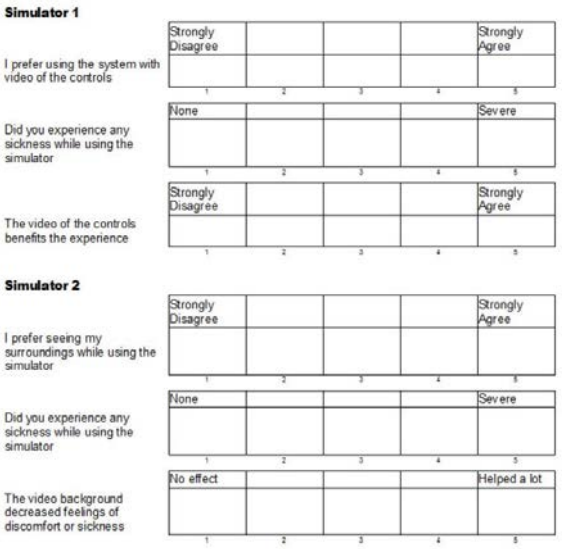


Figure 9. Usability questionnaire.

4.2.3 Questionnaire Results

Repeated questions were combined and an average value was calculated for each question. The average scores are shown in Figure 10.

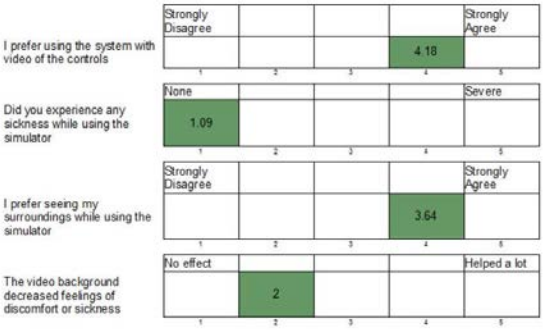


Figure 10. Usability questionnaire results.

5. CONCLUSION

5.1 Stencil Cutouts for Simulator Sickness

In the entire set of participants only a single participant experienced symptoms related to simulator sickness. The participant felt "unstable" while reversing the car in the simulator. The participant felt that the video background aided him, as it gave him a real world reference point while the virtual scene was moving. The participant did not experience any further symptoms after the video background was turned on. More tests are needed with prototypes de- signed with intentionally bad VR practices.

5.2 Stencil Cutouts for User Inputs

The objective QTE results show an overwhelming advantage when using the stencil cutouts. When considering more complex user controls, like the console in a drill rig or the flight yokes in an aircraft, the benefits would be even greater.

The results from the subjective usability questionnaire is represented on a scale in Figure 11. Most participants preferred being able to see the stencil cutout of the controller. Many participants commented that they would prefer to be able to toggle the background stencil on and off as needed. All participants were able to interact with real world objects, such as coffee mugs, while using the simulator.

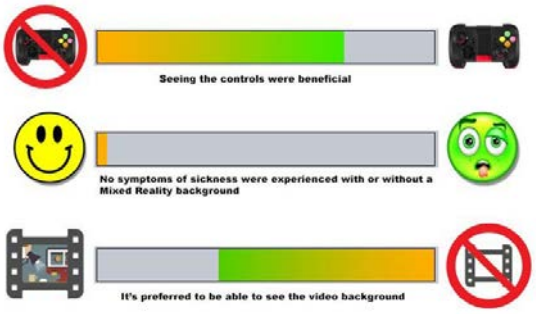


Figure 11. Usability scale.

A next step would be to automatically classify and track input devices using for example a deep neural network, with automatic stencil fitting to only reveal the tracked object.

6. REFERENCES

[1] S. Benford and G. Giannachi. *Performing mixed reality*. The MIT Press, 2011.

[2] J. E. Bos, W. Bles, and E. L. Groen. A theory on visually induced motion sickness. *Displays*, 29(2):47–57, 2008.

