Universitetet i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: *Computer Science*	Spring semester, 2017 Open / ~~Confidential~~
Author: *Christoffer K. Clausen*	……Christoffer K Clausen…. (signature author)
Instructor: *Erlend Tøssebro* External Supervisor(s): *Dag Sanna (Laerdal Medical)*	
Title of Master's Thesis: *Capillary Refill using Augmented Reality*	
ECTS: *30*	
Subject headings: *Augmented Reality, HoloLens, Universal Windows Platform, Capillary Refill, Image Recognition*	Pages: 64 + attachments: 1 Stavanger, 15th of June / 2017 Date/year

Capillary Refill using Augmented Reality

Author:
Christoffer K. Clausen

External Supervisor:
Dag Sanna

Instructor:
Erlend Tøssebro

Universitetet i Stavanger

Department of Electrical Engineering and Computer Science
Faculty of Science and Technology
University of Stavanger

Abstract

The opportunities within augmented reality is growing. Augmented reality is a combination of the real and the virtual world in real time, and large companies like Microsoft and Google is now investing heavily in the technology.

This thesis presents a solution for simulating a medical test called capillary refill, by using augmented reality. The simulation is performed with an augmented reality headset called HoloLens. The HoloLens will recognise a marker attached to an artificial hand. The marker is used to detect and keep tracking of the position and orientation of the hand. Then a virtual 3D hand will be rendered over the marker on the artificial hand. Inside the artificial hand there is a pressure sensor that will be used to detect when users are adding pressure to the index finger. The finger on the virtual 3D model will then change nail colour on user interaction, and thereby simulating capillary refill.

Preface

This thesis was written for the Department of Electrical Engineering and Computer Science at the University of Stavanger and Laerdal Medical. I would like to thank my instructor from the University of Stavanger, Erlend Tøssebro, for valuable comments and guidance. I would also like to thank Dag Sanna from Laerdal Medical, for being my external supervisor and proving me with the resources needed to complete the thesis.

Contents

List of Figures

1 Project Introduction

Laerdal Medical is a company that specialise in products used in medical training. They develop a wide range of products, where some of the most advanced products can be quite costly for the consumers. Other medical scenarios can also be difficult, or sometimes impossible to simulate with the use of physical equipment. That why it is important to explore the possibility to integrate some of the scenarios digitally, and in a realistic working environment.

Augmented Reality (AR) is a combination of the real and the virtual world. It integrates digital information with the user's real world environment in real time, see section 2.1. AR is one of the technologies that have potential to integrate advanced simulations, by adding digital content to the existing user's environments.

1.1 Motivation

Laerdal Medical have not yet found a good solution for simulating capillary refill. Capillary refill is a way to rapidly test blood flood flow to a tissue by adding pressure to the nail, see section 2.2. A very important criteria is that the simulation is realistic for the user, to trick the brain into thinking that this is a real simulation. Training as if it is a realistic situation is important, such that the user is not only training on the task, but doing so under pressure, which can make the users do mistakes they would not normally do.

Various of simulations have been tested using both light and pressure sensors. Even though it is possible to simulate this with for example a light inside a plastic hand, this can be hard to control. When the user adds pressure to the nail, the nail must change from pink to white in a realistic way. It is hard to turn on a light blob inside a plastic hand, and make it look as a real blood flow. The simulation would also be very sensitive to for example light from the surroundings, which will only make it suitable under certain circumstances.

Rendering a digital hand over the artificial hand, with the use of augmented reality, could be one way to make simulations such as capillary refill possible. If this kind of simulation is possible, it would show that AR is a field worth exploring for further research in medical study or other purposes. It would not only be possible to simulate new scenarios in a realistic way, but also make some of the existing simulations more cost effective.

1.2 Task

The task is to develop an application, which demonstrates how augmented reality can be used to perform a visually realistic capillary refill test on an artificial hand. The following issues must be addressed and resolved:

- From the observer's perspective shall the artificial hand be concealed with a graphical 3D model of the hand. The hand should be able to animate capillary refill. In AR the 3D model shall appear as a hologram, which covers the artificial hand.
- If the observer moves or changes his orientation in space, shall the aforementioned 3D model in AR retain its position and orientation relative to the artificial hand.
- If the artificial hand moves or changes its orientation in space, then the 3D model in AR shall retain its position and orientation relatively to the artificial hand.
- If any object is blocking the line of sight, or part of the line of sight between the observer and the artificial hand, then the parts of the 3D model representing the blocked areas should not be drawn in AR.

A part of the task is also to identify software and hardware needed to solve the above problems. And then see if features and limitations of the selected software and hardware makes it suitable for a medical simulation like capillary refill.

1.3 Outline

Chapter 2 - Background

Provides theoretical background information about tools and resources and relevant for this thesis.

Chapter 3 - System Overview

Presents an overview of the Hololens application, the sensor setup and the windows application.

Chapter 4 - Implementation

Explains how capillary refill is implemented in augmented reality.

Chapter 5 - Analysis

Presents an performance analysis of the HoloLens while running the application.

Chapter 6 - Discussion

Discuss choices and techniques used to create a medical simulation such as capillary refill with HoloLens.

Chapter 7 - Conclusion

Presents a conclusion of this thesis.

2 Background

This chapter introduces the theory behind augmented reality and capillary refill. Further it will give a briefly explanation about some of the most important tools and resources used in this thesis.

2.1 Augmented Reality

Augmented reality was defined by Azuma(1997) [3] as a system of the following three characteristics: A combination of the real and virtual world, interactions in real time and registration of three dimension. Unlike Virtual Reality (VR), which replaces the current reality with a simulation, AR supplements the real world with virtual elements. The coexistence of virtual elements and real world environments can be used to visualise complex concepts, which can be difficult or impossible to achieve in the real world.

The first functional AR systems was invented already in the early 90's, which means that the basic of the technology itself is not the latest thing. However, in the recent years, AR has been connected to new innovative technologies such as mobile devices and wearable computers. With portable devices, the user can in a very simple manner remotely interact with real world elements.

Some of the main challenges with AR on remote devices is the limited computer power. The central process unit (CPU) and the graphics process unit (GPU) need to be small enough to fit in an remote device, but still have enough computer power to make the AR experience seamless. Low processing power will give higher delays while tracking the environments and the objects in it. The optics do also have to be as seamless as possible to make the physical and digital blend realistically together.

Even though the potential of augmented reality is far from fully explored, more and more papers and developers are starting to show their interest for AR. The technology can be used to display useful information such as stock prices or sports score directly on the closest wall, or to play the favourite games in the living room. AR can also be used in study purposes, such as medical training for doctors and nurses. Examples of AR in medical training can be visualisations of organs inside the patients or simulations of patient symptoms, such as capillary refill in a real world environment.

2.2 Capillary Refill

Capillary refill time (CRT) is a nail test widely used by health care workers [2, 15]. It is used to rapidly test the blood flow to a tissue. CRT is a visual inspection of the time the blood uses to return to the nail after it has been emptied by pressure.

CRT is performed by a health care provider by adding pressure on the nail bed, forcing the blood from the tissue, until the nail turns white. Then pressure is removed, and health care measures the time it takes for the nail to change back to the original pink colour, which indicates that the blood has returned.

The upper limit for CRT is normally defined as 2 seconds. Factors such as age and temperatures can increase or decrease this value. The test itself is not enough to conclude any symptoms. CRT greater than 2 seconds may indicate dehydration, shock, hypothermia or other abnormalities.

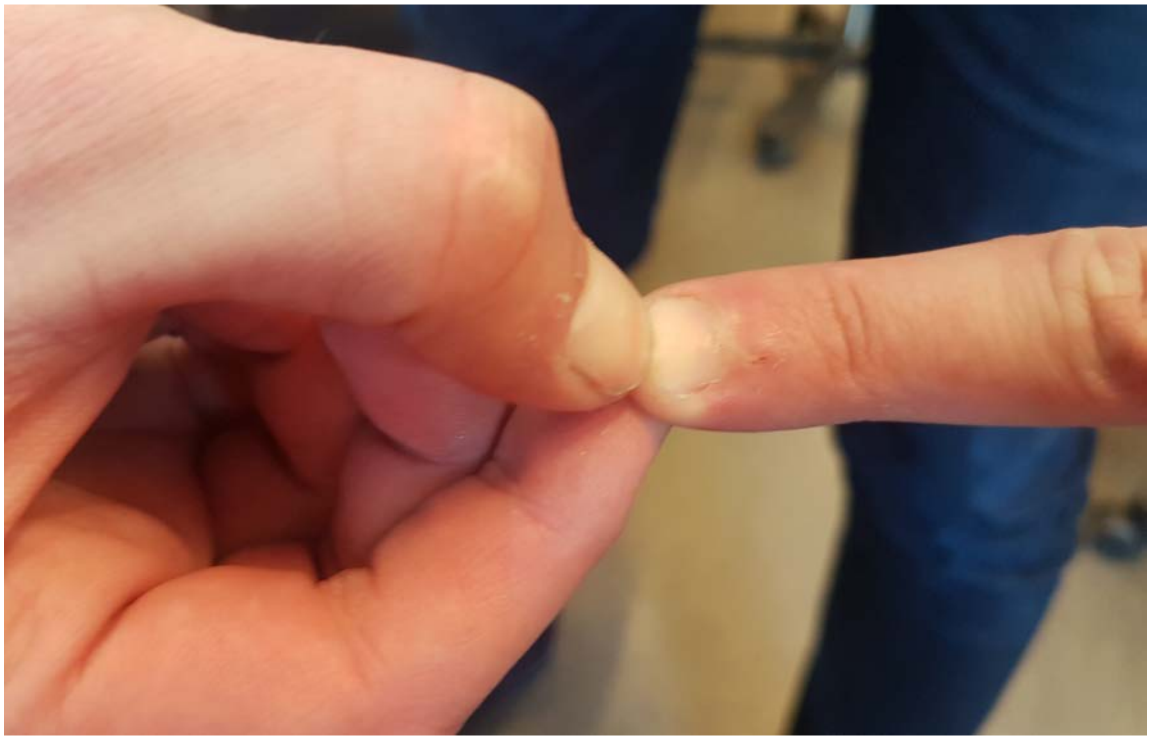


Figure 1: Capillary Refill

Figure 1 shows how capillary refill can be done. In the figure pressure is applied to the nail, and the nail is white. On release, the nail will change back to its original pink colour. The return time can indicate if there are reasons to do more tests on the patient.

2.3 Microsoft HoloLens

Microsoft describes HoloLens as the first self-contained holographic computer, enabling the user to engage their digital content and interact with holograms in the real world around them [11].

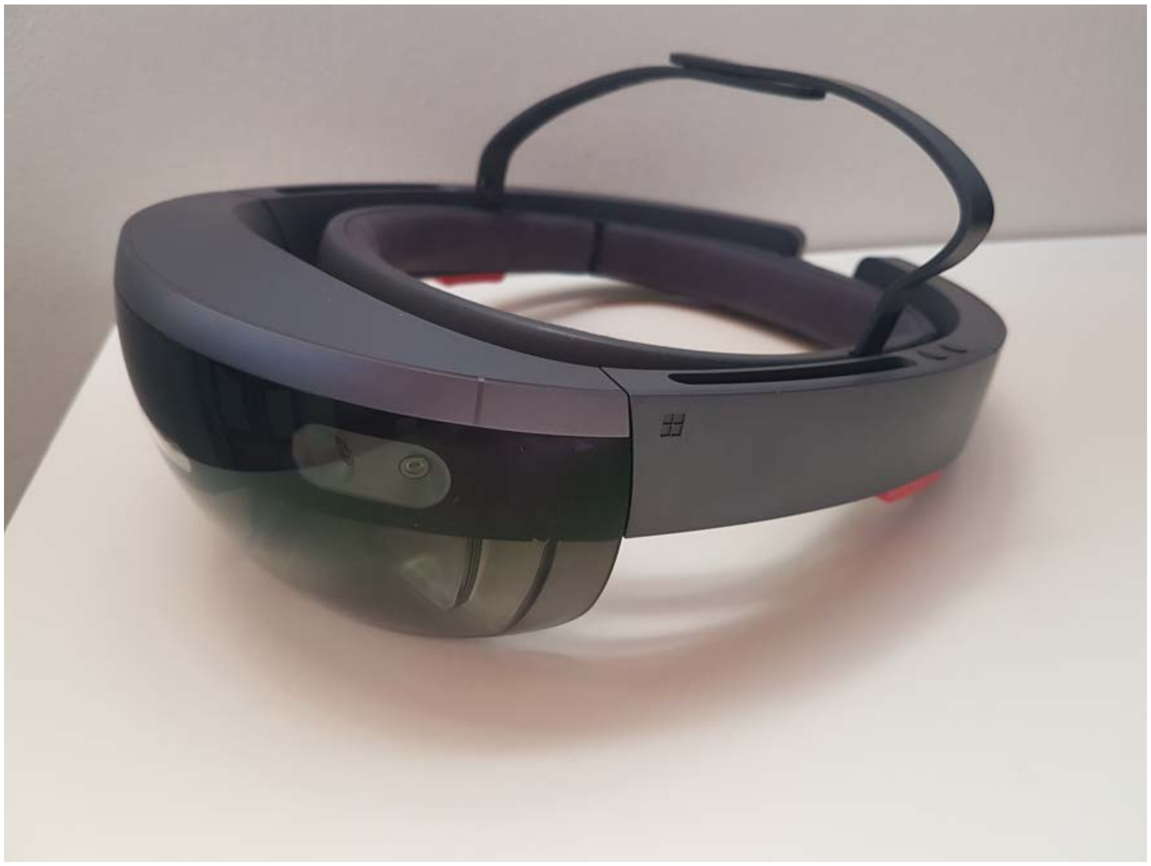


Figure 2: The Microsoft HoloLens

Figure 2 shows how the HoloLens device looks from the front-left. The weight of the device is distributed around the crown of the user's head, and an adjustment wheel is uses to ensure a comfortable fit for the user.

The HoloLens is a headset that uses augmented reality, see 2.1, to create a mixed reality of digital objects and real world environments. This is done by displaying 2D objects or holograms in the environment where the user currently are. Holograms are objects made of light and sound, appearing in the world around the user, simulating that they are actually there. Holograms renders as a frame directly in front of the user's eyes by adding light to the world, which makes it possible to see both the light from the HoloLens display and the world around the user.

The HoloLens knows its surroundings by scanning the environment in real time using spatial mapping [12]. Spatial mapping is a detail representation of the real-world surfaces in an environment. Spatial surfaces are represented as a triangle mesh used to describe real-world surfaces in a small space. By meshing the environment around the user, digital objects can interact naturally with their real environment, like for example a digital ball bouncing on a real floor.

