Using Virtual Reality- and Gesture Recognition Technology in Design Reviews

A Master's Thesis

Andreas Oven Aalsaunet

Thesis submitted for the degree of
Master in Programming and Networks
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2017

Using Virtual Reality- and Gesture Recognition Technology in Design Reviews

A Master's Thesis

Andreas Oven Aalsaunet

Using Virtual Reality- and Gesture Recognition Technology in Design Reviews

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Traditionally the idea of several people interacting in a virtual world, and the emerging virtual reality technologies (e.g Oculus Rift and HTC Vive etc) have been closely tied to the gaming- and entertainment industry. As of this date, these technological advances remains mostly irrelevant for most other industry segments, but could this be about to change? This essay will explore the possibilities of applying these technologies to the design and engineering segment of the industry.

Acknowledgements

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The field of virtual reality (VR) technology has seen an exciting development in recent years, with the release of the first commercial virtual reality headsets, such as Oculus Rift CV1 and HTC Vive, taking place in 2016.

The application area for these virtual reality headset have exceeded the expectations of many, with virtual reality technology being present in domains ranging from entertainment to educational training(Leadem, 2016). Leadem (2016) reports numerous domains where virtual reality is successful being used, including healthcare (e.g surgery), military, architecture/construction, art, fashion, entertainment (games, films etc), education, business, telecommunications, sports and rehabilitation.

Despite this early success, there are still a lot challenges associated with virtual reality technology. One of these challenges is related to human-computer interactions and will be expanded upon later in this chapter. This chapter will first discuss the virtual reality field and how gesture recognition technology can be very relevant for it, before defining the problem definition, limitations and outline for the rest of this thesis.

1.2 Virtual Technology

Virtual reality can be defined as a realistic and immersive simulation of a three-dimensional 360 degree environment, created using interactive software and hardware, and experienced or controlled by movement of the body (Leadem, 2016). This virtual environment is perceived through a virtual reality headset, which is a stereoscopic head-mounted display (HMD) that provide separate images for each eye (Kuchera, 2016). In addition to separate eye displays a HMD typically also contains head motion sensors such as gyroscopes, accelerometers and other sensors to track the user's head movements(Kelly, 2016). A person using a virtual reality head-mounted display should thus perceive a virtual world with realistic depth vision and be able to "look around" by turning his or her head.

Figure 1.1: The Oculus Rift Development Kit 1, released by Oculus VR in 2012.

The development of virtual reality head-mounted displays was in many ways fueled by the development of smart phones as many of the components are similar (e.g. gyroscopes), and these components also became more affordable by the prominence of smart phones. This led to the prototype HMD "Oculus Rift Development Kit 1", released by Oculus VR in 2012, being the first independently developed and sold virtual reality headset(Kelly, 2016).

As virtual reality technology enables users to experience virtual worlds in a new way, human-computer interaction (HCI) is also a highly relevant topic. This field has in many ways seen a resurgence as virtual technology gives new possibilities, but also set new constrains. One of these constraints is limiting the user's field of vision exclusively to that projected by the lenses, which may make interaction with traditional input devices, such as mouse and keyboard, more challenging. Because of this, alternate methods of interacting with the computer is a relevant topic. One of these methods is the use of gestures, which have long been considered an interaction technique that can potentially deliver more natural, creative and intuitive methods for communicating with our computers (Rautaray and Agrawal, 2015). To enable the use of gestures as a viable input method to a computer, responsive and reliable gesture recognition techniques are needed.

1.3 Problem definition

This thesis will evaluate the consequences of utilizing virtual reality technology in combination with vision based gesture recognition technology, and discuss the benefits it might bring, as well and the challenges it presents. The thesis will also review the design and implementation of a design review application, which is developed as part of this thesis with the aforementioned goal in mind. The design review application is also a prototype developed for the major international classification company DNV GL to evaluate how the use of virtual reality and gesture recognition technology might benefit their design review process. As such, the application requirements has been created in cooperation with DNV GL and represents common 3D object manipulating and navigation tasks. After discussing the design and implementation choices of this application, the user evaluation session will be discussed. This user evaluation sessions where performed by DNV GL employees, and potential end users, and contained invaluable feedback relevant to the use of virtual reality and gesture recognition technology in a professional setting.

1.4 Limitations

The initial list of application features had to be shortened significantly to focus more on the most relevant parts for this thesis. As such the design review application is more a prototype or proof-of-concept than a finished product. Section X outlines the application features and will explain more of whats include in the application and what isn't.

1.5 Outline

This thesis is organized as follows: In chapter 2 we will review the history and theoretical foundations for virtual reality and gesture recognition technology, as well as discuss some of their primary challenges. In chapter 3 we will review the design of the design review application, and how the applications defines and detect gestures. Chapter 4 will go more into the technical details of how the application is implemented and serve as a documentation for the source code. In chapter 5 the user trials will be covered, and the responses discussed and analyzed. Chapter 6 concludes the thesis with a quick summary, some thoughts about future work and a conclusion.

Chapter 2

Virtual Reality Technology

2.1 The virtual reality ecosystem

As described in the previous chapter, a virtual reality head-mounted device (HMD) is in simple terms a device that is fastened to the user's head and, when fastened, covers the user's entire field of vision. Each eye has its own display, and both of these are positioned about 2-3 centimeters from the eyes. In addition to this several head motion tracking sensors are built into the headset to detect any movement (Kuchera, 2016). This usually includes a gyroscope, which is responsible for measuring the orientation of the HMD, and some times an accelerometer to measure the proper acceleration of the HMD (Robertson, 2016). In addition, or instead of this, the first consumer versions of virtual reality headset also usually utilizes some other sensors or cameras outside the HMD. As an example the Oculus Rift CV1 utilizes constellation sensors Feltham (2015), which are usually positioned on a table, while the HTC Vive utilizes two "lighthouse stations", which are usually placed in opposite corners of the room, and uses photosensors and structured light lasers to obtain the users position and rotation Buckley (2015). It is worth noting that both of these virtual reality headset also is sold with their own controllers, which uses similar technology as the HMD, but as previously stated this thesis will investigate gesture recognition systems as the primary interaction method.