[3] P. Howarth and P. Costello. The occurrence of virtual simulation sickness symptoms when an hmd was used as a personal viewing system. *Displays*, 18(2):107–116, 1997.

[4] Y. Ohta and H. Tamura. *Mixed reality: Merging real and virtual worlds*. Springer Publishing Company, Incorporated, 2014.

[5] S. Sharples, S. Cobb, A. Moody, and J. R. Wilson. Virtual reality induced symptoms and effects (vrise): Comparison of head mounted display (hmd), desktop and projection display systems. *Displays*, 29(2):58–69, 2008.

Appendix B

Usability Questionnaire Results

To evaluate the usability and comfort of the mixed reality prototype and to ascertain if any simulator sickness symptoms occurred and if the use of mixed reality had any effect on the symptoms, participants were asked to fill in a usability questionnaire.

Bernardo

Simulator 1

I prefer using the system with video of the controls

Strongly Disagree				Strongly Agree
		X		
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
✓				
1	2	3	4	5

The video of the controls benefits the experience

Strongly Disagree				Strongly Agree
			X	
1	2	3	4	5

Simulator 2

I prefer seeing my surroundings while using the simulator

Strongly Disagree				Strongly Agree
		X		
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
✓				
1	2	3	4	5

The video background decreased feelings of discomfort or sickness

No effect				Helped a lot
1	2	3	4	5

Gladys Sonko

Simulator 1

I prefer using the system with video of the controls

Strongly Disagree				Strongly Agree
1	2	3	4	5

while learning then option to turn off

Did you experience any sickness while using the simulator

None				Severe
X				
1	2	3	4	5

The video of the controls benefits the experience

Strongly Disagree				Strongly Agree
				X
1	2	3	4	5

Simulator 2

I prefer seeing my surroundings while using the simulator

Strongly Disagree				Strongly Agree
X				
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
X				
1	2	3	4	5

The video background decreased feelings of discomfort or sickness

No effect				Helped a lot
1	2	3	4	5

V/A - did not get sick.

Ipiling

Simulator 1

I prefer using the system with video of the controls

Strongly Disagree				Strongly Agree
		X		
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
	X			
1	2	3	4	5

The video of the controls benefits the experience

Strongly Disagree				Strongly Agree
			X	
1	2	3	4	5

Simulator 2

I prefer seeing my surroundings while using the simulator

Strongly Disagree				Strongly Agree
			X	
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
		X		
1	2	3	4	5

The video background decreased feelings of discomfort or sickness

No effect				Helped a lot
				X
1	2	3	4	5

Thy's

Simulator 1

I prefer using the system with video of the controls

Strongly Disagree				Strongly Agree
	✓			
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
✓				
1	2	3	4	5

The video of the controls benefits the experience

Strongly Disagree				Strongly Agree
		✓		
1	2	3	4	5

Simulator 2

I prefer seeing my surroundings while using the simulator

Strongly Disagree				Strongly Agree
		✓		
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
✓				
1	2	3	4	5

The video background decreased feelings of discomfort or sickness

No effect				Helped a lot
✓				
1	2	3	4	5

Meshack

Simulator 1

I prefer using the system with video of the controls

Strongly Disagree				Strongly Agree
X				
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
X				
1	2	3	4	5

The video of the controls benefits the experience

Strongly Disagree				Strongly Agree
				X
1	2	3	4	5

Simulator 2

I prefer seeing my surroundings while using the simulator

Strongly Disagree				Strongly Agree
			X	
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
X				
1	2	3	4	5

The video background decreased feelings of discomfort or sickness

No effect				Helped a lot
				X
1	2	3	4	5

Hannes

Simulator 1

I prefer using the system with video of the controls

Strongly Disagree				Strongly Agree
			X	
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
X				
1	2	3	4	5

The video of the controls benefits the experience

Strongly Disagree				Strongly Agree
			X	
1	2	3	4	5

Simulator 2

I prefer seeing my surroundings while using the simulator

Strongly Disagree				Strongly Agree
				X
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
X				
1	2	3	4	5