The Microsoft HoloLens also have full windows 10 integration, making it easy to use common tools for web browsing, mail reading, conversation etc. Interacting with the virtual objects in the headset can be done with hand gestures, a clicker called the gaze, or by voice commands. For handling sound, the HoloLens has integrated speakers, so the user can still hear the real environment while wearing the headset.

All of the holographic features are possible due to multiple cameras stationed inside the headset. Inside the HoloLens there are also multiple sensors such as accelerometer and gyroscope, which are keeping track of movements and the position of the user at all times. To handle all the data from the sensors, while making holograms in the environments the HoloLens has a CPU, a GPU and first of its kind, a Holographic Process Unit (HPU).

It is a combination of all these features, like holograms, mapping of the environment and multiple cameras and sensor which makes the HoloLens headset unique compare to other AR headsets. Google Glass is an example of another pair of AR headset released by Google in 2013. Google Glass could provide the user with functions the user would normally find in an smartphone. It was an eyewear that could give the users directions take videos and search the web for information. Unlike HoloLens, the AR eyewear from Google could not recognise advanced hand gestures, head motions and 3D objects in the real world space.

2.3.1 Hardware limitations of HoloLens

One of the limitations of the HoloLens is the size of the lens where holographic objects are displayed. Another limitation is the CPU and GPU size, which have to be small enough to fit inside a headset such as the HoloLens device. This is why Microsoft has chosen to make the field of view smaller than an immersive AR experience would require. A large optical lens would require more computing power than HoloLens have available. Objects that are not inside the view of the lens are not visible to the user, which is something developers need to consider when create holographic applications.

Developers must also be aware of the memory limitation of 2GB RAM (with additional 1GB for the HPU) when developing apps. The app size limitation is set to 900 megabytes [10].

2.3.2 Other limitations of HoloLens

HoloLens have a solid motion tracking by using Kalman filtering combined with the cameras, but the raw data is not available for most developers. Only some partners have full access to the SDK, including all the data from the HPU. This makes it more difficult to make industrial AR with object recognition without the use of third party tools.

2.4 Unity

Unity is a cross-platform game engine. Game engines are designed for creation and development of video games. The game engine works as a template, dictating the possibilities and limits of how components like sound, lighting, physics and animation works in a game.

Unity is used for the creation of 2D and 3D games and for making interactive content. In Unity, scenes contains the content of the game. Scenes works as a stage or a level, where the developers add their unique environments, obstacles, lights, sounds etc. Everything in Unity is defined as GameObjects. A GameObject can be anything from the environment to a special effect. GameObject is nothing alone, but a container, where the developer can add pieces, named components. Components are defined by Unity as the nuts bolts of object and their behaviour in game.

If the developer wants to add new functionality, which is not found in the Unity Editor's functionality core, they can add scripts. Scripts works as a components to an object, making it possible to implements new features. Scripts are basically codes that decides how an object shall behave. Unity support JavaScript and C# as programming languages.

In the end of November 2016, Unity released version 5.5, where they added support of Microsoft Holographic (HoloLens). This update makes it possible to add holographic features right in the Unity editor. Unity was chosen to be one of the first engines with HoloLens support. One of the main reason Unity was chosen, is a mature game engine with .NET 3D support, in addition to a large developer base. This mean that it is relative easy for user to add and customise 3D models as holograms in the Unity editor, and then launch the application as a .NET Visual Studio solution.

2.5 Adobe Illustrator

Adobe Illustrator is vector drawing application used by graphics designers, artists, web designers etc [16]. It is used to create vector images like logos, website compositions and info graphics in both printed and digital form.

The main advantage of an vector images is that it can scale up and down infinitely without any penalty or pixelation. This means that the user can scale the image as much as desired without loosing any quality. Vector images has this quality because they are not made up of a grid of pixels. Instead, they are created by paths, which is a combination of starting and ending points with a combination of angles, lines and shapes in between them. The relation between these path are purely mathematical, allowing them to be scaled, and rescaled without making them blurry.

Adobe Illustrator also supports scripts, which is commands telling the application to perform some desired tasks. Example usage of scripts can be to break text into line or check if a vector image fulfil the desired requirements. AppleScript, JavaScript and VBScript are the supported scripts.

In this thesis Adobe Illustrator is used to create a VuMark target, see section 4.1.3, that is going to be recognised by the HoloLens camera.

2.6 Vuforia

Vuforia Software Development Kit (SDK) specialises in recognition capabilities for mobile devices. Vuforia can be used for developing both AR and VR applications that can recognise images, objects, cylinders and other things. The SDK uses computer vision technology to recognise and track images and objects in real time.

Combined with augmented reality it is mostly used to position a virtual object such as a 3D model in real time. This is done such that the viewers perspective correspond to the orientation of the virtual object in the real world.

In November 2016 Vuforia added the support for HoloLens in their Unity extension [22]. The extension makes it possible for developers implements the recognition capabilities provided by Vuforia using the Unity game engine.

2.6.1 Vuforia VuMark

VuMark is a marker defined by Vuforia acting as a AR target. An AR target is a marker that contains data, which can be recognised and encoded by an AR device. The user can customise the VuMark, but withing a few guidelines made by Vuforia. The guidelines for a VuMark, and how to follow these, are explained further in section 4.1.3. Compared to tracking an image, a VuMark provides a lot more unique unidentifiable instances. More unique instances make is easier to detect a VuMark than for example an image. Similar looking markers can also be separated by detecting a VuMark based on the ID, which is also encoded into the marker.

VuMark is the selected detection and tracking method used in this thesis. It is used as a marker on an artificial hand, which the HoloLens camera uses to detect its position and orientation.

2.7 Arduino

Arduino is an open-source platform, producing hardware and software for electronics [4]. Even though it is made for specific electronics, it can be used to send and receive information to most devices. Arduino is popular because it is easy to set up, and it is known to be user friendly. Today, Arduino is mostly used in microcontroller hardware and software. A microcontroller is a circuit board with a programmable chip, made for various task such as reading data from a sensor, and send it to a computer. Microcontrollers are usually programmed in the low level programming languages C and C++, or languages based on these.

2.7.1 Arduino/Genuino Micro

Arduino Micro(Genuino) is a the smallest board in the Arduino family. Arduino is the name of the products inside US, and Genuino is the name of the products outside. With the exception of the name, there is no difference between the devices.

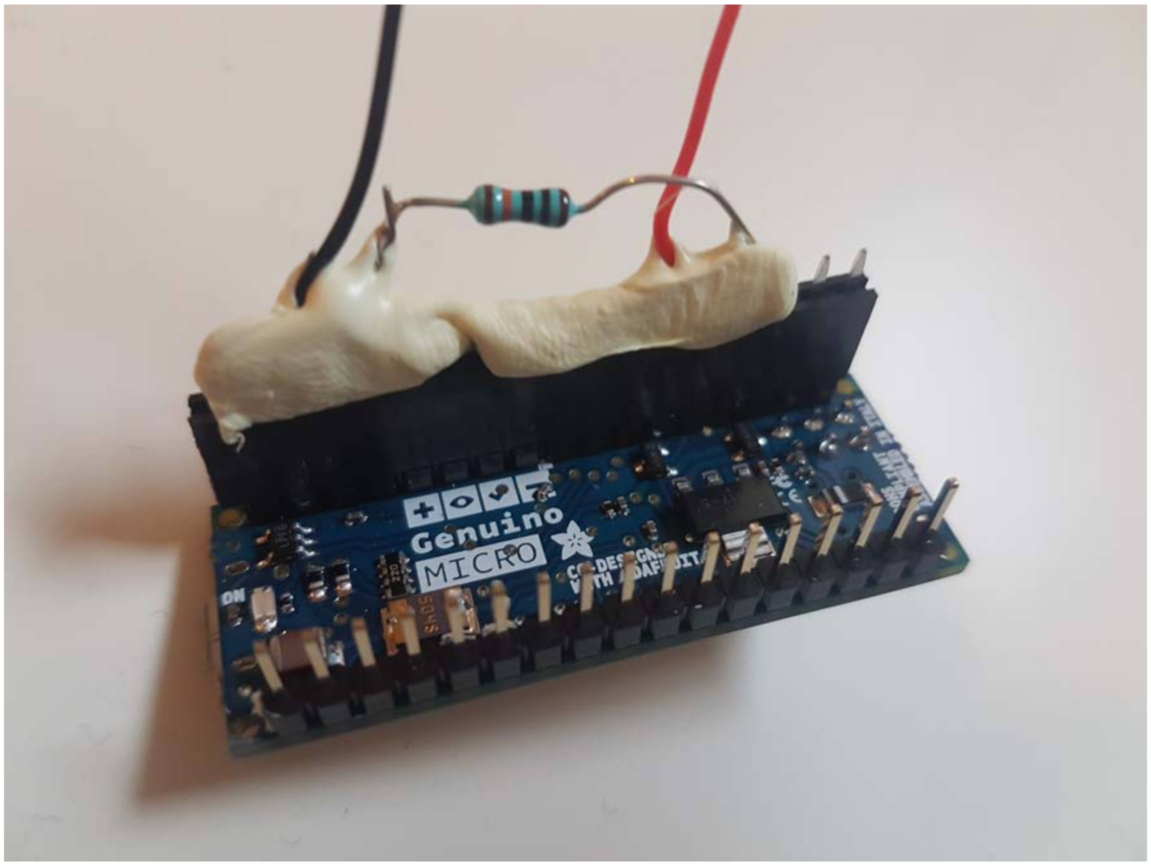


Figure 3: Soldered Genuino Micro

Figure 3 shows the Genuino Micro with soldered wires attached to a Force Sensitive Resistor, see 4.2.1.

The small size of the device makes it ideal for project that require functionality from a microcontroller, without have a lot of space available to store the Genuino. In this report Genuino Micro is used to read data from a force sensitive resistor, and send the data to a computer, see 4.2.2. Further in the report, the Genuino Micro will be referred to as the Arduino. Arduino Integrated Development Environment (IDE), described in the next subsection, is used to program the microcontroller.

2.7.2 Arduino Integrated Development Environment (IDE)

Arduino IDE is an open-source IDE made primarily for Arduino programming. The Arduino language is based on C/C++, but it is simplified with common microcontroller commands from the Arduino Library.

2.8 Force Sensitive Resistor

Force Sensitive Resistors (FSR) are used in many application needing to detect force or pressure in the physical world [17].

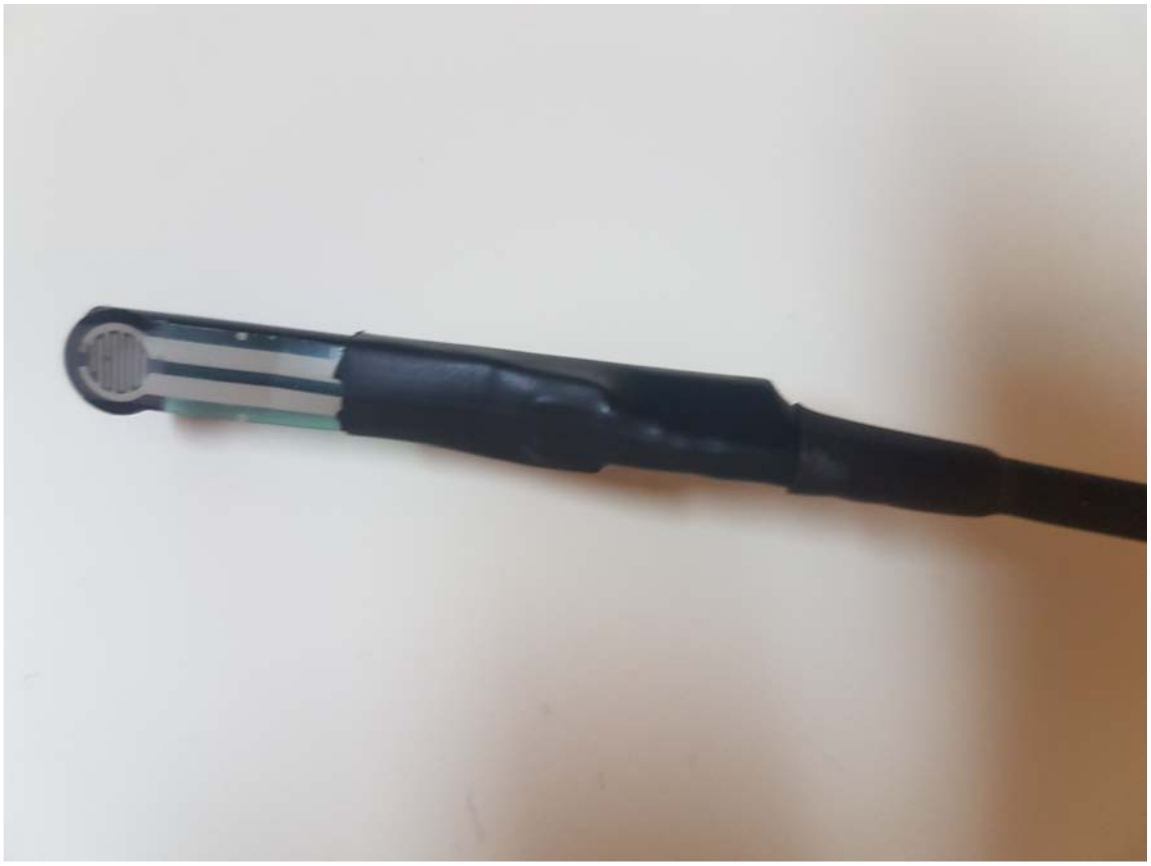


Figure 4: Force Sensitive Resistor taped to a plastic piece

The FSR used in this setup is showed in figure 4 above. A FSR of this type is not outputting where the pressure is applied, but only the maximum amount of pressure applied at any point of the FSR's surface. This FSR will have high resistance when no pressure is applied. The harder the pressure applied by user, the lower the resistance of the sensor, meaning the resistance is inversely proportional to the amount of pressure applied. Two pins extends from the bottom of the sensor, which are used to connect to a board, such as a Genuino Micro. The FSR only gives a rough estimate of the amount of pressure. Therefore it is recommended to only use these type of sensor to recognise if there is pressure applied, but not how much pressure.

3 System Overview

The intentions of this chapter is to give an overview the system setup, and to describe the role of the HoloLens, the PC application, and the force sensitive resistor. The chapter will also provide an insight into the user interface.

3.1 Architecture

The capillary refill simulation setup consist of three main elements. There will be an application running on the Microsoft HoloLens, referred to as the HoloLens application in this thesis. There will also be a separate application compatible with computers, which will be referred to as the PC application. Lastly, there will be a force sensitive resistor connected to an Arduino, mostly referred to as the sensor.