2.2 Virtual reality performance demands

Virtual reality places some strict demands on performance and software design to avoid discomfort for the user. This is in many ways connected to how virtual reality "tricks" the user's brain into thinking the virtual experiences are actually real, thus giving it its "reality feel". Failing to meet these demands can quickly result in significant discomfort for the user, often called ***virtual reality sickness***, a kind of motion sickness (Orland, 2013). Some of this demands are described below:

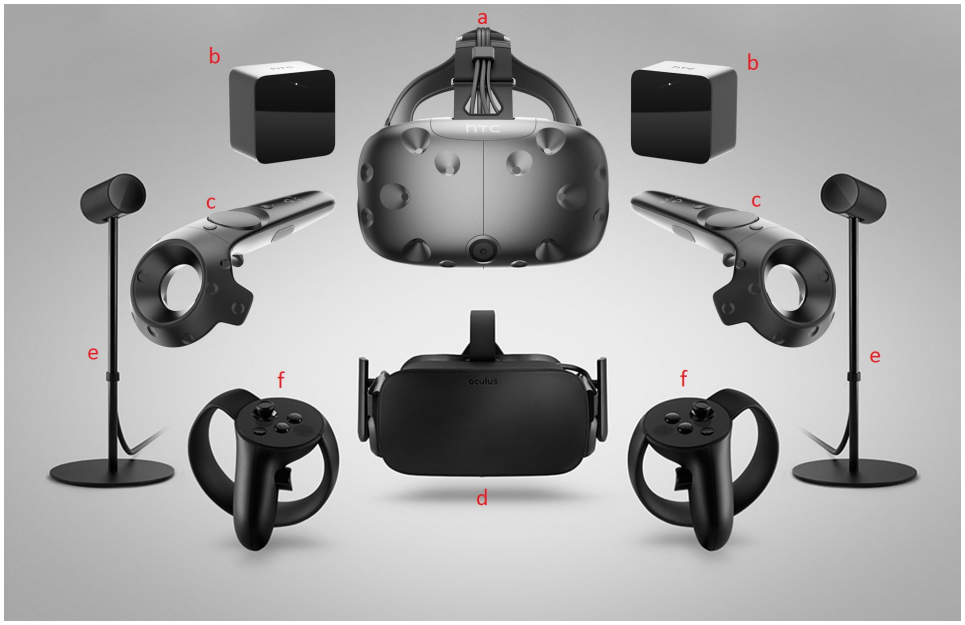

Figure 2.1: The HTC Vive and Oculus Rift Hardware. a) The HTC Vive headset (HMD). b) The HTC Vive Lighthouse Stations. c) The HTC Vive Controllers. d) The Oculus Rift headset (HMD). e) The Oculus Rift Constellation Sensors. f) The Oculus Rift Touch Controllers. Picture from Bye (2016)

2.2.1 Latency requirements

Virtual reality headsets have a much stricter requirements for latency, i.e the time required for an input to have a visible effect, than with use of regular displays (Lang, 2013). If this demand isn't met the system might feel "sluggish", and user's actions and visual feedback might feel disjoint, which often lead to virtual reality sickness. According to one of the engineers behind the HTC Vive, the ideal latency is between 7 and 15 milliseconds (Orland, 2013). One important component of this latency is the refresh rates of the displays, i.e how often the display hardware updates its buffers and thus "draws" a new image on the displays. As an example both the Oculus Rift CV1 and the HTC Vive has a refresh rate of 90 Hz (i.e the display updates 90 times per second), as opposed to the 60 Hz which is more common in commodity displays. In addition to refresh rate frame rate, i.e how often the graphics processing unit (GPU) renders frames/images, is also important. To ensure that the displays don't "redraws" and identical frame on a buffer update the frame rate should thus ideally be the same or higher than the refresh rate (e.g 90 frames per second for the Oculus Rift CV1 or HTC Vive). Refresh rate and frame rate are thus highly codependent, where performance is only as good as the weaker of the two, and the target computer should thus have a GPU strong enough to meet a frame rate equal or above to the HMD's refresh rate.

Asynchronous reprojection

To reduce the perceived latency, or to compensate for a frame rate that is too low, several virtual reality HMDs makes use of ***asynchronous reprojection*** (equivalent to what Oculus VR refer to as "asynchronous time warp") (S., 2016). This is a technique in which the virtual reality system generates intermediate frames in situations where the software (e.g a game) can't maintained the required frame rate (which is typically 90 fps with 90 Hz). In simple terms asynchronous reprojection produces "in-between frames", which is a manipulated version of an older rendered frame. This is done my morphing the frame according to the most recent head tracking data just before the frame is presented on the displays (S., 2016). By doing this, software that runs at e.g 45 FPS (frames per seconds) natively can be transformed into 90 FPS by applying asynchronous reprojection to each rendered frame. Every other frame is thus actually a manipulated version of the former frame.

2.2.2 Display resolution and quality

Virtual reality headsets also have strict demands in respect to display resolution and quality. As the eyes of the user is closer to the displays than with a regular monitor, and the displays have to "wrap around" the user's whole field of view, flaws and shortcomings in the display technology become more apparent. One such example is ***the screen-door effect (SDE)***, which is when the lines separating the display pixel or subpixels is visible in the displayed image (**?**). To illustrate this issue **?** had the following remark about the Oculus Rift DK1 (released in 2013 with a resolution of 640×800 per eye):

"Its low resolution screen (combined with magnification lenses that helped wrap the image around your view) made even the most beautifully rendered 3D environment look dated. It was like you were sitting too close to an old TV, or staring at the display through a screen door (aptly, this shortcoming quickly came to be known as "the screen door effect")"

2.3 Virtual reality sickness

As mentioned in the previous sections, virtual reality sickness, a condition similar to ***simulator sickness***, can be a consequence of the usage of virtual reality headsets, and is considered a major barrier to using virtual reality. Virtual reality sickness causes symptoms that are similar to those of motion sickness, and can include symptoms like headache, stomach awareness, nausea, vomiting, pallor, sweating, fatigue, drowsiness and disorientation (Kolasinski, 1995).

Contrary to "regular" motion sickness, where the user visually perceived to be still while in actual motion, virtual reality sickness turns this around: The user visually perceive to be in motion which he or she is still. Virtual reality sickness can thus in many ways be considered as "a reverse

Figure 2.2: An example of the screen-door effect.

motion sickness". Both these condition can thus be caused by ***sensory conflict***, i.e that there exist a discrepancy between the information given by the senses ("the human sensors"). The susceptibility for this condition vary widely among users. Some user might experience it shortly after putting on the headset, while others may never experience it Stanney et al. (2003). The causes for virtual reality sickness can vary and while some are less under the VR application designer's control than others, they should still be understood by the VR designer Stanney et al. (2003). The following two subsections will review factors that contribute to virtual reality sickness, and make a distinction by what are mostly determined by individual differences and whats mostly determined by the application design.

2.3.1 Individual differences in susceptibility

Research has identified some individual differences that correlate with susceptibility for virtual reality sickness. One observation is that the susceptibility of virtual reality sickness correlates heavily motion sickness susceptibility, and factors that influence motion sickness susceptibility also usually influence virtual reality sickness susceptibility (Stanney et al., 2003). Below are some of the major contributing factors that are based on individual differences, and are difficult to account for during the design of a virtual reality application.