The video background decreased feelings of discomfort or sickness

No effect				Helped a lot
X				
1	2	3	4	5

Vusi

Simulator 1

I prefer using the system with video of the controls

Strongly Disagree				Strongly Agree
			X	
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
			X	
1	2	3	4	5

The video of the controls benefits the experience

Strongly Disagree				Strongly Agree
		X		
1	2	3	4	5

Simulator 2

I prefer seeing my surroundings while using the simulator

Strongly Disagree				Strongly Agree
			X	
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
X				
1	2	3	4	5

The video background decreased feelings of discomfort or sickness

No effect				Helped a lot
			X	
1	2	3	4	5

Terence

Simulator 1

I prefer using the system with video of the controls

Strongly Disagree				Strongly Agree
		✓		
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
✓				
1	2	3	4	5

The video of the controls benefits the experience

Strongly Disagree				Strongly Agree
		✓		
1	2	3	4	5

Simulator 2

I prefer seeing my surroundings while using the simulator

Strongly Disagree				Strongly Agree
			✓	
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
✓				
1	2	3	4	5

The video background decreased feelings of discomfort or sickness

No effect				Helped a lot
✓				
1	2	3	4	5

Dumisani

Simulator 1

I prefer using the system with video of the controls

Strongly Disagree				Strongly Agree
				✓
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
✓				
1	2	3	4	5

The video of the controls benefits the experience

Strongly Disagree				Strongly Agree
				✓
1	2	3	4	5

Simulator 2

I prefer seeing my surroundings while using the simulator

Strongly Disagree				Strongly Agree
			✓	
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
✓				
1	2	3	4	5

The video background decreased feelings of discomfort or sickness

No effect				Helped a lot
✓				
1	2	3	4	5

Jaco

Simulator 1

I prefer using the system with video of the controls

Strongly Disagree				Strongly Agree
			X	
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
X				
1	2	3	4	5

The video of the controls benefits the experience

Strongly Disagree				Strongly Agree
				X
1	2	3	4	5

Simulator 2

I prefer seeing my surroundings while using the simulator

Strongly Disagree				Strongly Agree
		X		
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
X				
1	2	3	4	5

The video background decreased feelings of discomfort or sickness

No effect				Helped a lot
X				
1	2	3	4	5

Kim

Simulator 1

I prefer using the system with video of the controls

Strongly Disagree				Strongly Agree
				5
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
1				
1	2	3	4	5

The video of the controls benefits the experience

Strongly Disagree				Strongly Agree
				5
1	2	3	4	5

Simulator 2

I prefer seeing my surroundings while using the simulator

Strongly Disagree				Strongly Agree
				5
1	2	3	4	5

Did you experience any sickness while using the simulator

None				Severe
1				
1	2	3	4	5

The video background decreased feelings of discomfort or sickness

No effect N/A				Helped a lot
1	2	3	4	5

Appendix C

Quick Time Event Results

In order to objectively determine if the mixed reality user inputs are beneficial to the experience, a Quick Time Event (QTE) evaluation system was implemented. The QTE system requires the user to repeat a sequence of input events and measures how long each input event takes and whether it was successful or not.