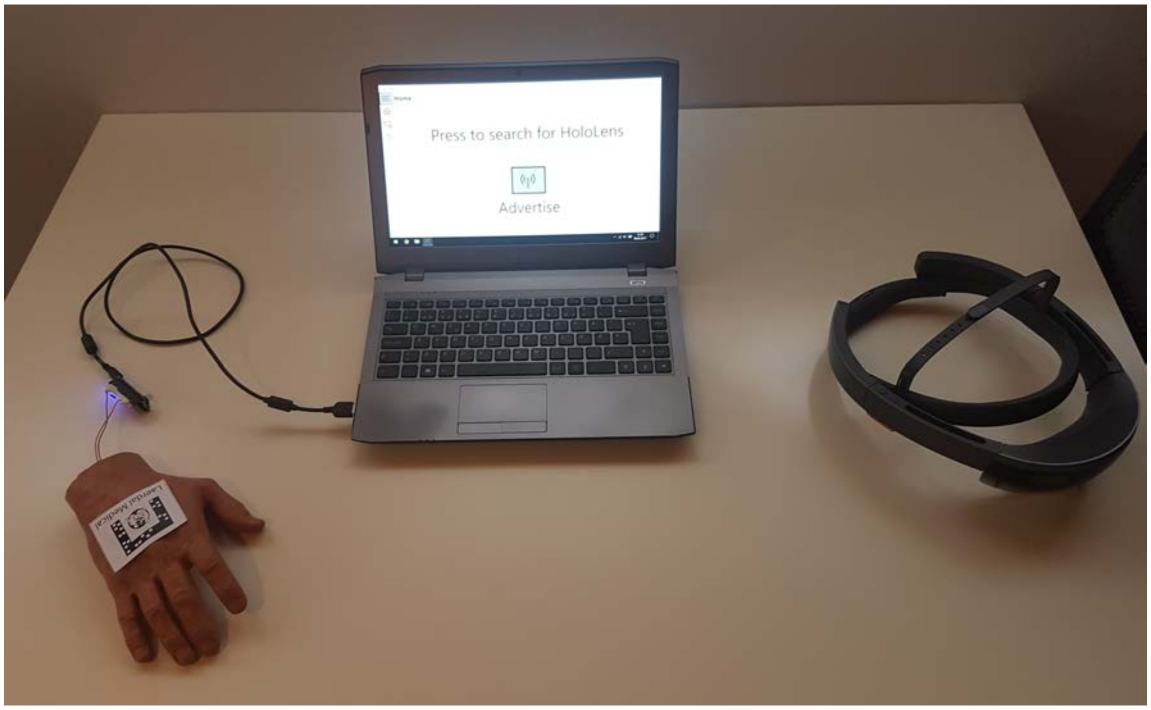


Figure 5: Hardware Overview

Figure 5 shows the three main elements in this simulation setup. As the figure shows, there are wires inside the artificial hand. That is the force sensitive resistor, see section 2.8, which is placed inside the index finger of the hand. The wires are soldered to an Arduino, which is further connected to the computer via an USB cable.

The computer uses the PC application to read the serial data, which is then transmitted to the HoloLens. The computer is wirelessly connected to the HoloLens. Only two devices, respectively a HoloLens and a computer is needed for this setup, and a two-way commutation setup was thereby chosen.

To establish a connection between the devices, the PC application creates a TCP socket, and advertises the socket over Bluetooth Low Energy. The HoloLens will look for the advertisement socket from the computer over Bluetooth. After the devices have found each other, they establish a connection, and stop listening for further Bluetooth packets. The applications will then start listening for messages, which is interchanged over the TCP socket.

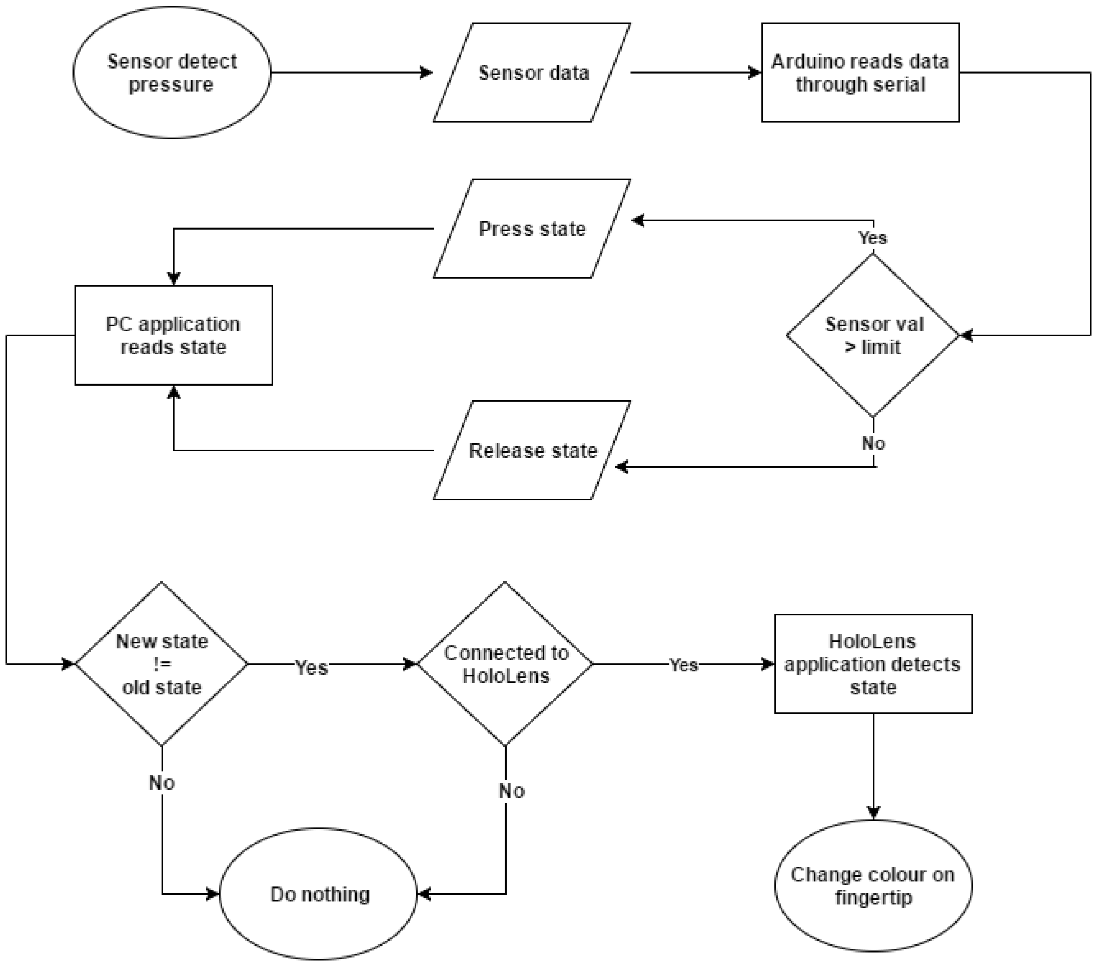


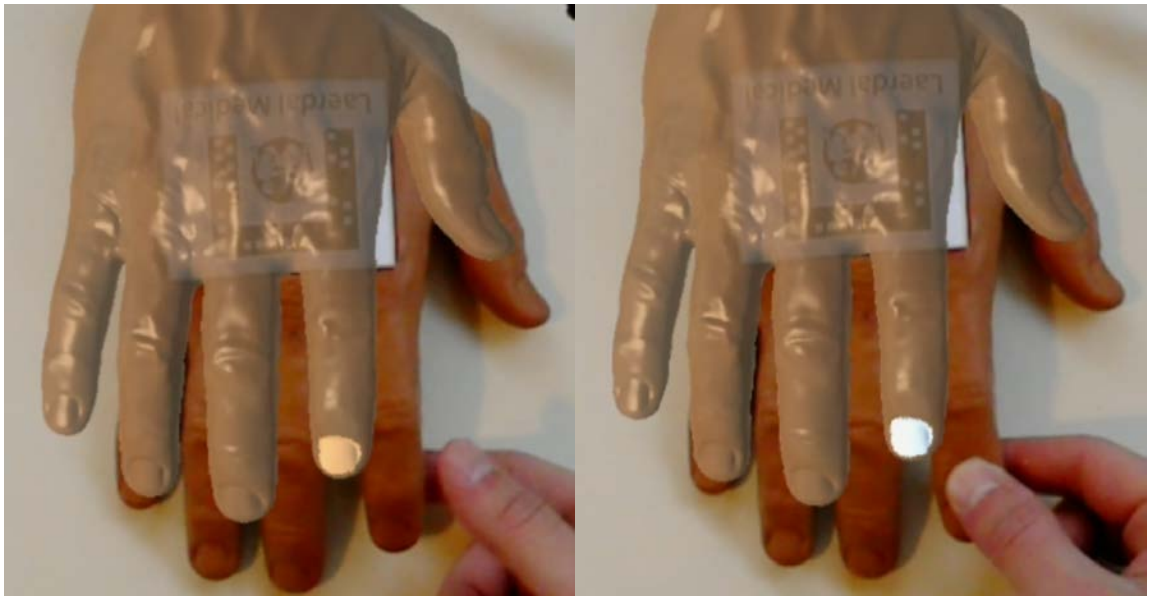
Figure 6: Flowchart illustrates how data i sent from the Arduino, on to the PC, and finally to the HoloLens

Figure 6 is a flowchart, which shows how the sensor inside the artificial hand's fingertip, is used to detect pressure. If pressure is detected on the sensor, the data is forwarded to the Arduino. The Arduino checks if the values are above a specific limit, see section 4.2.2, and will decide if the pressure is high enough for a finger press to be detected. The state of pressure (press/release) will then be sent to the PC application. Each time the finger is pressed or released, the PC application sends the new state to the HoloLens application, which will then change the colour of the fingernail accordingly. The next section will demonstrate the simulation.

3.2 Demonstration

Capillary Refill is simulated by changing the colour on a virtual finger when the users are adding pressure to the index finger on the artificial hand. Completely realistic fingernail colours is not used, since it is hard to distinguish between light pink and white colour through the lenses of the HoloLens. As explained in section 2.3.2, holograms are rendered by adding light to the world, and in a bright user environments, a realistic nail colour would appear white. Then it would be difficult to distinguish between the press and release colour on the fingernail.

To position the virtual hand over the artificial hand, the HoloLens uses a marker from Vuforia called VuMark, see section 2.6.1, as a reference point.

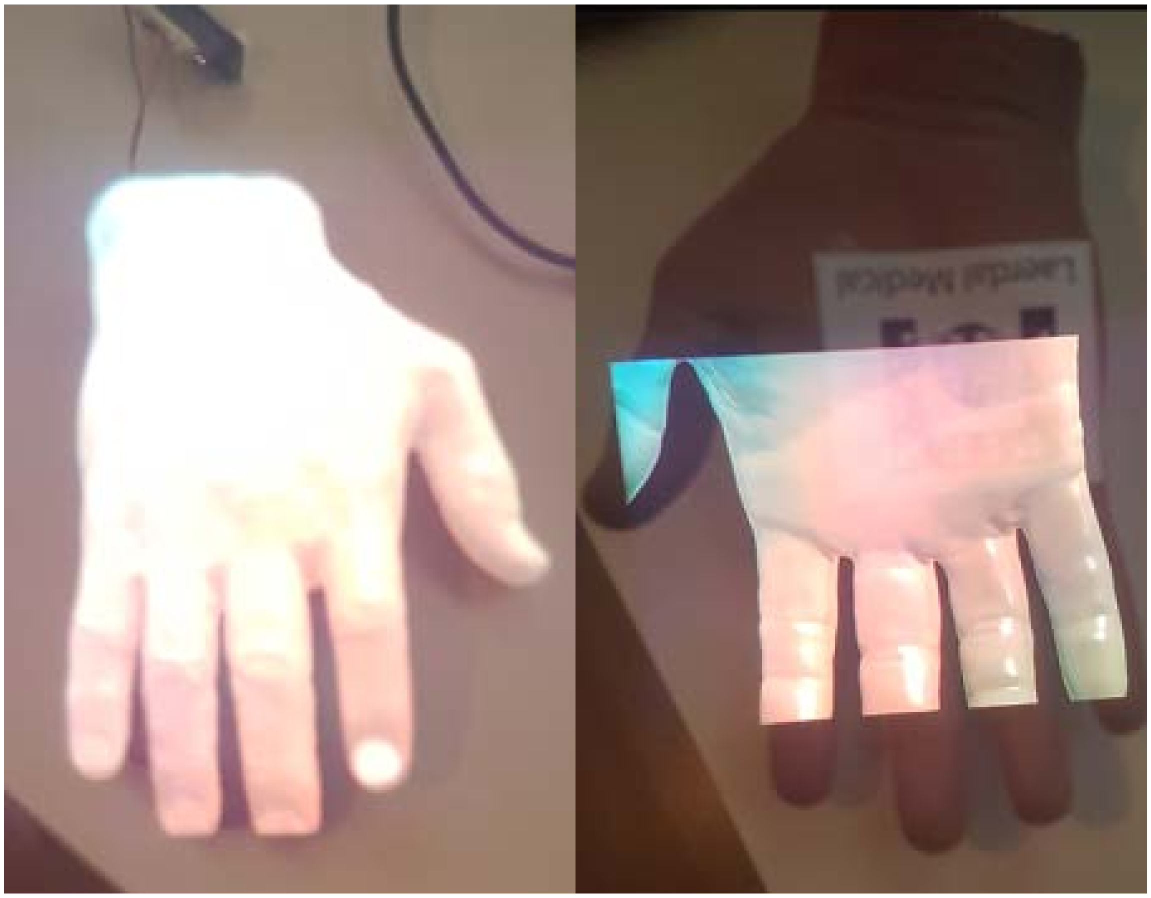


(a) Virtual hand without pressure (b) Virtual hand with pressure

Figure 7: Virtual Hand over an Artificial Hand demonstrating Capillary Refill

Figure 7 shows a recording from the HoloLens camera, displaying the difference between the virtual hand with and without pressure. In figure 7a the fingernail displays a light skin colour, while in the pressed state in 7b the fingernail turns white.

The recording camera is attached in the centre of the HoloLens device, thus unable to display an identical holographic rendering. There are two small lenses directly in front of the user's eyes, which renders the holograms in the world around the users. Thus, the user will have another viewing angle than the HoloLens camera used to record this.



(a) Dorsal side of the virtual hand (b) Palmar side of the virtual hand

Figure 8: Demonstrate how the virtual hand is seen from the user's perspective

Figure 8 tries to demonstrate how the virtual hand will be displayed from the user's perspective. The photo is taken with a another camera through the one of the lenses on the HoloLens device, and the quality of the image is accordingly. Figure 8a shows the dorsal side of the hand, while 8b shows the palmar side. In 8b the image is taken from a closer position than 8a, demonstrating the field of view limitations of HoloLens.