Age

Research suggest that users between the ages of 2 and 12 are the most susceptible to virtual reality sickness (Kolasinski, 1995). The susceptibility then decreases rapidly until an age of about 21, before it start decreasing more slowly until and age of 50, where the susceptibility increases again (Brooks et al., 2010).

Gender

Women have proven more susceptible to virtual reality sickness than men (Kennedy, 1985). The most common theories to explain this difference point out the genders' differences in hormonal composition, field of view (women has a wider field of view than men) and differences in depth cue recognition (Kennedy, 1985). Women are most susceptible to virtual reality sickness during ovulation (Clemes and Howarth, 2005).

Ethnicity

Some ethnicities seem to be more susceptible to virtual reality sickness than others, suggesting a genetic component. Several studies prove asians people to be more susceptible to visually-induced motion sickness, with the Chinese being more susceptible that European-Americans and African-Americans on measures to motion sickness induced by a circular vection drum, and with Tibetans and Northeast Indians having greater susceptibility than Caucasian races Barrett (2004).

Health

Symptoms of virtual reality sickness are more prevalent in people who are fatigued, sleep deprived, are nauseated or have an upper respiratory illness, ear trouble or influenza Kolasinski (1995).

Postural stability

Users with a postural instability has been found to be more susceptible to visually-induced motion sickness, such as virtual reality sickness, and to experience stronger symptoms of nausea and disorientation Kolasinski (1995).

Experience with the application

More exposure to virtual environments can train the brain to be less sensitive to their effects (Stanney et al., 2003). Users tend to become less likely to experience virtual reality sickness as they become more familiar with the virtual reality application. This adaption may occur with only a few seconds of exposure to the application Kennedy (1985).

In addition to this, people with a low threshold for detecting flicker and low mental rotation ability are more susceptible to virtual reality sickness Kolasinski (1995).

2.3.2 Virtual reality design factors

This section identifies some of the most common contributers to virtual reality sickness that can be lessened or mitigated completely by the VR application design.

Acceleration

As mentioned earlier sensory conflict during a virtual reality session might occur. This is especially noticeable during acceleration that is conveyed visually, but not to the vestibular organs (inner ear organs that responds to acceleration). The speed of movement does not seem to contribute to virtual reality sickness in the same scale as the vestibular organs do not respond to constant velocity.

Camera control

Some theories indicates that the ability to anticipate and control the motion the user experiences plays a significant role in staving off motion- and virtual reality sickness (Rolnick and Lubow, 1991). Unexpected movement of the camera should thus be avoided in the virtual reality application. If the camera control is taken away from the user it is considered good practice to cue the impending camera movement to help the user to anticipate and prepare for the visual motion (Lin et al., 2004).

Field of view

The term "field of view" (FOV) can refer both to "display FOV" and "camera FOV", which are similar, but still distinct concepts that can both have an effect on the user's proneness to virtual reality sickness.

Display FOV refers to the area of the visual field subtended by the display. As motion perception is more sensitive in the periphery view a wide display FOV can contribute to VR sickness by providing the visual system with more visual input, i.e more "area" in the periphery, than a smaller display FOV. This can lead to more sensory conflict as more of the visual view suggest that the user is moving, which he or she might be standing or sitting still. Reducing display FOV can reduce the changes of VR sickness (Draper et al., 2001), but can also reduce the level of immersion and awareness, and require the user to turn his or her head more than with a higher display FOV.

Camera FOV refers the area of the virtual environment that the graphics engine draws to the display. If the camera FOV is setup wrong, movement of the user head can lead to unnatural movement in the virtual environment (e.g a 15° rotation of the head can lead to a 25° rotation of the camera in the virtual environment). In addition to begin highly discomforting, this can lead to a temporary impairment in the vestibulo-ocular reflex, which is a reflex to stabilize images on the retinas during head movement (Stanney, 2002).

Focus distance

To avoid discomfort and fatigue it is important to place content the user will be focusing on for extended amounts of time in an optimal range. As an example Oculus VR recommends such content to be placed a distance

in the range of 0.75 to 3.5 Unity units/meters away from the camera (Dean Beeler and Pedriana, 2016).

Latency and lag

As mentioned earlier in this chapter, latency and lag can have a major impact VR sickness and the usability of the virtual reality application as a whole. Although designers and developers have no control over many aspects of a system's performance, it's important to make sure the target virtual reality application doesn't drop frames or lag on a minimum technical specifications system (Dean Beeler and Pedriana, 2016). While some dropped frames or occasional jitter can be a minor annoyance in conventional applications or video games, it can have a much more discomforting effect on the user of a virtual reality application.

Some research indicates that a fixed, and thus predictable, latency creates about the same degree of VR sickness whether it's as short as 48 milliseconds or as long as 300 milliseconds, and that big and predictable latency or lag are more comfortable for VR users than smaller, but more unpredictable, latency or lag (Draper et al., 2001).

Mouse and keyboard usage

While a user is wearing a virtual reality headset, interaction with external input devices such as a keyboard, might be inconvenient or difficult. Put simply, this is because the user can't see his or her hands and thus can't get the visual hand positional feedback they could get without the virtual reality HMD. Because of this, many virtual reality applications makes use of a gamepad controller instead.

Chapter 3

Gesture Recognition Technology

3.1 Gesture recognition devices

Gesture recognition technology is a field that has gained much attention with the growth of the virtual reality field, and it's a very diverse one with roots in sensor technology, image processing and computer vision (Vafadar and Behrad, 2014). The first attempts at a commercial hand gesture recognition system were typically glove-based control interfaces, often called ***data gloves*** and were gloves with sensors attached to it. As the image processing and computer vision technology wasn't mature yet, these ***contact-based devices*** remained the primary gesture recognition technology, until the image processing-reliant ***vision-based devices*** began to see some success in the 2000s (Premaratne, 2014). Another factor which made data gloves ideal was a very limited requirement for processing power, as any pre-processing were rarely done, and thus the systems could run optimally on the commodity 1980s and 1990s computers (Premaratne, 2014).

Today, both contact-based and vision-based devices are utilized for gesture recognition purposes.

Contact-based devices are usually wearable objects, such as gloves or armbands, which register the user's kinetic movement through sensors and attempt to mirror it in the virtual world. Some notable products making use of this technology include the Nintendo Wii remote controller and the Myo armband (see figure 3.2).