Results were saved as XML. The $< stencilstate >$ saves whether the stencil was on or off. The $< float >$ saves the time taken for each input in *seconds*, while a value of -1 indicates failure. Upon failure the QTE sequence stops.

```
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>3.29719234</float>
    <float>0.8855297</float>
    <float>0.852327347</float>
    <float>1.49564052</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>6.79645061</float>
    <float>9.509756</float>
    <float>0.8755429</float>
    <float>1.190115</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>4.82374048</float>
    <float>4.792015</float>
    <float>0.821015537</float>
    <float>0.950973868</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil On</stencilstate>
  <times>
```

```
    <float>1.85762715</float>
    <float>0.0357498452</float>
    <float>0.8590503</float>
    <float>0.7018404</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>3.582906</float>
    <float>0.9855834</float>
    <float>0.7381477</float>
    <float>1.10307944</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>-1</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>5.23294544</float>
    <float>1.9185766</float>
    <float>2.84624553</float>
    <float>-1</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
```

```
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>4.50448847</float>
    <float>-1</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>1.699936</float>
    <float>1.40198576</float>
    <float>1.23602128</float>
    <float>1.59813273</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>2.109095</float>
    <float>-1</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>3.71050858</float>
    <float>1.60395753</float>
    <float>4.212658</float>
    <float>0.9395827</float>
  </times>
```

```
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>4.12708664</float>
    <float>1.11004233</float>
    <float>1.22228634</float>
    <float>2.13724828</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>2.3101542</float>
    <float>1.79280686</float>
    <float>1.758761</float>
    <float>1.92397821</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>-1</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>3.05060768</float>
    <float>2.26305723</float>
```

```
    <float>2.1263907</float>
    <float>1.49213552</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>7.22927475</float>
    <float>1.23773849</float>
    <float>0.96962</float>
    <float>1.21174836</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>1.63634574</float>
    <float>2.48213959</float>
    <float>0.8423982</float>
    <float>0.7470754</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>2.82289124</float>
    <float>1.01611531</float>
    <float>8.374638</float>
    <float>-1</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
```

```
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>2.09832931</float>
    <float>13.8231163</float>
    <float>1.51688087</float>
    <float>1.37689674</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>2.16075635</float>
    <float>1.20026135</float>
    <float>0.931368053</float>
    <float>-1</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil Off</stencilstate>
  <times>
    <float>2.786484</float>
    <float>-1</float>
  </times>
</QTEData>
<?xml version="1.0" encoding="utf-16"?>
<QTEData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <stencilstate>Stencil On</stencilstate>
  <times>
    <float>2.22698069</float>
    <float>-1</float>
  </times>
```

```
</QTEData>
```

Bibliography

[1] Thales simulators. https://www.thalesgroup.com/en/worldwide/defence/training-simulation.

[2] Microsoft flight. http://www.microsoft.com/games/flight/.

[3] Clare Regan. An investigation into nausea and other side-effects of head-coupled immersive virtual reality. *Virtual Reality*, 1(1):17–31, 1995.

[4] Sarah Sharples, Sue Cobb, Amanda Moody, and John R Wilson. Virtual reality induced symptoms and effects (vrise): Comparison of head mounted display (hmd), desktop and projection display systems. *Displays*, 29(2):58–69, 2008.

[5] PA Howarth and PJ Costello. The occurrence of virtual simulation sickness symptoms when an hmd was used as a personal viewing system. *Displays*, 18(2):107–116, 1997.

[6] Epson moverio augmented reality. https://epson.com/moverio-augmented-reality.

[7] Yuichi Ohta and Hideyuki Tamura. *Mixed reality: merging real and virtual worlds*. Springer Publishing Company, Incorporated, 2014.

[8] Steve Benford and Gabriella Giannachi. *Performing mixed reality*. The MIT Press, 2011.

[9] Jannick P Rolland, Richard L Holloway, and Henry Fuchs. Comparison of optical and video see-through, head-mounted displays. In *Photonics for Industrial Applications*, pages 293–307. International Society for Optics and Photonics, 1995.

[10] Ovr vision. http://ovrvision.com.

[11] William Steptoe. Ar-rift tutoria. *Website tutorial, URL: http://willsteptoe.com/tagged/Oculus-rift*, 2015.

[12] Lightwave. https://www.lightwave3d.com/.

[13] Cinema lives cinema loves. https://miriamruthross.wordpress.com/terminology/.

[14] Cullen Jennings. Robust finger tracking with multiple cameras. In *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999. Proceedings. International Workshop on*, pages 152–160. IEEE, 1999.