A short demonstration video of capillary refill on HoloLens is available in the appendix A.1.

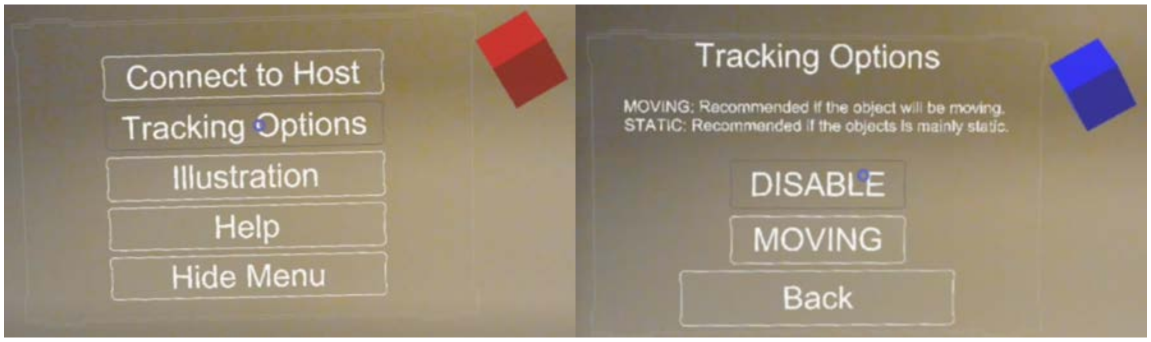
3.3 User Interface

This section presents the graphical user interface on both the HoloLens application and the PC application.

3.3.1 HoloLens Application

When launching the HoloLens application a navigation menu will appear. It is important that the menu is not displayed in unfavourable positions in the environment. By using the finger gesture to hold on the menu, the users can drag the menu around the room, and place it where they desire. The menu was also designed too not draw to much attention, by being partly transparent.

The following figures will display photos of the menu taken with the HoloLens camera. The graphics will be experiences as higher while using the HoloLens.



MOVING: Recommended if the object will be moving.
STATIC: Recommended if the objects is mainly static.

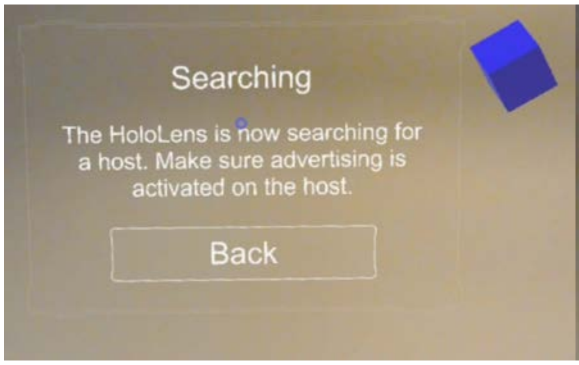
(a) Main Menu (b) Tracking Options

Figure 9: HoloLens Menu

Figure 9a displays the main menu, which is launched when the application is started. The red cube in the top-right corner displays the connection state. The cube will be turn blue when the HoloLens is searching for an advertisement socket from the PC application. When the HoloLens and PC establish a connection, the cube will turn green.

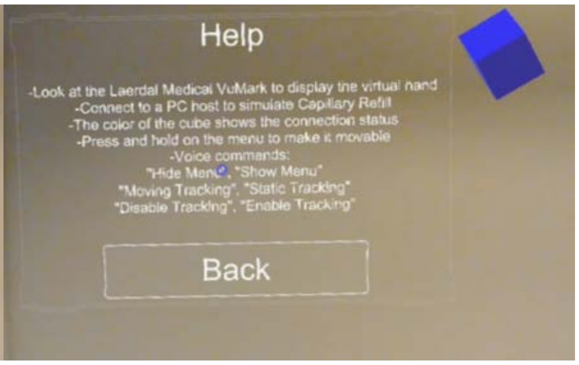
In the main menu displayed in figure 9a, the cursor is pointing on the "Tracking Options" menu item, which is displayed in figure 9b. In this menu, the tracking can be disabled, meaning the HoloLens will not recognise the VuMark anymore. However, if the virtual hand is already rendered, it will not disappear. This functionality is implemented since Vuforia uses the main camera to detect and track the VuMark, thus the camera is not available for live streaming. Live streaming can be useful if other people wants to see the same things as the HoloLens users. Live streaming is implemented in the PC application, as seen in the next section 3.3.2.

There are two types of VuMark tracking implemented into the HoloLens application. The reason for this is discussed in section 6.4. I figure 9b the option is set to "MOVING", which is the recommended option if the artificial hand is moving. This means that the artificial hand will be recognised fast even if it is moving up, down or sideways. However, in other situations, the hand will not be moving, and the tracking can then be set to "STATIC". When the "STATIC" option is selected, the virtual hand will be rendered more stable on the VuMark, but it will not follow a moving hand as well as the "MOVING" option.


Searching

The HoloLens is now searching for a host. Make sure advertising is activated on the host.

(a) Search Menu

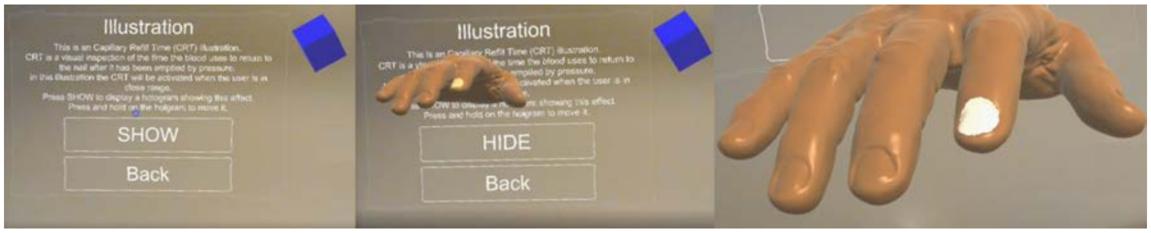

Help

-Look at the Laerdal Medical VuMark to display the virtual hand
-Connect to a PC host to simulate Capillary Refill
-The color of the cube shows the connection status
-Press and hold on the menu to make it movable
-Voice commands:
"Hide Menu", "Show Menu"
"Moving Tracking", "Static Tracking"
"Disable Tracking", "Enable Tracking"

(b) Help Menu

Figure 10: Informative menu items

As seen in figure 10, informative menus are used to explain some of the key functionalities for the users. As listed in figure 10b, the user can also use voice commands to active and deactivate the key functionalities. The application will constantly be listening for voice commands, which is useful if the user don't want to navigate in the menus.



Illustration

This is an Capillary Refill Time (CRT) illustration.
CRT is a visual inspection of the time the blood uses to return to the nail after it has been emptied by pressure.
In this illustration the CRT will be activated when the user is in close range.
Press SHOW to display a hologram showing this effect.
Press and hold on the holgram to move it.

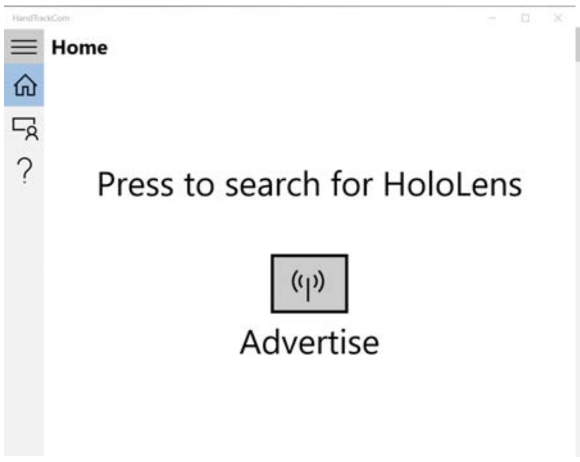
Figure 11: Illustration menu

Figure 11 shows the illustration menu. In the left image in the figure, an explanation of capillary refill is explained. In the middle, the "SHOW" button is pressed and a virtual hand is displaying. In the right image, the user is approaching the virtual hand physically with the HoloLens, and the nail will change colour to white. For the users to be able to adjust the position of the virtual hand used in the illustration, there will be an "invisible cursor" letting the user control and move the virtual hand in this case.

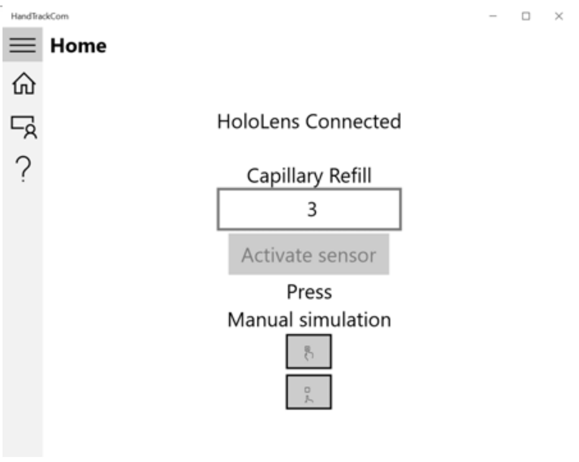
The intention with this illustration is to explain capillary refill, but also show the potential of simulating capillary refill without an external sensor and an external computer. As mentioned, the fingernail is changing colour according to the user's head position, which in this case is the HoloLens camera. For a realistic simulation, the fingernail would have to change colour according to the user's hand position, and also notice if pressure is applied.

3.3.2 PC Application

The Arduino does not have a method to communicate directly with the HoloLens, see section 6.7.2. Thus, a PC application was made to be the link between the HoloLens and the Arduino. The main task of the PC application is to read data from the Arduino, and forward it to the HoloLens.



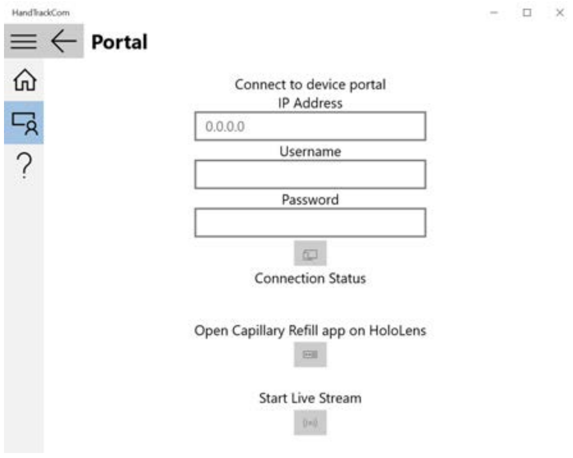
(a) Default



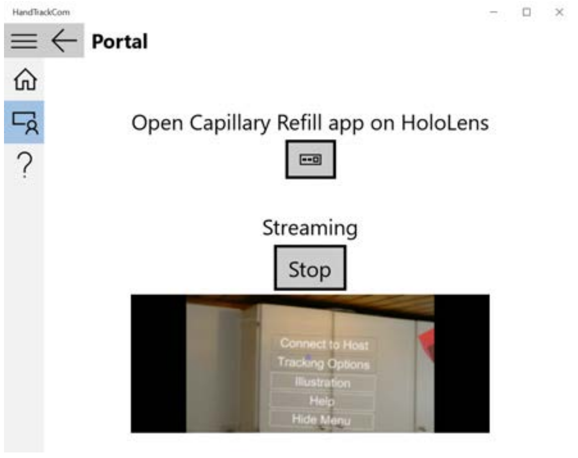
(b) Connected to HoloLens

Figure 12: Home Page

In figure 12 the home page of the PC application is shown. When the user press the advertise button in figure 12a, the PC will start advertising a TCP socket over Bluetooth [19]. If a connection is successfully established, the application redirect the user to a new page, as seen in figure 12b. In the new page, the PC application is automatically sending messages to the HoloLens when the user is adding pressure to the force sensitive resistor. If an Arduino is not available, it is also possible to manually send packets to the HoloLens by pressing the buttons in figure 12b.



(a) Default



(b) Connected to HoloLens

Figure 13: Portal Page

Figure 13 shows the portal page in the PC application. In the portal page is is possible to connect to the "Windows Device Portal" for HoloLens. The "Windows Device Portal" is web server, where he user can manage different options on their HoloLens. Figure 13a shows that IP, username and password is needed to connect the device portal. In figure 13b the user is connected to the device portal, and streaming directly from the HoloLens. Is is also possible to launch the HoloLens application on the HoloLens through this page.

4 Implementation

This chapter aims to explain how the HoloLens application is implemented. It explains how a force sensitive resistor is connected to an Arduino, and communicating with a PC application. Further the chapter describes how the PC application establish a connection with the HoloLens application.

4.1 HoloLens application

This section explains how the HoloLens application, and how the most relevant elements of the application is implemented. To make an application compatible with HoloLens, it have to be an Universal Windows Platform (UWP) application [13]. UWP is the application platform made to work with all Windows 10 devices like HoloLens, PC, tablet, phones etc. This does not necessarily means that all UWP applications works on all Windows 10 device, but the core of the UWP API are the same for all classes of Windows devices. For HoloLens there is an extension SDK, which takes advantage of the HoloLens specific API.

The main tools used to build the HoloLens application in this thesis is Visual Studio 2015, Unity and Vuforia. Visual Studio is used as the code editor, but its primarily use is to deploy the UWP application into the HoloLens. The unique aspect of HoloLens compare to other UWP applications is the 3D holograms, and Unity is used to take advantage of these 3D elements in Windows Mixed Reality API. Vuforia is used to create the recognition specific items in the environment.

Newly developed holographic applications can be tested on a PC without a physical HoloLens, by deploying the solution to the HoloLens Emulator. The emulator can be used to visually watch 3D elements in space, and interact with virtual elements. However, it can not be used to test recognition on a marker such as the Vuforia VuMark made in section 4.1.3. This is why a physical HoloLens is needed to test all the functionalities implemented in this HoloLens application.

4.1.1 Setup Unity for HoloLens

There are some Unity settings needed to make the application compatible and optimised with HoloLens. Firstly, the Unity Camera components will be used to handle stereoscopic rendering and track the movements of the user. This is done by changing the device type to "Windows Mixed Reality", however, a few other settings is needed to optimise the Unity Camera component. The position must be set to x:0, y:0 and z:0 representing the users head position. All background colours must be removed to have a transparent background, which is achieved by changing the RGBA values to (0,0,0,0).

Microsoft recommends setting the minimum rendering distance to 0.85 meters to avoid discomfort by looking at holograms too closely. However, in this thesis the virtual hand must be rendered at a closer distance while pressing on the fingertip to simulate capillary refill. For this reason, the minimum rendering distance is set to 0.2 meters.