Vision-based devices usually make use of either depth-aware cameras or stereo cameras to approximate a 3D representation of what's output by the cameras, which in many ways are similar to how the human eyes work. Products making use of this technology include the Microsoft's Kinect and the Leap Motion controller (see figure 3.3).

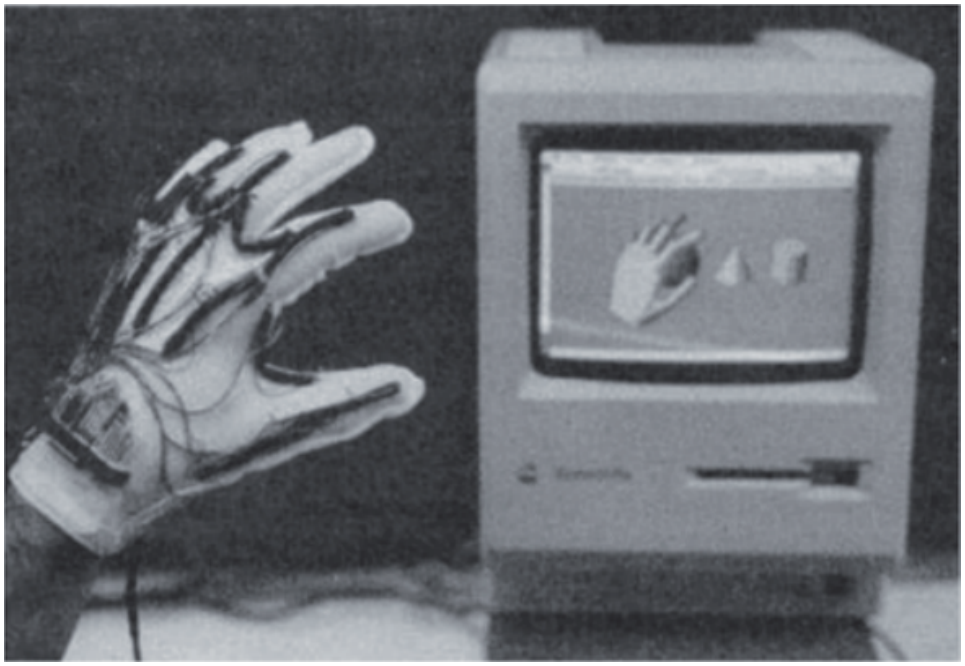

Figure 3.1: The Z Glove, developed by Zimmerman in 1982. Picture from Premaratne (2014)

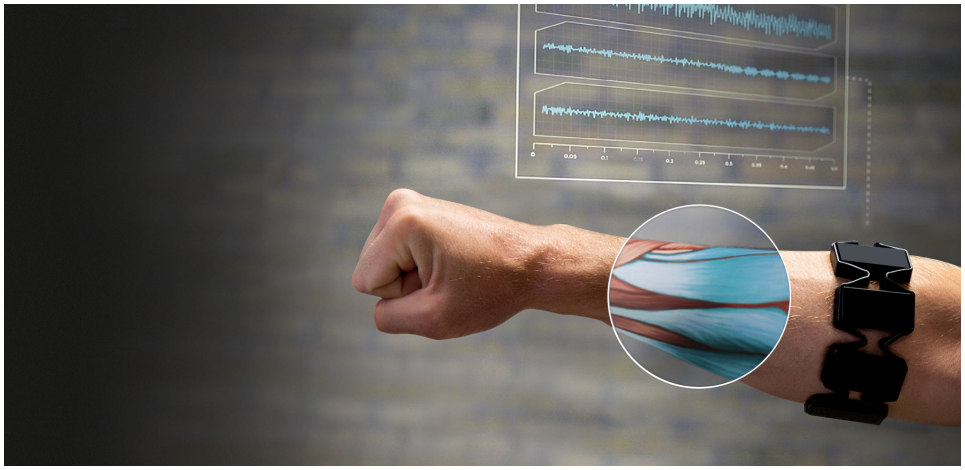

Figure 3.2: The Myo armband is a gesture recognition device worn on the forearm and manufactured by Thalmic Labs. The Myo enables the user to control technology wirelessly using various hand motions. It uses a set of electromyographic (EMG) sensors that sense electrical activity in the forearm muscles, combined with a gyroscope, accelerometer and magnetometer to recognize gestures (Silver, 2015).

Figure 3.3: The Leap Motion Controller is a small USB peripheral device which is designed to be placed on a physical desktop, facing upward. Using two monochromatic IR cameras and three infrared LEDs, the device observes a roughly hemispherical area, to a distance of about 1 meter, and generates almost 200 frames per second of reflected data (Colgan, 2016).

Both approaches have their advantages and disadvantages (see Rautaray and Agrawal (2015) for a deeper discussion of these). Contact-based devices generally have a higher accuracy of recognition and a lower complexity of implementation than that of vision-based ones. Vision-based devices are on the other hand seen as more user friendly as they require no physical contact with the user.

The main disadvantage of contact-based devices is the potential health hazards, which may be caused by some of its components (Maureen Schultz, 2003). Research has suggested that mechanical sensor materials may raise symptoms of allergy and magnetic component may raise the risk of cancer (Nishikawa et al., 2003). Even though vision-based devices have the initial challenge of complex configuration and implementations, they are still considered more user friendly and hence more suited for usage in long run. Because of the reasons outlined above this thesis will primarily be oriented towards vision-based gesture recognition technologies.

3.1.1 The primary Vision-based Technologies

Today, there are three primary vision-based technologies that can acquire 3D images: Stereoscopic vision, structured light pattern and time of flight (TOF) (Ko and Agarwal, 2012). These all make use of one or several cameras and lights to capture and recognize certain movements or poses from the user, and transform it to a certain action on the computer (e.g. a recognized finger tap might be the equivalent to left mouse button click).

Stereoscopic vision is the most common 3D acquisition method and uses two cameras to obtain a left and right stereo image. These images are

	Stereoscopic vision	Structured light	Time of flight (TOF)
Software complexity	High	High	Low
Material cost	Low	High/Middle	Middle
Response time	Middle	Slow	Fast
Low light	Weak	Light source dep (IR or visible)	Good (IR, laser)
Outdoor	Good	Weak	Fair
Depth ("z") accuracy	cm	μm ~ cm	mm ~ cm
Range	Mid range	Very short range (cm) to mid range (4–6 m)	Short range (<1 m) to long range (~ 40 m)
Applications			
Device control			✔
3D movie	✔		
3D scanning		✔	

Figure 3.4: Comparison of Vision-based sensor technologies (Ko and Agarwal, 2012).

slightly offset on the same axis as the human eyes. As the computer compares the two images, it develops a disparity image that relates the displacement of objects in the images.