[15] Shahzad Malik. Real-time hand tracking and finger tracking for interaction csc2503f project report. *Department of Computer Science, University of Toronto, Tech. Rep*, 2003.

[16] Peng Song, Hang Yu, and Stefan Winkler. Vision-based 3d finger interactions for mixed reality games with physics simulation. In *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, page 7. ACM, 2008.

[17] Christian Von Hardenberg and François Bérard. Bare-hand human-computer interaction. In *Proceedings of the 2001 workshop on Perceptive user interfaces*, pages 1–8. ACM, 2001.

[18] Jakub Segen and Senthil Kumar. Gesture vr: vision-based 3d hand interace for spatial interaction. In *Proceedings of the sixth ACM international conference on Multimedia*, pages 455–464. ACM, 1998.

[19] Ivan Tashev. Kinect development kit: A toolkit for gesture- and speech-based human-machine interaction. *Signal Processing Magazine*, September 2013. URL http://research.microsoft.com/apps/pubs/default.aspx?id=201229.

[20] Jagdish L Raheja, Ankit Chaudhary, and Kunal Singal. Tracking of fingertips and centers of palm using kinect. In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2011 Third International Conference on*, pages 248–252. IEEE, 2011.

[21] Primesense nite. http://openni.ru/files/nite/.

[22] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, volume 1, page 3, 2011.

[23] Leap Motion. Leap motion controller, 2014.

[24] E Stern, V Barth, W Durfee, M Rosen, T Rosenthal, E Schold-Davis, and J Zola. A protocol for avoiding driving simulator sickness. In *STISIM Users Conference, MIT AgeLab, Cambridge, MA*, 2006.

[25] Jelte E Bos, Willem Bles, and Eric L Groen. A theory on visually induced motion sickness. *Displays*, 29(2):47–57, 2008.

[26] JA Irwin. The pathology of sea-sickness. *The Lancet*, 118(3039):907–909, 1881.

[27] T Sharkey and M McCauley. Does a motion base prevent simulator sickness? In *Flight Simulation Technologies Conference*, page 4133, 1992.

[28] BS Cheung, IP Howard, and KE Money. Visually-induced sickness in normal and bilaterally labyrinthine-defective subjects. *Aviation, space, and environmental medicine*, 1991.

[29] VR Oculus. Oculus rift-virtual reality headset for 3d gaming. *URL: http://www. oculusvr. com*, 2012.

[30] Neil A Dodgson. Variation and extrema of human interpupillary distance. In *Electronic imaging 2004*, pages 36–46. International Society for Optics and Photonics, 2004.

[31] Unity. http://unity3d.com/.

[32] Open computer vision. http://opencv.com/.

[33] Juyang Weng, Paul Cohen, Marc Herniou, et al. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on pattern analysis and machine intelligence*, 14(10):965–980, 1992.

[34] Kai San Choi, Edmund Y. Lam, and Kenneth K. Y. Wong. Automatic source camera identification using the intrinsic lens radial distortion. *Opt. Express*, 14(24):11551–11565, Nov 2006. doi: 10.1364/OE.14.011551. URL http://www.opticsexpress.org/abstract.cfm?URI=oe-14-24-11551.

[35] Mathworks. http://www.mathworks.com/.

[36] Pierre Moulon, Pascal Monasse, Renaud Marlet, and Others. Openmvg. https://github.com/openMVG/openMVG.

[37] Elan Dubrofsky. *Homography estimation*. PhD thesis, UNIVERSITY OF BRITISH COLUMBIA (Vancouver, 2009.

[38] Leap motion. https://www.leapmotion.com/.

[39] Qualcomm vuforia. https://developer.vuforia.com/.

[40] Samsung gearvr. www.samsung.com/global/galaxy/gear-vr/.

[41] Oculus rift. http://www.oculusvr.com/.

[42] Htc vive. https://www.vive.com/.

[43] Microsoft hololens. https://www.microsoft.com/en-us/hololens/.