Unity has a quality scale where the user can set the quality suited for their application. It is on scale from ”fastest” to ”fantastic”, where ”fastest” will prioritise frame rate and ”fantastic” will attempt to render the highest graphical quality possible. In this application the Unity quality level is set to ”fastest”. The computing power of the device is limited, and it is therefore recommended to prime the frame rate for the best user experience. The performance of the HoloLens application will be evaluated in chapter 5.

HoloToolkit is a collection of components and scripts that Microsoft have made to accelerate the development of holographic applications [14]. In the report the Github repository of the Unity edition of HoloToolkit will only be referred to as HoloToolkit. HoloToolkit is used to import a HoloLens cursor component and the scripts controlling the cursor. The cursor component is modified to only be visible when it is pointing at the virtual menu. Spatial Mapping was also imported, but due to performance issues, see section 6.5, it will not be included in the solution.

To launch the application onto the HoloLens, the Unity project must first be generated to a Visual Studio solution. Then the solution can be opened in Visual Studio and deployed to the HoloLens.

4.1.2 Implement recognition with Vuforia

HoloLens does not have a built in support for image or object recognition, and another solution is thereby required, see 6.2.3. As explained in section 2.6.1, Vuforia VuMark is used as the recognition method for this thesis. All the tools used from Vuforia are found at Vuforia's web page [27], which requires a licenses to access, and will from now on be referred to as the Vuforia developer portal.

The recognition capabilities of Vuforia is implemented directly into the Unity project, with an extension found at the Vuforia developer portal. The VuMark used for recognition can be downloaded directly from the Vuforia database. The Vuforia database is a web based tool, where the users can manage their tracking markers.

Vuforia uses a prefab called ARCamera, which is acting as a scene camera in the Unity project. This camera prefab is used to detect and track the VuMark in the scene. The ARCamera prefab must be bound to the HoloLens camera component in Unity to transform the target into the HoloLens coordinate system. The dataset which contains the VuMark from the Vuforia database is also imported into the ARCamera component.

4.1.3 Creating the VuMark in Adobe Illustrator

The VuMark is acting as the AR target to recognise the artificial hand in this project setup. It is designed in Adobe Illustrator, following specific guidelines and it is also customised to reflect the company Laerdal Medical's brand identity.

In order to design a VuMark that both meets the users design goals and is recognisable by the Vuforia SDK, there are some key elements to every VuMark design [29]. Therefore this thesis will present a brief explanation of these elements. The foundation of a VuMark design is its contour, which is the first thing that the Vuforia computer vision algorithm detects. A VuMark contour must be made up of straight lines between from 4-20 segments. It is the outline formed by the intersection of the border and clear space areas. Further, to make the VuMark detectable, there have to be a high contrast between the border and the clear space area. Another important factor is the VuMark design elements, which are used to encode data within the VuMark design. Each of these code elements consists of dark and bright states, which are alternated in combination with other elements to code data into the design. The code elements acts as a visible representation of the encoded ID in the VuMark. There are also a design area, where the users can add contents of their own choosing, such as a icons or brand logos.

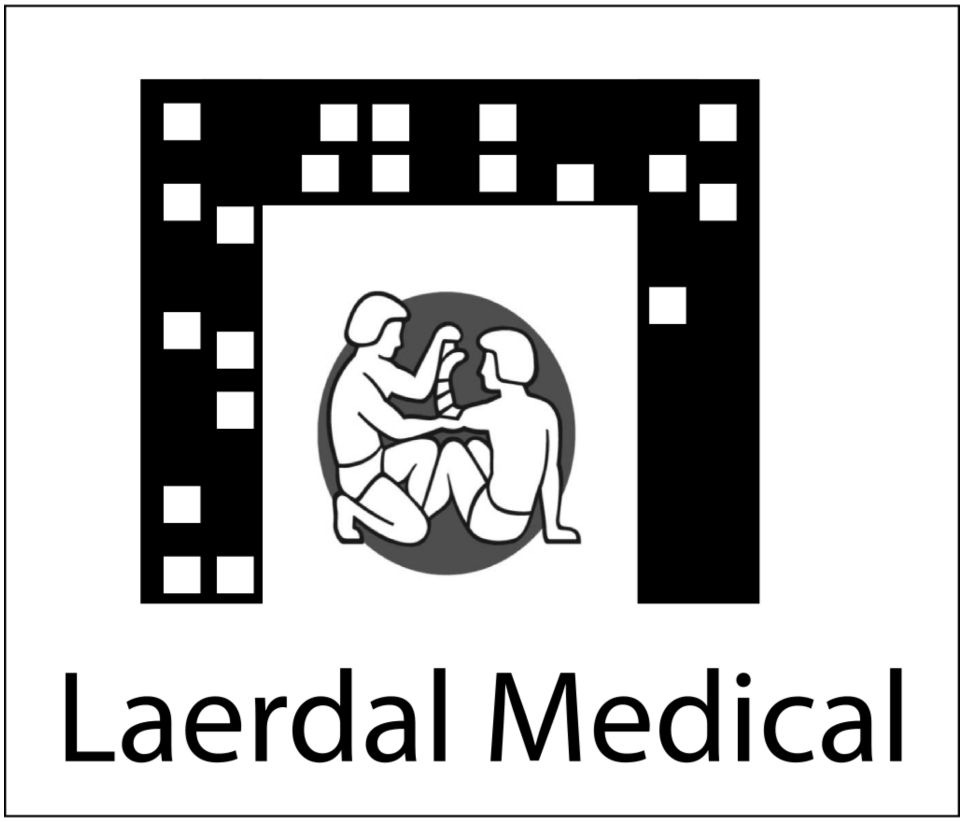


Figure 14: VuMark used for hand recognition

Figure 14 shows how the VuMark used for the hand recognition is designed. The figure shows that there is a high contrast between the border and clear space area, which is black and white. The code elements of the VuMark are all the small visible white squares, and invisible black squares. These are the dark and bright states of the elements, and will have a different combination depending on the ID of the VuMark. For this hand recognition, there are two versions of the printed VuMark. Both with a different ID, one for each side of the artificial hand. Figure 14 also has a Laerdal Medical logo and a text design area. This area is not used for detection, but a graphically rich background improves the tracking.

The first step of creating a VuMark is to acquire the necessary scripts from the Vuforia developer portal. Then these scripts has to be added to the Adobe Illustrator installation. After that, one of these scripts can be used to create a Illustrator template for the VuMark, where name, type and ID length is specified. For the

VuMark used in this project, the type is selected to be a string with an ID length of 2. Meaning two ASCII characters can be used as an ID to recognise different instances of the VuMark. These a represented as the white squares visible in figure 14. As mentioned, there is a printed version of the VuMark for each side the artificial hand. The strings that are encoded into these VuMark's is "up" and "dn", and the corresponding ID is then recognised by the HoloLens with a simple C# script..

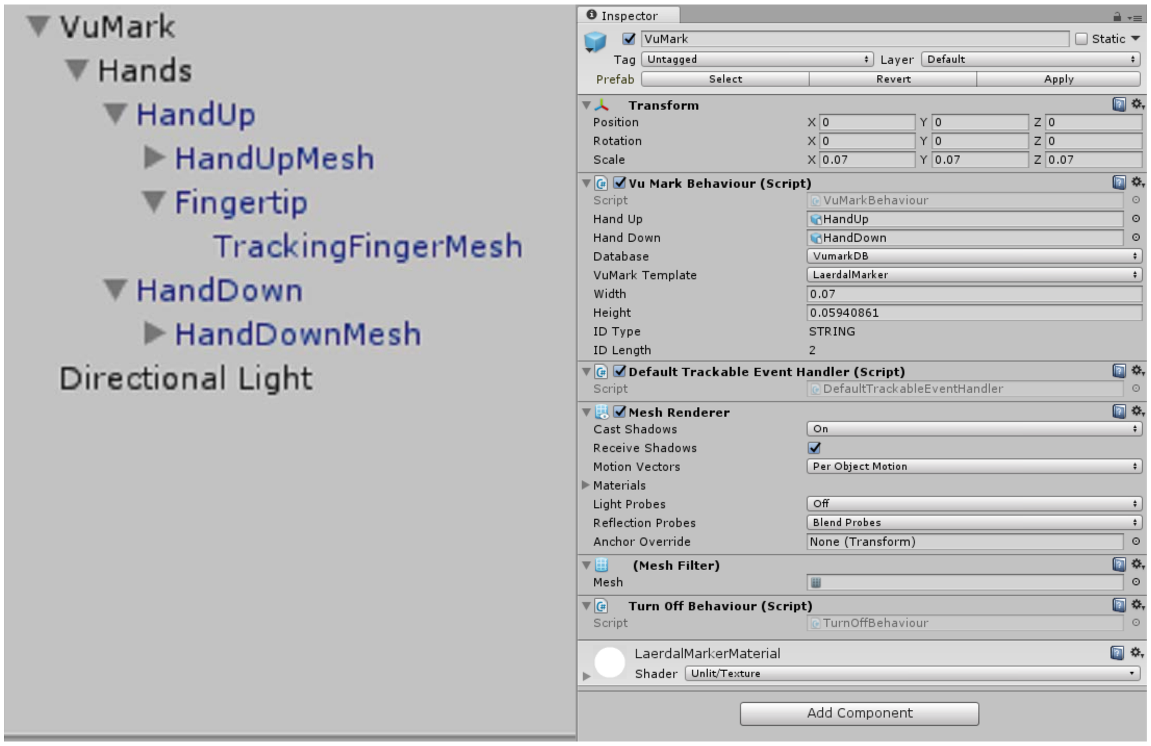
The next step in the design in process is to use all the generated VuMark layers, such as bright- and dark elements, background, border etc. that are explained on the Vuforia web page [21]. All the layers, including all the code elements, have to be manually added in design.

After the design is completed, the last step in Illustrator is to use another script to verify the guidelines of the newly designed VuMark. When all the tests passes, the files is exported as a saleable vector graphics (svg) file, and uploaded to the Vuforia database found at the developer portal. After the VuMark is added to the database, it is can be integrated into the Unity project, and a VuMark instance can also be generated for printing. This is the printed VuMark attached to the artificial hand.

4.1.4 3D model of the hand

Creating the virtual hand is based on a stl file of an arm from Laerdal Medical. Laerdal uses this file to print the skin on their manikins. A 3D modelling tool called Meshmixer is used to convert the stl file to an object file, which required for the Unity project. Meshmixer is also used to cut the arm, such that only the hand remains. The size of the 3D object is also reduced with 70 percent for the HoloLens to use less computing power each time the 3D object hand is being rendered.

A separate object file is made to represent the nail on the index finger. This is the nail which will change colour while simulating capillary refill. Both object files are then implemented into the Unity project. In Unity the fingernail object is positioned on the nail position of the hand object. The 3D objects are added as child GameObjects of the VuMark GameObject in Unity.



```
▼ VuMark
  ▼ Hands
    ▼ HandUp
      ▶ HandUpMesh
      ▼ Fingertip
          TrackingFingerMesh
    ▼ HandDown
      ▶ HandDownMesh
  Directional Light
```

```
Inspector
VuMark                                   Static
Tag Untagged          Layer Default
Prefab  Select  Revert  Apply

Transform
Position      X 0      Y 0      Z 0
Rotation      X 0      Y 0      Z 0
Scale         X 0.07   Y 0.07   Z 0.07

Vu Mark Behaviour (Script)
Script            VuMarkBehaviour
Hand Up           HandUp
Hand Down         HandDown
Database          VumarkDB
VuMark Template   LaerdalMarker
Width             0.07
Height            0.05940861
ID Type           STRING
ID Length         2

Default Trackable Event Handler (Script)
Script            DefaultTrackableEventHandler

Mesh Renderer
Cast Shadows        On
Receive Shadows     ✓
Motion Vectors      Per Object Motion
Materials
Light Probes        Off
Reflection Probes   Blend Probes
Anchor Override     None (Transform)

(Mesh Filter)
Mesh

Turn Off Behaviour (Script)
Script            TurnOffBehaviour

LaerdalMarkerMaterial
Shader  Unlit/Texture

Add Component
```

(a) VuMark GameObject structure (b) Components on the VuMark GameObject

Figure 15: VuMark structure in Unity

Figure 15a shows how the VuMark GameObject is structured in Unity. It shows how the virtual hands are child objects of the VuMark. Functionality are added to GameObjects as components (scripts) in Unity. Figure 15b shows that there are three scripts added to the VuMark. The "VuMarkBehaviour" script is used to initialise the VuMark as a target used for tracking, and to check the ID if the VuMark. If the ID is matched with "up" or "dn", the dorsal or the palmar side of the virtual hand will be rendered on the HoloLens. The other two scripts are also used for tracking related tasks, which will not be explained in details here.

There are also a couple of tracking options made available for the user in the virtual menu. The user can select if the VuMark on the artificial hand is moving or static, and the user can choose to disable tracking if they want to.

Disable tracking stops all the features of Vuforia completely. Normally the virtual hand will not be visible if the HoloLens cannot detect the VuMark. However, if the virtual hand is already visible at the moment of disabling, the virtual hand is set to still be visible. Then the users can still have a virtual hand positioned on the artificial hand. The main reason for this feature is to be able to enable recording, and to save computing power, see section 5.

4.2 Sensor setup

This section explains the setup of the Force Sensitive Resistor (FSR) [1]. The FSR is placed inside the artificial hand, and is used to detect when pressure is added to the index finger.