Structured light pattern measure or scan 3D objects through illumination. Light patterns are created using either a projection of lasers or LED light interference or a series of projected images. By replacing one of the sensors of a stereoscopic vision system with a light source, structured-light-based technology basically exploits the same triangulation as a stereoscopic system does to acquire the 3D coordinates of the object. Single 2D camera systems with an IR- or RGB-based sensor can be used to measure the displacement of any single stripe of visible or IR light, and then the coordinates can be obtained through software analysis.

Time of flight is a relatively new technique among depth information systems and is a type of light detection and ranging (LIDAR) system that transmits a light pulse from an emitter to an object. A receiver determines the distance of the measured object by calculating the travel time of the light pulse from the emitter to the object and back to the receiver in a pixel format.

Of these technologies stereoscopic vision is perhaps the most promising one for the consumer market as it has the lowest material cost (Ko and Agarwal, 2012), and has proved more reliable in variable light conditions than its counterparts. One of the latest consumer-oriented devices of this kind is the Leap Motion Controller, which distinguishes itself for having a higher localization precision than other depth vision-based devices (Weichert et al., 2013), and also for capturing depth data related to palm direction, fingertips positions, palm center position, and other

relevant points (Lu et al., 2016). The Leap Motion Controller will be reviewed more in-depth in the next section.

3.1.2 How vision-based devices functions

3.2 Gesture Recognition Principles

A gesture can be defined as a physical movement of the hands, arms, face and body with the intent to convey information or meaning (Mitra and Acharya, 2007), Even though the use of keyboard and mouse is a prominent interaction method, there are situations in which these devices are impractical for human-computer interaction (HCI). This is particularly the case for interaction with 3D objects (Rautaray and Agrawal, 2015).

To be able to convey semantically meaningful commands through the use of gestures one must rely on a gesture recognition system, which is responsible for capturing and interpreting gestures from the user and, if applicable, carry out the desired action. Often this process is seen as a sum of three fundamental phases: Detection, tracking and recognition (Rautaray and Agrawal, 2015). This section will describe what makes up a gesture recogntion system.

3.2.1 Detection

The first step in a typical gesture recognition system is to detect the relevant parts of the captured image and segment them from the rest. This segmentation is crucial because it isolates the relevant parts of the image from the background to ensure that only the relevant part is processed by the subsequent tracking and recognition stages (Cote et al., 2006). A gesture recognition system will typically be interested in hand gestures, head- and arm movements and body poses, and thus only these factors should be observed by the system.

3.2.2 Tracking

The second step in a gesture recognition system is to track the movements of the relevant segments of the frames, e.g. the hands. Tracking can be described as the frame-to-frame correspondence of the segmented hand regions and aims to understand the observed hand movements. This is often a difficult task as hands can move very fast and their appearance can change vastly within a few frames, especially when light condition is a big factor (Wang and Li, 2010). One additional note is that if the detection method used is fast enough to operate at image acquisition frame rate, it can also be used for tracking (Rautaray and Agrawal, 2015).

3.2.3 Recognition

The last step of a gesture recognition system is to detect when a gesture occurs. This often implies checking against a predefined set of gestures,

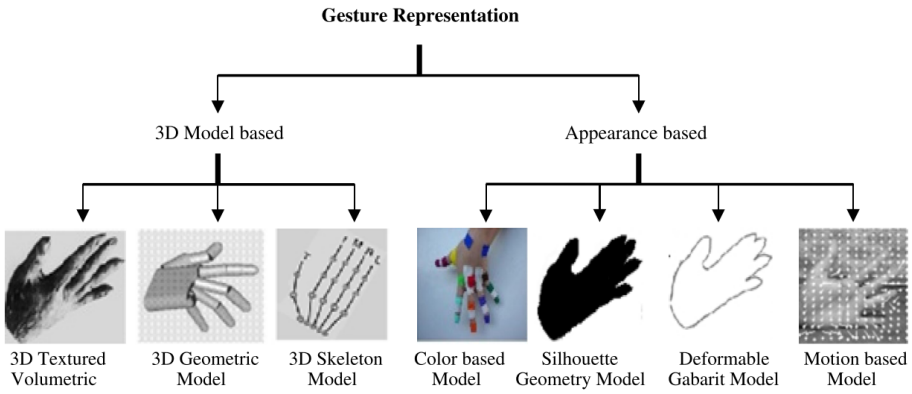

Figure 3.5: Vision-based hand gesture representations (Bourke et al., 2007)

each entailing a specific action. To detect static gestures (i.e postures involving no movement) a general classifier or template-matcher can be used, but with dynamic gestures (which involves movement) other methods, which keep the temporal aspect, such as a Hidden Markov Model (HMM), are often required (Benton, 1995). The recognition technology often makes uses of several methods from the field of machine learning, including supervised, unsupervised and reinforced learning.

When a gesture recognition system detects a relevant segment, it is thus tracked and represented in some way in the system. For hand gesture representations, which is the most relevant for this thesis, there are two major categories of hand gesture representations: 3D model-based methods and appearance-based methods (Rautaray and Agrawal, 2015).

3.3 Challenges with VR and GRT

Problems with using VR + e.g Leap over mouse + keyboard + display. E.g:

3.3.1 The "writing issue"

Virtual keyboards are bad. Regular keyboards are impractical. See "ideer til masteroppgaven.txt"

3.3.2 Challenges in "designing" gesture schemes

People have different preferences. Have intuitive gestures. Have gestures that is not too fatiguing. Have gesture with high precision and recall (F-score) (high TP and TN. Low FP, FN). Have a system that doesnt mistake one gesture for another.

Fixes?

User-gesture calibration.

3.4 Related work

Chapter 4

A review of the Leap Motion Controller

The latest technological breakthrough in gesture-sensing devices has come in the form of a Leap Motion Controller (Leap Motion, San Francisco, CA, United States). The controller, approximately the size of a box of matches, allows for the precise and fluid tracking of multiple hands, fingers, and small objects in free space with sub-millimeter accuracy (Guna et al., 2014).

4.1 Physical properties

The Leap Motion Controller (see fig. 3.3 and 4.1) contains two stereoscopic cameras and three infrared LEDs, which periodically emit a light pulse with a wavelength of 850 nanometer, and thus outside the visible light spectrum. During the light pulses, which light up about eight cubic feet in front of the controller, grayscale stereo images are captured by the cameras and sent to the Leap Motion tracking software (Colgan, 2016). In the software, the images are analysed to reconstruct a 3D representation of what the device sees, compensating for static background objects and ambient environmental lighting.

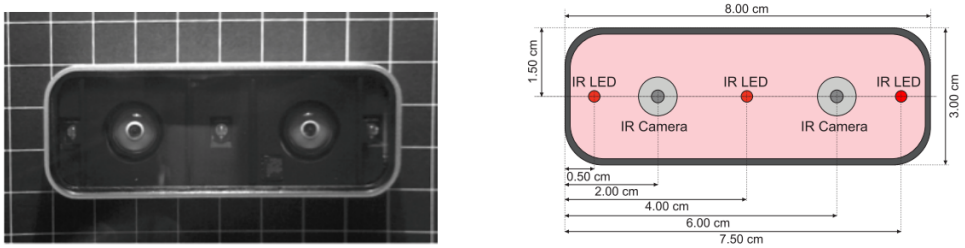

Figure 4.1: Visualization of a Leap Motion Controller, with Infrared Imaging (left) and a Schematic View (right) (Weichert et al., 2013).

4.2 The Leap API

The controller itself can be accessed and programmed through Application Programming Interfaces (APIs), with support for a variety of programming languages, including C++, C#, Objective-C, Java, JavaScript and Python. Although the API is programmed almost exclusively in C, access through a variety of other languages is achieved by virtue of various "wrapper libraries", which exposes and translates functions from their respective languages into the corresponding C function[cite].

The Leap Motion SDK also features integration with commercial game engines such as Unity and the Unreal Engine (Guna et al., 2014). One example of what the Leap Motion API enables is acquisition of the recognized object's position through Cartesian and spherical coordinate systems, which are used to describe positions in the controller's sensory space (Guna et al., 2014).

Technically, very few details are publicly known about the precise nature of the algorithms used due to patent and trade secret restrictions.

4.3 Important Leap components

Hands, fingers, palm, directions, frames, interaction box etc. See

4.4 Detectors - The building blocks of gesture recognition

The detector scripts. How they can be combined. Logic gates.

4.5 Integration with the Unity editor

How the pieced fit together. How the stuff is organized (e.g the modules).

Chapter 5

Designing the virtual design review application

5.1 DNV GL and their motivations

DNV GL is the world's largest classification society with more than 13 000 vessels and mobile offshore units, which represents a global market share of 21% (Jeffery, 2015). It is the world's largest technical consultancy to onshore and offshore wind, wave, tidal, and solar industries, as well as the global oil & gas industry – 65% of the world's offshore pipelines are designed and installed to DNV GL technical standards (Paschoa, 2013). A major part of DNV GL's work is evaluation and quality assurance of a client's product (e.g. a ship) , where a DNV GL "Approval Engineer" conducts a design review of the client's model of the proposed product. This process usually consists of the following steps:

1. The designer sends the model to DNV GL for evaluation.
2. The approval engineer inspects the model noting down aspects that doesn't meet DNV GL requirements.
3. The designer receives the remarks and makes the necessary changes to the model.
4. This process is repeated until both parties are satisfied.

DNV GL is looking into the possibilities of digitilizing this process, and making it more interactive and efficient by using virtual reality technology to conduct virtual design review meetings in the 3D models. As the sense of scale is important in a 3D model review, virtual reality technology is deemed promising as it gives a unique sense of scale and a depth, which is hard to match by regular "2D screens". DNV GL is also interested in alternate interaction methods, as mouse and keyboard can have some limitation when working in a 3D environment (Rautaray and Agrawal, 2015). As mentioned in the previous chapters this thesis will use the Leap Motion Controller, a vision-based device using stereoscopic cameras, as a primary input device to the application.

5.2 Initial design ideas

The core functionality in the application should be to navigate the 3D model and annotate it, primarily by using the advantages of virtual reality and gesture recognition. The users should be able to create "sessions" that enable several users to be virtually present in the same instance of the 3D model, and to interact with it using gestures. During a session the 3D model may thus contain one or several annotations, which can be interacted with (e.g. edited or deleted). Beyond this there is a lot of functionality which should be in place for a complete product, but which will not be a priority for this thesis as the virtual reality and gesture recognition aspects are the focus.

In the final product the application should support a lot more functionality, some of which is described in the next section.

5.2.1 Application use cases

This section gives an overview of the use cases which is intended for the finished application, some of which will be implemented in this thesis. The user stories that have not been implemented as part of this thesis will be explicitly mentioned or marked italics, while the rest will be in plain text.

Launcher

Note that the Launcher was not implemented as part of the thesis. The Launcher should show up when the application is launched and in many ways function as a "server browser". Here the user can decide between hosting a session on a 3D model he or she has access to, or join an existing session from a list/browser. When the user wishes to initiate or join a session with a particular 3D model to be inspected, he/she should be able to:

- Choose between hosting or joining a session.
- If the user wishes to **host** a session, he or she should be able to:
  - Specify a 3D model from a standard file format to host the session on.
  - Give the session a name
  - Define a password, which will be required to enter the session.
  - Choose between different visibility settings for the session (e.g whether it should show up in the session list)
- If the user wishes to **join** a session, he or she should be able to:
  - Choose a session from the session list and click the join-button to enter.
  - Enter the name of a session in the search text field to search for a session by name. This should also enable to find sessions that are otherwise "hidden".

The Inspector

Once a user has either created or joined a session and is loaded into the model, he or she should be able to:

- Look around, e.g. by using mouse movements or head motions (picked up from sensors in the VR head device).
- Move around on the horizontal plane by using the arrow keys, the WASD keys or by specific gestures (e.g. a "dragging motion").
- Move in the vertical plane increasing or decreasing altitude (i.e "flying").
- Zoom in and out by virtually changing the avatar's own size.
- Annotate the surface he/she is looking at. This should furthermore enable:
  - Choosing between a placeholder-, predefined- or custom defined text and/or an icon.
  - Choosing between several annotation states, e.g. "unresolved", "work in progress", "Ready for approval" and "approved".
  - Automatic saving of the annotation and its coordinates to a log used for easy retrieval of the annotation entries.
  - A threaded follow up discussion of the annotation (i.e adding comments).
- Annotate regions or areas of the 3D model (e.g. annotate an entire room). These "area annotations" should subsequently be modifiable to change the size.
- Link annotations to the DNV GL rules and requirements.
- Draw on the desired surface to make suggestions or highlight, e.g. drawing arrows.
- Choose between enabling or disabling collision and gravity. By default the user should be able to traverse freely without collision, but to enable it can be practical in certain circumstances.
- Obtain the real-world distance between two specified points.
- Bookmark the avatar's current location and orientation to easily be able to go back to bookmarked locations.