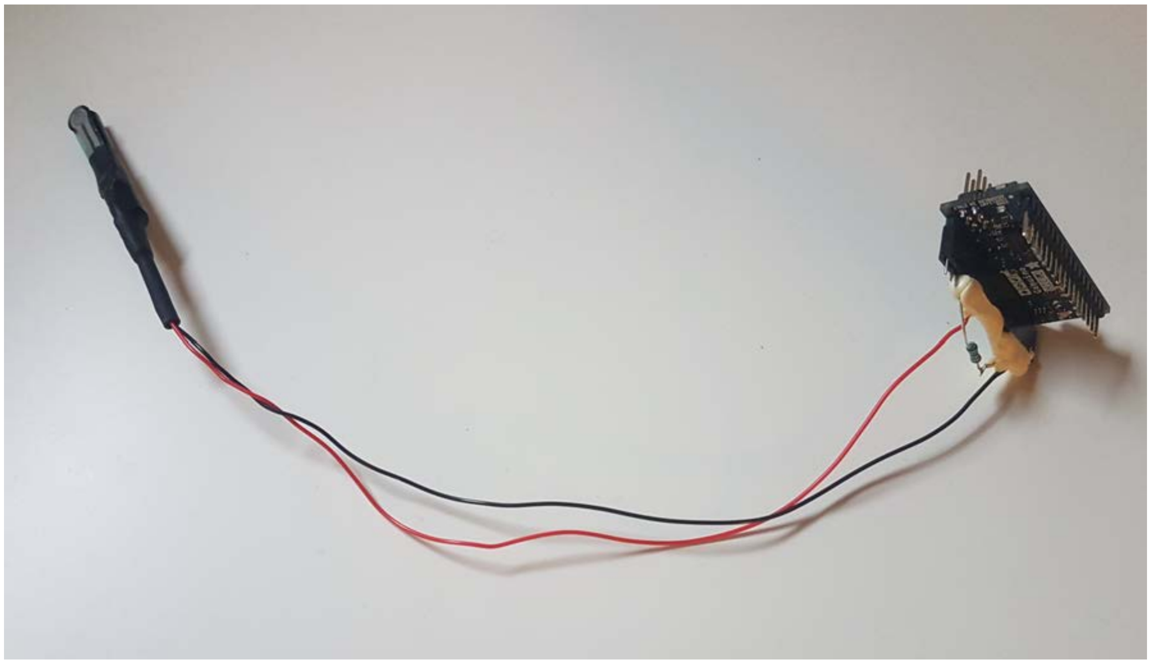


Figure 16: Sensor wired to Arduino

Figure 16 displays an image of how the FSR is soldered to an Arduino Micro board with two wires. The Arduino is programmed to read data from an analog pin.

4.2.1 Attach sensor to Arduino

As mentioned the Arduino is attached to the FSR with soldered wires. It is important that the FSR is able to detect pressure when it is placed inside the artificial hand.

Inside the artificial hand the sensor will always be exposed to some pressure due to direct contact with the artificial hand. A 100K ohm pull-down resistor is chosen, such that the sensor will not detect pressure form the skin inside the finger in the artificial hand.

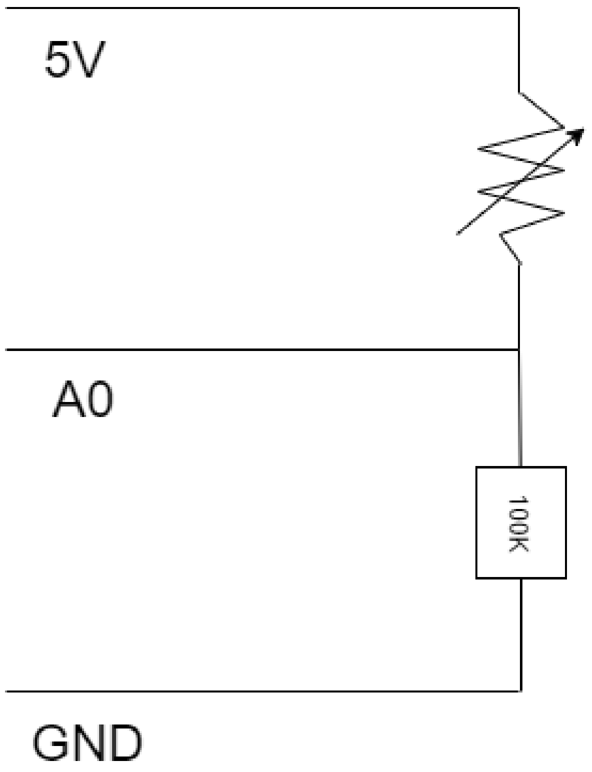


Figure 17: Sensor Setup

Figure 17 illustrates how the wires are soldered to the Arduino. When no pressure is applied to the sensor it acts as an open circuit. Then, when pressure is applied to the sensor its resistance decreases. As shown in the figure, one of the terminals of the FSR is connected to the 5V pin on the Arduino, and the other is connected to analog input 0 (A0). The 100K pull-down resistor is connected between A0 and ground. Thus the voltage on A0 will be 0 when no pressure is applied, and it will decrease with increasing levels of pressure.

4.2.2 Programming Arduino Micro

The Arduino is programmed in the Arduino IDE with the Arduino language. It needs to be programmed to know what pins to read, and to know what information it shall pass into its serial port. The information sent through the serial port is read from the PC application, as showed in the next section.

The Arduino is programmed to constantly read data from the sensor on analog pin 0. If the pressure from the sensor is below a limit, the Arduino will pass through 1's to the PC application. When the value exceeds the limit, the Arduino will pass through 2's to the PC application. These are just arbitrary number, which is easy to interpret for the PC application. The 2's are used to tell the PC application that the pressure is high enough to register pressure to the index finger, and the 1's are the opposite. The limit is set to be low enough to detect when pressure is applied, and high enough to not detect false pressure.

4.3 PC application

The main task of the PC application is to act as a link between the HoloLens application and the Arduino. How this is implemented will be explained in this section.

Just like the HoloLens application, the PC application is also a Universal Windows Platform (UWP) application. This means that this application will be compatible with all Windows 10 devices.

4.3.1 Reading data from microcontroller

The PC application reads data from the Arduino microcontroller to know when the users are adding pressure to the sensor. Firstly the port must be opened to read on the same serial rate as the Arduino is writing to. When the port is open, the application starts to read data form the port.

Listing 1: Reading port data

```
try
{
    Task<uint> loadAsyncTask;
    dataReader.ByteOrder = ByteOrder.BigEndian;
    dataReader.InputStreamOptions = InputStreamOptions.Partial;
    dataReader.UnicodeEncoding = UnicodeEncoding.Utf8;
    uint readBufferLength = 4;
    loadAsyncTask = dataReader.LoadAsync(readBufferLength).AsTask();
    uint ReadAsyncBytes = await loadAsyncTask;

    if (ReadAsyncBytes > 0)
    {
        string data = dataReader.ReadString(ReadAsyncBytes);
        if (data[0].Equals('2'))
        {
            PortStatus.Text = "Press";
            if (!btnPress)
            {
                OnNailPress();
                btnPress = true;
            }
        }
        else if (data[0].Equals('1'))
        {
            PortStatus.Text = "Release";
            if (btnPress)
            {
                OnNailRelease();
                btnPress = false;
            }
        }
    }
}
```

Listing 1 shows how the PC application reads data from the port. The application is reading the data asynchronously, and then converting it to a string. As explained in section 4.2.2, the Arduino will pass through 1's and 2's as the data stream. Listing 1 shows how these numbers are used to decide if the user are adding pressure to the fingernail or not.

4.3.2 Connecting to the HoloLens application

The goal was to create a connection between the PC application and the HoloLens application, which should be intuitive and simple to establish. Two Windows 10 devices, including the HoloLens device, will communicate over a local WiFi network with low minimal infrastructure/setup.

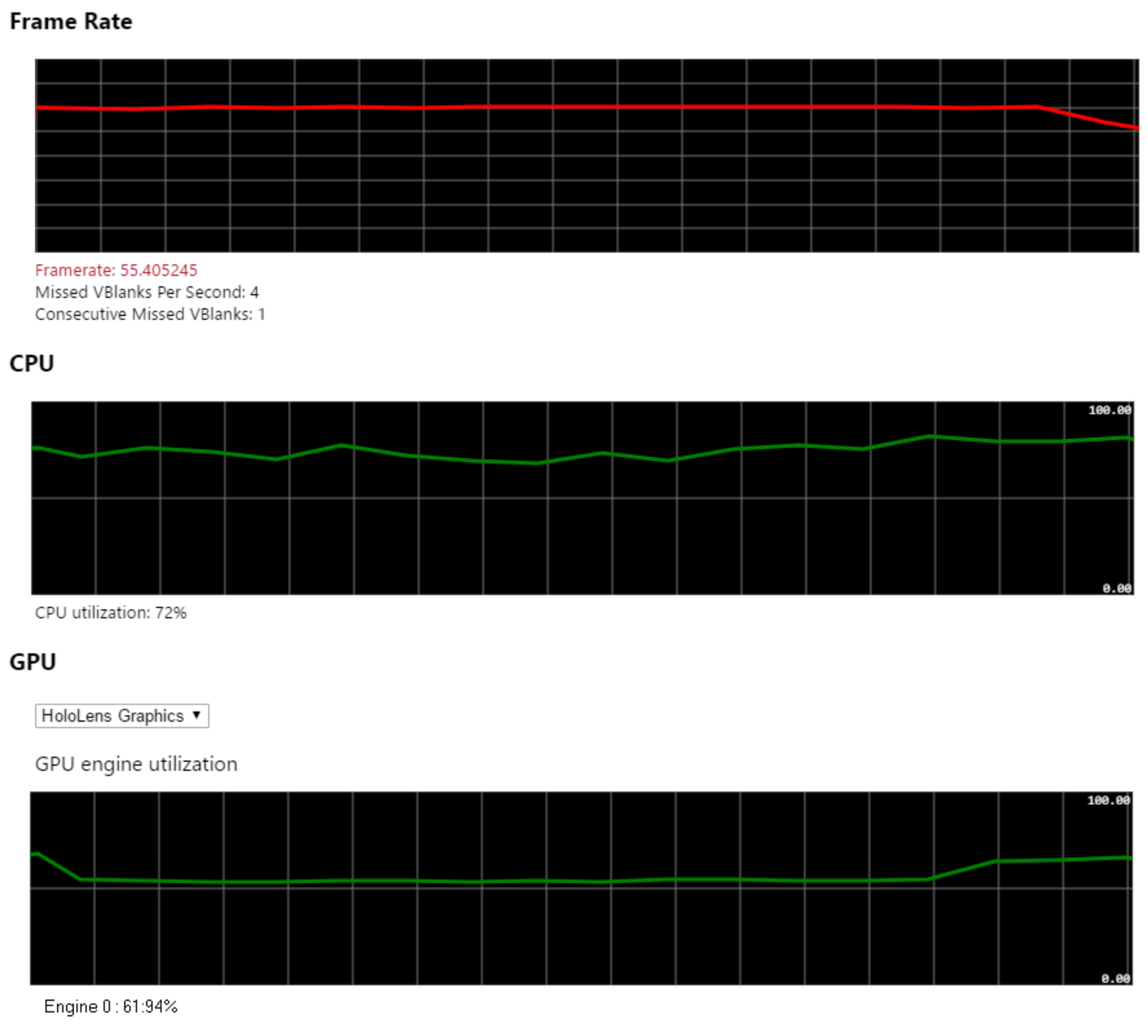
Since only two devices is needed for this setup, a two-way communication is used to establish the connection. A socket library by Mike Taulty, released on Github [20], is used for the two-way commutation. Both HoloLens and computers are supported in the library. The library was also implemented on the HoloLens application, to be able to connect and receive message from the PC application.

The library uses Bluetooth Low Energy to establish a connection by interchanging messages over a TCP socket, see section 3.1. Then the users do not have to setup a network from the PC, by entering a IP address or hostname to establish a connection. The users do only need to press one button to advertise the TCP socket on the PC, and one button to search for the TCP socket on the HoloLens.

After the connection is established, the PC application sends bytes messages to the HoloLens. The HoloLens application uses these messages to know what colour the fingernail on the virtual hand shall be. Due to a small time delay between the PC application and the HoloLens application, the fingernail is instantly changing colour, instead of fading. Then it is clearer for the users to notice when pressure is applied to the sensor inside the artificial hand.

5 Analysis

This chapter presents an analysis of the performance of the HoloLens while the capillary refill application is running on the HoloLens. The performance will be evaluated to see if it matches the optimal frame rate for a smooth user experience. Monitoring the performance might have a small impact on the result, and the real performance might thereby be slightly higher.



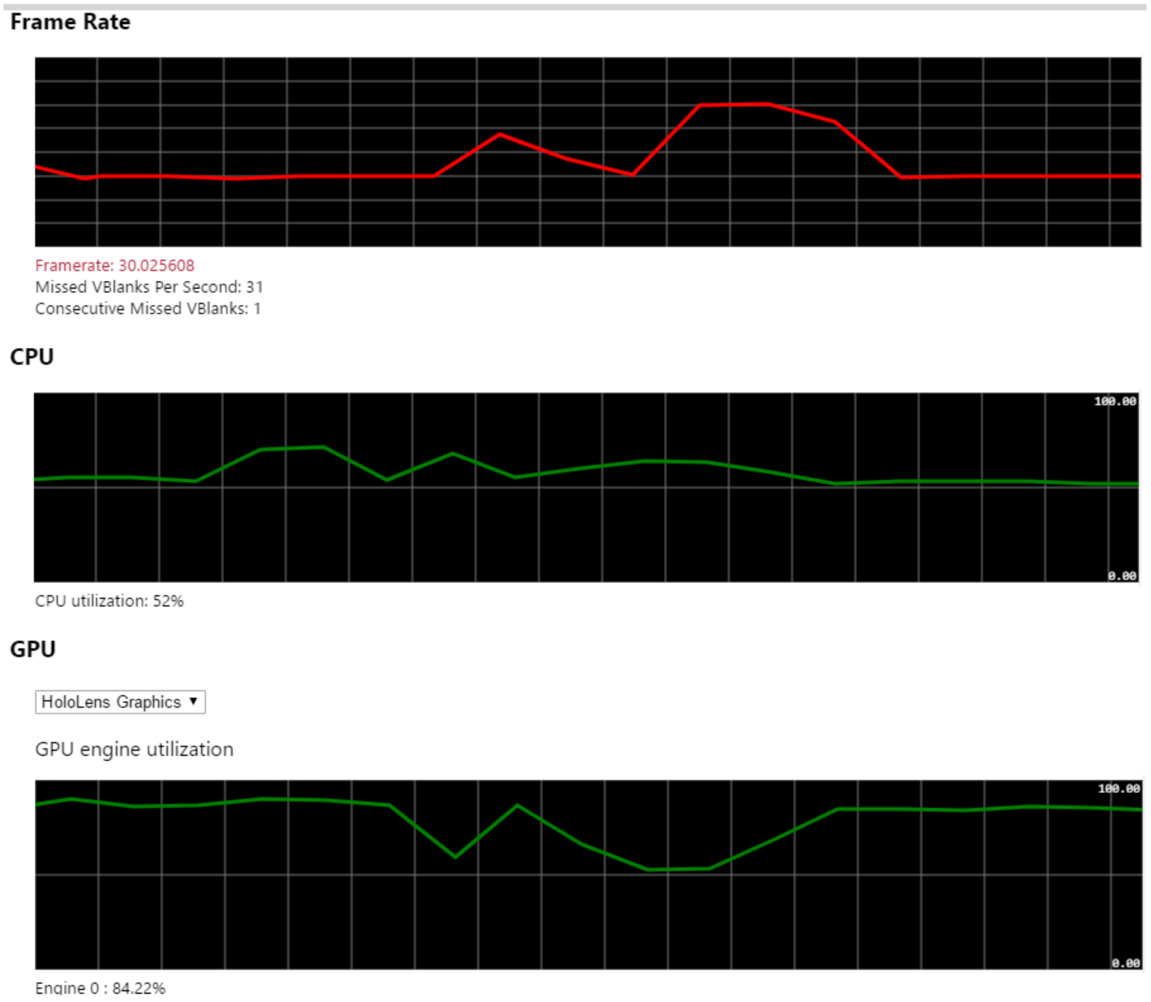
Framerate: 55.405245
Missed VBlanks Per Second: 4
Consecutive Missed VBlanks: 1

CPU utilization: 72%

Engine 0 : 61:94%

Figure 18: HoloLens performance while focusing on the virtual menu

Figure 18 shows the frame rate, CPU and GPU utilisation. Under this performance test, the HoloLens was focusing on the virtual menu object. Focusing on the virtual menu will as expected require more CPU and GPU power than staring out in clear air with the HoloLens. This will therefore adversely affect the frame rate. In the figure the frame rate is around 55 frames per second (FPS), while the CPU is running at 72 percent, and the GPU at around 62 percent.