Actions done during the 3D model session (such as annotating an object) should continuously be stored in a database. If a user wants to re-enter the session at a later time, this database is read, and the actions done in previous sessions are loaded into the model. By utilizing a database in this way the model files themselves can also remain unedited throughout a session, as opposed to saving annotations into the model files itself, which

could be more inefficient and create model versioning issues. Another upside with utilizing a database is that it enables exposure of the actions done in the sessions to other platforms, such as web applications. This can enable annotation and comments done on the 3D model to become "issues" or "remarks" in more traditional collaboration tools such as Atlassian's Jira or Confluence, although this will not be a focus point for the thesis.

To ensure that the desired application is as intuitive and functional as possible the upcoming master's thesis will also look into several ways of interacting with the 3D model while using virtual reality lenses. Special emphasis will be put on using gestures for certain tasks (such as marking and annotating objects) and evaluating the performance through user testing. Using gestures in combination with mouse and keyboard, game controller and joysticks will also be evaluated to ensure a satisfactory user experience.

5.3 The gestures

5.3.1 The pinch gesture

5.3.2 The straight-hand gesture

5.3.3 The fist gesture

5.3.4 The point gesture

Chapter 6

The Unity Implementation

Chapter 7

Evaluation of the implementation

To evaluate the application's ability to meet user requirements, two rounds of user testing seasons were conducted at the DNV GL headquarters in Høvik, Norway. The first of these season were held the 24th March 2017 and involved one test person, while the second round were held at X and involved Y persons. The users were brought in individually and asked to take a seated position at an ordinary work stations with a mouse, keyboard and display, in addition to a leap motion controller positioned at the desk between the keyboard and the user. A HTC Vive head mount was also present for use during the experimentation phase.

The computer used for the testing had the following specifications (hardware and software):

- An Intel i7 as processor.
- 8 GB of RAM.
- A Nvidia Geforce GTX 1080 graphics card.
- A Windows 10 64-bit operating system (build 14393).
- Unity 5.5.2
- Leap Motion Control Panel version 3.2.0+45899
- Steam VR runtime (for use with the HTC Vive head mount)

After the user was seated the test phases were conducted in the following order (including an estimated of alloted time):

1. 5 minutes of introduction. The users were informed about the purpose of the application, some of its long term goals and its limitations.
2. 10 minutes of demonstration. The users were shown each of the possible actions and the different gestures available to them.

3. 15 minutes of instructions. The users followed a series of instructions and oral explanations to teach them to use the program.

4. 20 minutes of experimentation. The users were asked to use the program freely without any instructions.

5. 10 minutes of questions. The users were interviewed with a series of questions related to the application and their experience using it.

With the exception of the experimentation phase, all the steps above were conducted without the use of a VR head mount. In the experimentation phase the users were asked to divide their time equally between using the application in "desktop mode" (i.e using an regular display without a VR head mount) and "VR mode" (i.e using a VR head mount).

7.1 The instructions

The users were asked to perform the following tasks:

1. The pinch gesture is performed by pushing the thumb and index finger together, while keeping the palm directed against the table surface. Move the hand which holding the pinch gesture to rotate the camera along the X and Y axis.

2. The X gesture is performed by holding your hand straight with all fingers extended, pointing towards the screen and the palm facing downward towards the table surface. Lift and lower your hand to change move the camera along the Y axis.

3. The Y gesture is performed by holding your hand straight with all fingers extended, pointing towards the screen and the palm facing to the side, perpendicular to the table surface. Move it from side to side to move the camera along the X axis.

4. The Z gesture is performed by holding your hand curled up into a fist with no finger extended, pointing towards the screen and the palm facing downward towards the table surface. Move your fist closer or further from the screen to move the camera along the Z axis.

5. Maneuver from your current position around one of the pipes present in the 3D model and back to your original position, using one or both hands.

6. Hold your left hand straight and rotate it so the palm if facing towards you. A menu shaped like a fan should appear and follow the movements of your left hand as long as this gesture is held. Use the index finger of the right hand to select "Toggle Options" and then "Combine XYZ Gestures". To select a button hold the tip of the right index finger close enough (in terms of X, Y and Z axis) to the button for it to gradually highlight. When "Combine XYZ Gestures" has

been selected the X, Y and Z gestures are combined/replaced by a combined XYZ gesture, which is performed the same way as the Y gesture (hand straight and palm down). When now performing and holding this gesture the user can move along the X, Y, and Z axis in the virtual space by moving the hand correspondingly in the physical space.

7. Maneuver as in instruction #5, but this time by using in the combined XYZ gesture. After the user has completed this s/he might switch back to the other gesture scheme by bringing up the menu and select "Toggle Option" and "Distinguish XYZ Gestures", or keep the the combined XYZ gesture.

8. By utilizing the gestures introduced thus far, move the camera so the cursor/crosshair in the middle of the screen is positioned over a nearby object. Perform a pointing gesture by only having the index finger extended and point at the screen (away from you). If this is done correctly a blue sphere should occur, which is called an "Annotation Sphere". This is in short a unit of information related to the position it is attached to. Create two more Annotation Spheres by moving the cursor/crosshair over other nearby surfaces and point.

9. Now annotate/mark an entire object or surface by pointing two fingers ("double pointing") instead of one. These two fingers should ideally be held in a bit of an angle, like a scissor. When done correctly the entire surface or object the cursor/crosshair is indicating should be colored in a similar blueish color as the annotation spheres.

10. Now place the cursor/crosshair over an annotation sphere or an annotated object and either point (if an annotation sphere is selected) or double point (if an annotated object is selected). When done correctly a form containing a text field, a virtual keyboard and some buttons should be displayed.

11. Write "DNV GL" in the text field by utilizing the virtual keyboard. After this click on one of the colored buttons to set a color on there annotation (used to indicate a priority), and click submit to save the changes to the annotation.

12. Open the same annotation again by and delete it by pressing the delete button.

7.2 The questions

At the end of the individual test session the users were asked the following questions:

1. Did you prefer to have distinct gestures for movement along the X, Y or Z axis or did you prefer having it combined in a single gesture?

2. How effective and responsive did you find:
    (a) The pinch gesture?
    (b) The X gesture?
    (c) The Y gesture?
    (d) The Z gesture?
    (e) The combined gesture?
    (f) The point gesture?
    (g) The double point gesture?
3. How easy to use was the menu?
4. How difficult was it to place the cursor/crosshair where you wanted?
5. How difficult or impractical was it to use the annotation form?
6. How was using the application with a virtual reality head mount different from using it in "desktop mode"? Which one did you prefer?

Chapter 8

Conclusion

This essay has given a brief summary of the virtual reality design review application that is going to be implemented for DNV GL as part of the master's thesis, and how virtual reality- and gesture recognition technology can be utilized to potentially improve the human-computer interaction experience beyond that of more conventional interaction methods.

Gesture recognition technology is often divided into the categories of vision-based and contact-based, where the former usually is the preferred one because of user-friendliness and the health concerns associated with the latter. Vision-based gesture recognition devices usually utilize either stereoscopic vision-, structured light pattern- or time of flight techniques, where stereoscopic vision-based devices have proved the most promising. One device of this kind is the Leap Motion Controller, which consists of two stereoscopic cameras and three infrared LEDs and periodically captures grayscale stereo images which are sent to the tracking software, where 3D representations are constructed.

The master's thesis aims to evaluate the performance and user experience of utilizing a Leap Motion Controller in combination with the Oculus Rift and HTC Vive virtual reality headsets during a virtual design review in a complex 3D model. The final application should thus be primarily focused on utilizing the most intuitive ways of interacting with complex 3D models in a collaborative virtual reality setting.

Bibliography

Barrett, J. (2004). Side effects of virtual environments: A review of the literature. *Information Sciences*.

Benton, S. A. (1995). Visual Recognition of American Sign Language Using Hidden Markov Models Accepted by.

Bourke, A., O'Brien, J., and Lyons, G. (2007). Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm. *Gait and Posture*, 26(2):194 – 199.

Brooks, J. O., Goodenough, R. R., Crisler, M. C., Klein, N. D., Alley, R. L., Koon, B. L., Jr., W. C. L., Ogle, J. H., Tyrrell, R. A., and Wills, R. F. (2010). Simulator sickness during driving simulation studies. *Accident Analysis and Prevention*, 42(3):788 – 796. Assessing Safety with Driving Simulators.

Buckley, S. (2015). This is how valve's amazing lighthouse tracking technology works. *Gizmodo*.

Bye, K. (2016). Comparing oculus touch and htc vive technology and ecosystems. *Road to VR*.

Clemes, S. A. and Howarth, P. A. (2005). The Menstrual Cycle and Susceptibility to Virtual Simulation Sickness. *Journal of Biological Rhythms*, 20(1):71–82.

Colgan, A. (2016). Controller - leap motion javascript sdk v2.3 documentation. *developer.leapmotion.com*.

Cote, M., Payeur, P., and Comeau, G. (2006). Comparative study of adaptive segmentation techniques for gesture analysis in unconstrained environments. pages 28–33.

Dean Beeler, E. H. and Pedriana, P. (2016). Asynchronous spacewarp. *Oculus Developer Blog*.

Draper, M. H., Viire, E. S., Furness, T. a., and Gawron, V. J. (2001). Effects of image scale and system time delay on simulator sickness within head-coupled virtual environments. *Human factors*, 43(1):129–146.

Feltham, J. (2015). Palmer luckey explains oculus rift's constellation tracking and fabric. *VR Focus*.

Guna, J., Jakus, G., Pogačnik, M., Tomažič, S., and Sodnik, J. (2014). An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking. *Sensors (Switzerland)*, 14(2):3702–3720.

Jeffery, K. (2015). Dnv gl to unveil rules this year. *Tanker Operator*.

Kelly, K. (2016). The untold story of magic leap, the world's most secretive startup. *Wired*.

Kennedy, R. S. ; Frank, L. H. (1985). A Review of Motion Sickness with Special Reference to Simulator Sickness. *Naval Training Equipment Center*.

Ko, D.-i. and Agarwal, G. (2012). Gesture recognition : enabling natural interactions with electronics. page 13.

Kolasinski, E. M. (1995). United states army research institute for the behavioral and social sciences. *Tech Crunch*.

Kuchera, B. (2016). The complete guide to virtual reality in 2016 (so far). *Polygon*.

Lang, B. (2013). John carmack talks virtual reality latency mitigation strategies. *Road to VR*.

Leadem, R. (2016). Applications of virtual reality. *Virtual Reality Society*.

Lin, J. J. W., Abi-Rached, H., and Lahav, M. (2004). Virtual guiding avatar: An effective procedure to reduce simulator sickness in virtual environments. *Conference on Human Factors in Computing Systems - Proceedings*, 6(1):719–726.

Lu, W., Tong, Z., and Chu, J. (2016). Dynamic Hand Gesture Recognition With Leap Motion Controller. *IEEE Signal Processing Letters*, 23(9):1188–1192.

Maureen Schultz, Janet Gill, S. Z. R. H. F. G. (2003). Bacterial contamination of computer keyboards in a teaching hospital. *Infection Control and Hospital Epidemiology*, 24(4):302–303.

Mitra, S. and Acharya, T. (2007). Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324.

Nishikawa, A., Hosoi, T., Koara, K., Negoro, D., Hikita, A., Asano, S., Kakutani, H., Miyazaki, F., Sekimoto, M., Yasui, M., Miyake, Y., Takiguchi, S., and Monden, M. (2003). Face mouse: A novel human-machine interface for controlling the position of a laparoscope. *IEEE Trans. Robotics and Automation*, 19(5):825–841.

Orland, K. (2013). How fast does "virtual reality" have to be to look like "actual reality"? *Ars Technica*.

Paschoa, C. (2013). Jip collapse assessment of offshore pipelines with d/t < 15. *Marine Technology News*.

Premaratne, P. (2014). *Human Computer Interaction Using Hand Gestures*.

Rautaray, S. S. and Agrawal, A. (2015). Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54.

Robertson, A. (2016). The ultimate vr headset buyer's guide. *The Verge*.

Rolnick, A. and Lubow, R. E. (1991). Why is the driver rarely motion sick? the role of controllability in motion sickness. *Ergonomics*, 34(7):867–879. PMID: 1915252.

S., J. (2016). The tech behind playstation vr and how it delivers 120 hz on console. *Game Debate*.

Silver, C. (2015). Gift this, not that: Myo armband vs this toaster. *Forbes*.

Stanney, K. M. (2002). *Handbook of virtual environments: design, implementation, and applications*.

Stanney, K. M., Hale, K. S., Nahmens, I., and Kennedy, R. S. (2003). What to expect from immersive virtual environment exposure: Influences of gender, body mass index, and past experience. *Human Factors*, 45(3):504–520.

Vafadar, M. and Behrad, A. (2014). A vision based system for communicating in virtual reality environments by recognizing human hand gestures. *Multimedia Tools and Applications*, 74(18):7515–7535.

Wang, X. and Li, X. (2010). The study of movingtarget tracking based on kalman-camshift in the video. pages 1–4.

Weichert, F., Bachmann, D., Rudak, B., and Fisseler, D. (2013). Analysis of the accuracy and robustness of the Leap Motion Controller. *Sensors (Switzerland)*, 13(5):6380–6393.