Framerate: 30.025608
Missed VBlanks Per Second: 31
Consecutive Missed VBlanks: 1

CPU utilization: 52%

Engine 0 : 84.22%

Figure 19: HoloLens performance while focusing on the VuMark

Figure 19 shows the frame rate, CPU and GPU utilisation while focusing on the VuMark with the HoloLens. The tracking option was set to ”MOVING”, explained in section 3.3.1, under this performance test. This means that the HoloLens was constantly detecting the VuMark, and rendering the virtual hand on detection. This resulted in an significant frame drop compared to figure 18. The frame rate in figure 19 was at 30 frames per second. The GPU utilisation while focusing on the VuMark was at 84 percent, which is significantly higher than the 62 percent in figure 18. The CPU utilisation was actually lower while focusing on the VuMark compared to the virtual hand.

It is a well known fact that applications that requires higher computing power affects the FPS negatively. As figure 19 shows, the high GPU utilisation results in a lower FPS. The graphs shows when the GPU suddenly drops, the frame rate was raised accordingly, and vice versa. It seems like the constant rendering of the virtual hand on the VuMark will use more graphical powers than a static virtual menu in the air. On the other hand, figure 18 shows that the virtual menu require more CPU power, but not enough to lower the FPS in the same degree. Higher degree of CPU utilisation while focusing on the virtual menu is probably due to more elements like a cursor with click events etc. on the menu, compared to rendering the virtual hand with fewer components involved.

As mentioned in section 4.1.1, the HoloLens application will prioritise frame rates instead of graphics to get the best user experience. The displays on HoloLens has a refresh rate at 240 times a second, which is divided on four separate colour fields, resulting in an user experience of 60 FPS. In other words, to provide the optimal user experience, applications must run at 60 FPS, which is equivalent to a new image every 16 milliseconds. The frame rate shown in figure 18 is stable at 55 FPS, which is close to the optimal value at 60 FPS, thus provide a good user experience. The frame rate in figure 19 is stable at only half of what is optimal. This is only when the VuMark tracking is set to ”MOVING” by the user. If an object is moving, it is important to update the position as often as possible, and higher FPS is thereby sacrifices.

Another important aspect to frame rate is consistency. Occasionally drops in frame rate is inevitable. However, a hologram with a fluctuating frame rate is very noticeable to the user. Even though the frame rate in the HoloLens application will drop when changing focus from the virtual menu to the virtual hand, the drops will not fluctuate. The frame rate will drop quickly, and then stabilise at a lower FPS. Thus, the users will not experience many unexpected frame drops, which could potentially result in a bad user experience.

6 Discussion

This chapter aims to explain some of the choices and techniques used to make the capillary refill simulation. Further the chapter will explain why this type of research is important. Finally new functionality and improvements of the current functionality will be discussed.

6.1 Choosing hardware and software

To selected the proper hardware and software was a part of the research in the making of an augmented reality capillary refill simulation. Hardware and software was chosen with regard to the tasks in section 1.2. Since Laerdal Medical had a Microsoft HoloLens available, this was the obvious choice. HoloLens is regarded as the leading device in augmented reality. There are many other upcoming augmented reality headsets that will give Microsoft more competition in the future, like Magic Leap [8] and Meta 2 [5], but right now the competition is small.

Microsoft and Unity have been working closely to develop a tool to create applications for HoloLens. They have adapted the Unity game engine to be compatible with the HoloLens. At this time Unity is the only game engine with HoloLens built-in support. Other engines like Unreal Engine will most likely be supported in the future, but the options for Hololens development at this point are limited.

Developing 3D applications by using the Windows Mixed Reality API with DirectX is a possibility, but it would be very time consuming compared to using the Unity engine. If the intention was to only create 2D apps, this could easily be done with C#, XAML and Windows 10 UWP API. However, the tasks for this thesis required 3D modelling, thus Unity was the best choice.

To select a proper method for hand recognition was a more complicated task. This will be explained in more detail in the next section.

6.2 Select tracking method

While the choice of HoloLens and Unity was relatively simple, selecting the best tracking method required more research. The task was to choose a recognition method for a artificial hand that best fulfils the requirements listed as tasks in section 1.2. This section will explain why some methods was discarded before testing, and how other methods failed to give a good enough result.

There are a few option when it comes to object recognition, but not many have an open API with official HoloLens support. One way of detecting an object could be to capture an image with the HoloLens camera, and then send the data to Microsoft Cognitive Service Vision API. With some modification it could then be possible to detect the object, and place a 3D model on the current position. However, this would probably not be fast enough for constant tracking, and thereby not a solution suited for this augmented reality simulation.

Another solution that might be more common in the future is OpenCV. The OpenCV library was made specifically for image processing [18]. It can do complex mathematics on images and still provide good speed for the application. The problem with OpenCV under these circumstances is that it is mainly C/C++. As explained in the previous section, Unity was chosen as the engine, because other engines like Unreal Engine is not supported. The issue is that Unity only support C#, while other engines like Unreal Engine have good support for C/C++. There exists different user made OpenCV beta's for Unity, but non of these are well documented, and some of them are known to be very slow.

Using Spatial Mapping could also be a solution some time in the future. By meshing the room, and then analyse the result for a specific pattern within some constraints could potentially be used to recognise different objects. A case study of an expanded version of Spatial Mapping have been done. It is called "Spatial Understanding" [6], and expands the Spatial Mapping boundaries to recognises objects like walls, floors, chairs and tables. The drawback with this expansion is that it can only recognise flat surfaces, and as explained in chapter 5, Spatial Mapping requires a lot of computing power.

After excluding multiple alternatives, Vuforia SDK was selected as the tracking and detection method. Vuforia have official support for both Unity and HoloLens. Being one of the partners of Microsoft, means that they have access to data from the HoloLens that most developer has not. But Vuforia is still a cross-platform SDK, which means that it is not only working on HoloLens, but also on smart phones and tablets. As a result of being a cross-platform SDK, Vuforia SDK does not utilise the HPU of the HoloLens when recognising images or objects.

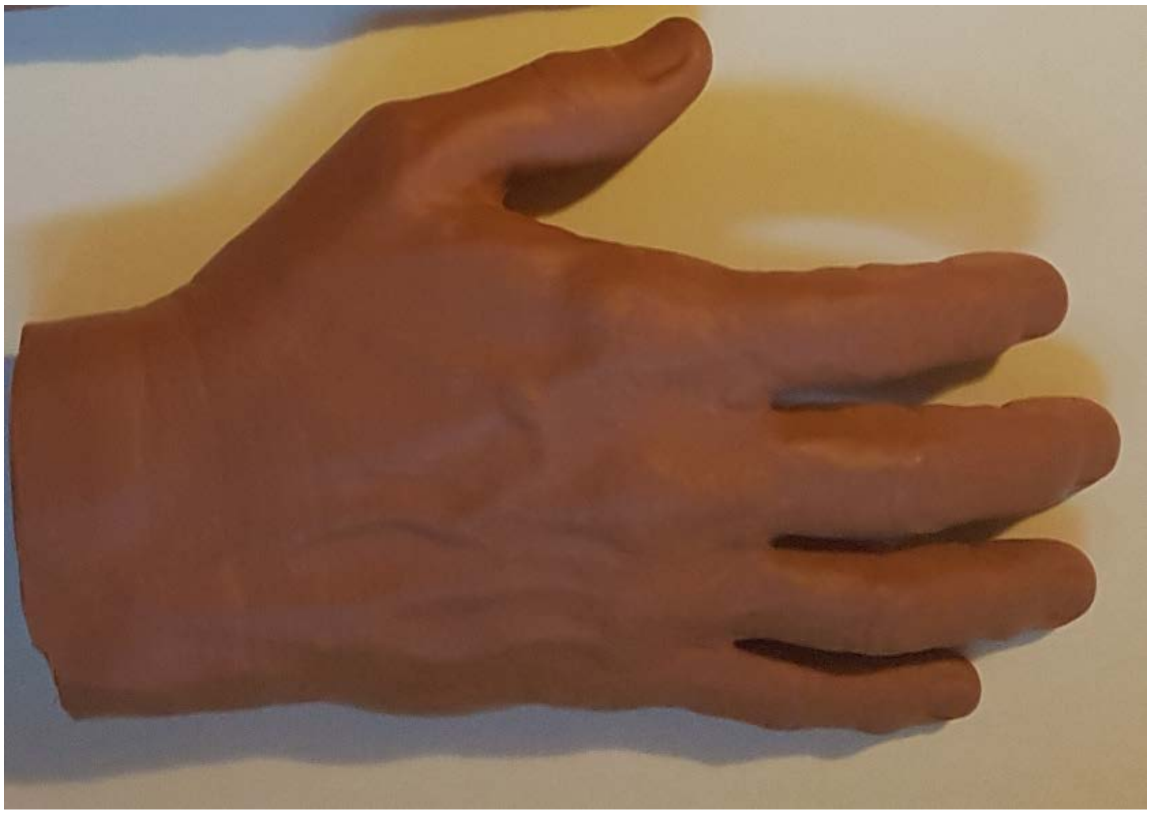


Figure 20: The artificial hand

Figure 20 shows the artificial hand that is used for recognition and tracking. The hand does not have many distinguished points, making it harder to recognise and track it.

As mentioned in section 2.6.1, there are different ways recognise a target by using the Vuforia SDK. In the following sections, some of the recognition methods will be explored.

6.2.1 Image Recognition

In the Vuforia SDK recognition of image targets is supported [25]. Image targets do not need any specific pattern to be detected. The images that the user wants to track must be added to the database in the Vuforia target manager.

After the desired image is added to the target manager, the database can be downloaded and implemented through the Unity engine. When the application is deployed to the HoloLens, the SDK search for the selected image in the user environment. This is done through the optical camera from the HoloLens. To detect and track the image, the SDK will compare naturally features found in the image from the HoloLens feed, by naturally features against the image in the resource database.

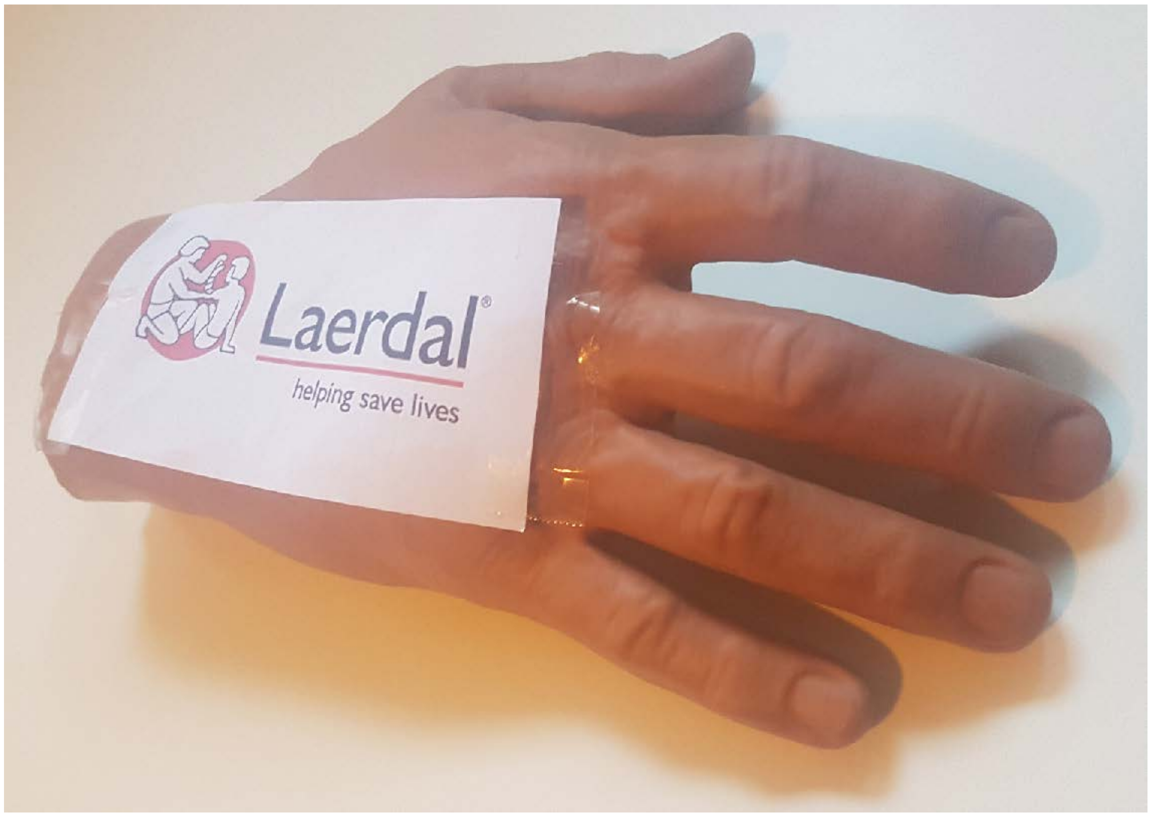
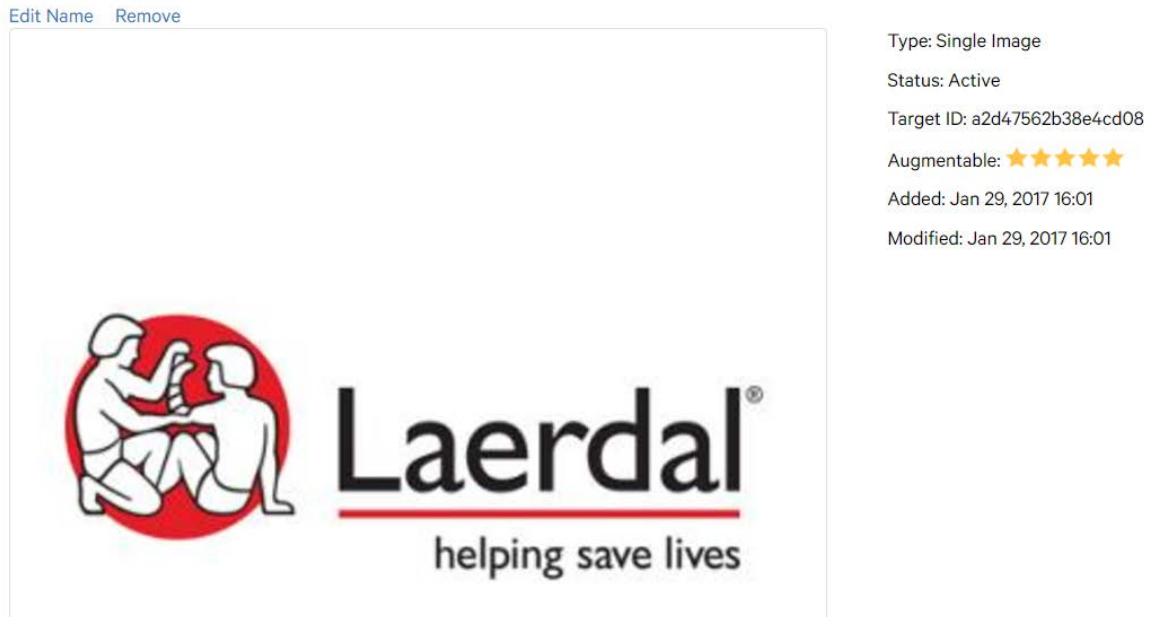


Figure 21: Image attached to the artificial hand

Figure 21 shows how an image was attached to the artificial hand to test image recognition with the Vuforia SDK and HoloLens. The goal was to use the HoloLens to detect the image, and then render a digital 3D model of the hand. Then the image, which is attached to the artificial hand, would not be visible to the user.

The main reason why this method was not chosen, was because HoloLens struggled to recognise the image. Vuforia image recognition with HoloLens did only manage to detect the target when the printed image was significantly larger than ideal, and when light conditions was very good.



Type: Single Image

Status: Active

Target ID: a2d47562b38e4cd08

Augmentable: ★★★★★

Added: Jan 29, 2017 16:01

Modified: Jan 29, 2017 16:01

Figure 22: Image Target rating

Different images was tested, and they all showed full score in the detection scale provided by Vuforia. Figure 22 shows how the rating is displayed in the Vuforia target manager. Adjusting and optimising the texture settings in Unity was also tested, but the tracking result still wasn't sufficient enough to use as a tracking method. For an image detection to be used as a tracking method, it would have to be small enough to fit on the artificial hand, and the light conditions cannot be too restricted.

6.2.2 Object Recognition

The Vuforia SDK also have a feature for object detection and tracking of 3D objects, which was also tested [26].

Like image target, object target also have to be added to Vuforia target manager. They must be in a Vuforia specific format, which only can be created by using an Android scanner application. The scanner can be downloaded from the Vuforia developer portal, and produces an object data output file. The scanner uses the Android camera to scan the object in an specific environment, which we will not go further into in this report.

After the object target was added to the database in the target manager, the database was implemented in the unity project. Then the application was tested on the HoloLens, and the result showed that HoloLens was struggling to detect the object. The reason for this was that the Android object scanner did not find enough distinguished points on the artificial hand, which is displayed in figure 20. Due few distinctive features, the HoloLens did only manage to recognise the artificial hand in a very close range, from some specific angles.

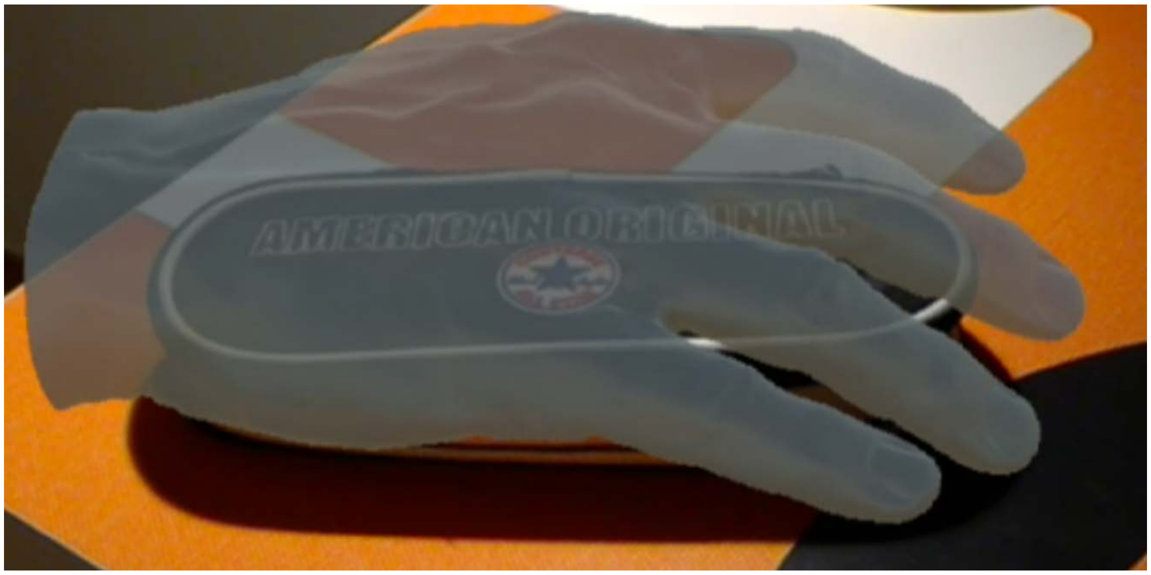


Figure 23: 3D hand following pencil case

Further testing was done on objects with more unique characteristics, such as a pencil case showed in figure 23. In this case the object detection was performing much better. To optimise the tracking of the artificial hand, about 20 small pieces of tape was stuck to the hand to make more unique tracking points for the HoloLens camera to recognise.

Manually creating more unique characteristics for the camera to recognise by adding pieces of tape did not work as hoped. The detection was still performing much worse than required for this setup. The artificial hand was still not characteristic enough for the camera to detect it. The hand showed in figure 20 is the product that Laerdal Medical produces and uses as skin on their manikins. If this hand shall be used as a marker for the HoloLens to detect, it would be unfavourable if the users of this product have to make too many adaptions of the artificial hand. For this reason, the last relevant detection method from Vuforia was tested, namely VuMark.

6.2.3 VuMark

Neither Vuforia image target or object recognition met the requirements for detecting the artificial hand. Object targets discussed in 6.2.2 do not work on an object that has few characteristic features. And as explained in 6.2.1, even image target in perfect condition on the Vuforia image database rating scale, used in decent light condition, was too difficult to recognise.

VuMark is described by Vuforia as the next generation bar code [28]. They present it as a bar code, which allows for customisation and brand-conscious design, while still acting as an AR target used to encode data. The time it takes for the camera to recognise VuMark's is superior compared to the other recognition methods by Vuforia. On moving objects it is important that the recognition occurs fast, since it must constantly update the position and orientation during this period. Image target and object recognition did not always manage to detect changes, and when they did, the time lag was significantly greater. Due to considerably improvements with VuMark compared to the other recognition methods that was tested, VuMark was chosen to be the recognition method used in this thesis.

6.3 Coordinating eye position with Vuforia Object Detection

VuMark was selected as the detection method in this thesis. There are advantages of both VuMark and object detection. In some situations, object detection might be the preferred method. Avoiding too many alterations of the artificial hand made object detection unfavourable for this project setup, as explained in the previous section.

Nevertheless, for other detection and tracking situations, there might be easier to detect an object instead of a VuMark. It could also be favourable to not have a marker such as an image or a VuMark on the object used for detection. An important note in these cases is that object detection by Vuforia does not work as good with

HoloLens, as it does with smartphones. This might, and probably will change in the feature, and even though the Unity integration makes it relative simple to add object detection to HoloLens, it is not optimised for holographic devices yet.

The main reason for this is that the holograms are displayed on two separate lenses, one for each eye. While there is a camera in the middle that Vuforia utilises in all their detection methods.

This lead to one of the main issues in the start of this research being the positioning of 3D objects. Vuforia uses the camera to track a marker or an object, and will then notify the HoloLens to show a virtual object where the marker or object is located. With object targets, the targets would be recognised fine, but the holograms appearing would not align correctly on the object. Placing the hologram at the 0,0,0 position in the local marker coordinate system should display the virtual element on top of the object. The problem was that the hologram appeared behind the target, and the position of the hologram would be different seen from the left and the right eye.

One way to prevent the virtual object to appear in space behind the object target, is to transform the object positions of a competent in the Vuforia detection script.

Listing 2: Correcting position offset

```
//Restore local scale on a new tracking
//Adjust local scale distance
renderComponent.transform.localScale = initScale[renderComponent];
renderComponent.transform.localScale =
   renderComponent.transform.localScale * offsetScale;

//Restore local position on a new tracking
//Correct local position z-axis offset
//Correct pivot position
renderComponent.transform.localPosition = initPos[renderComponent];
renderComponent.transform.localPosition =
   renderComponent.transform.localPosition +
   initPos[renderComponent] * offsetScale;
renderComponent.transform.localPosition =
   renderComponent.transform.localPosition + new Vector3(0, 0,
   0.03f);
```

```
//Set hit pont of the position
renderComponent.transform.position = hitInfo.point;
```

Listing 2 shows how to use the Unity transform API to manipulate the position of a component. This code will run on while the instance is successfully tracked. The variable "offsetScale" is the scale percentage, and this variable will be different depending on the size of the object used for detection.

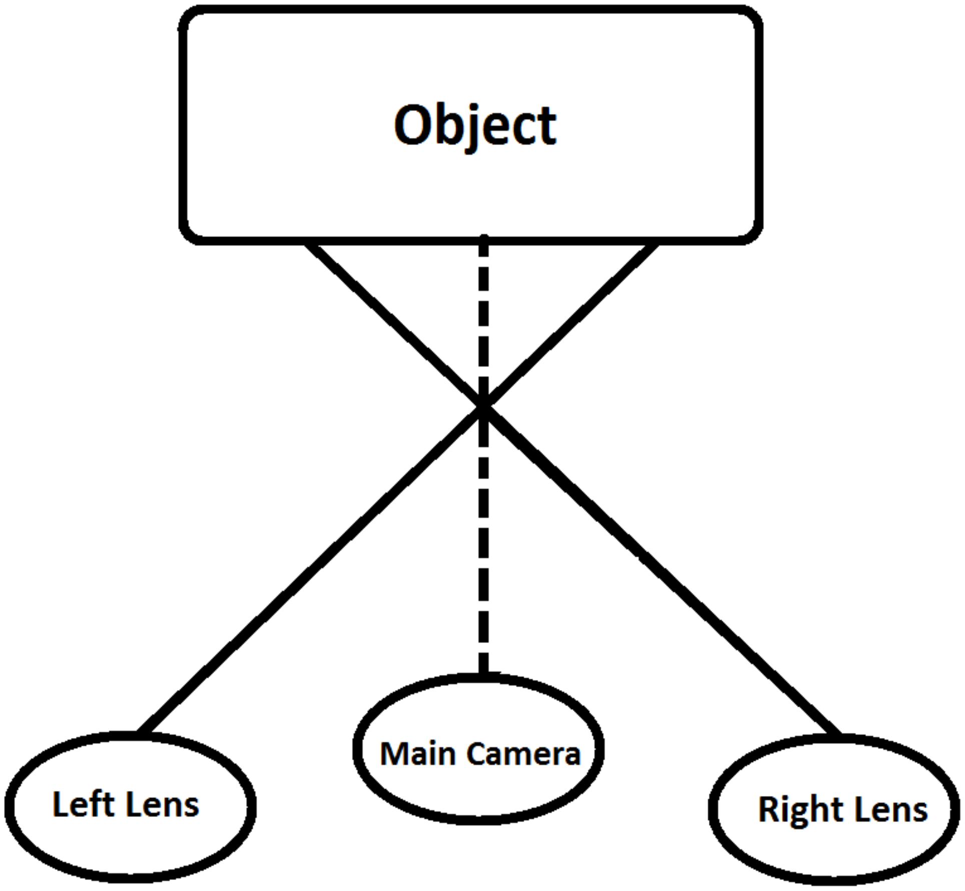


Figure 24: Cross-Eyed Viewing

After correcting for the target appearing behind the object, further adjustment needed to be done to correct the difference from the left and right eye angle. Figure

24 shows how the right eye is looking on the left side of the object, and the left eye would look at the right side of the object. This is not a problem for the main camera, which is the camera used for detection.

To align the hologram correctly on both eyes with Vuforia object detection, two new camera objects with HoloLens configurations, see section 4.1.1, was added to the project in Unity. The camera in Unity is a camera Unity GameObject, and must not be confused with the physical built-in HoloLens camera. The main camera GameObject was still kept for tracking the object, while the two new camera, one for each eye, was added to align the angle properly. This thesis will no go further into how this was done, but a combination of Unity camera depth and layers was used to only see the virtual hand from these two new GameObjects. The main HoloLens camera object was still used to display the menu, cursor etc.

According to the result in this thesis, object detection will not position virtual object correctly on HoloLens without some tweaks and modifications. However, the cross-eyed viewing showed in figure 24 is corrected with VuMark recognition for HoloLens. An important note is that the VuMark have to be defined in meters. This can be confusing according to the tutorial provide by Vuforia, which says the following: “Define the width of your VuMark in scene units. We recommend defining this to correspond with the printed VuMark’s width in whichever unit of measurement you prefer” [24]. Nevertheless, with default Unity setup, Unity uses meters as scene units. Thus, the VuMark have to be in meters to prevent the same position issues that follow object detection.

6.4 Vuforia Extended Tracking

Instead of utilising the HoloLens HPU under tracking, Vuforia has made it possible to track the target from the HoloLens using a features called Extended Tracking [23]. Extended Tracking passes the pose of the target to the HoloLens Spatial Mapping engine. This means that Vuforia uses their own SDK to recognise the target, but once the target is detected, the role of keeping track of the target is done by the HoloLens. The pose of the target is then transformed into the HoloLens spatial mapping coordinate space. The tracking by Vuforia is thereby deactivated. The target will then exists in both the HoloLens and Vuforia spatial coordinate system, but not simultaneously.

Extended Tracking uses the environment around to predict the targets position. In theory, it is possible to keep track of the target even if it is out of view. This is made possible by using the map around the target to assume its position and orientation. Extended Tracking is assuming that both the environment around the user and the targets used for tracking is fairly static. The quality of the mapping of the surroundings works better with a detailed and feature-rich environment.

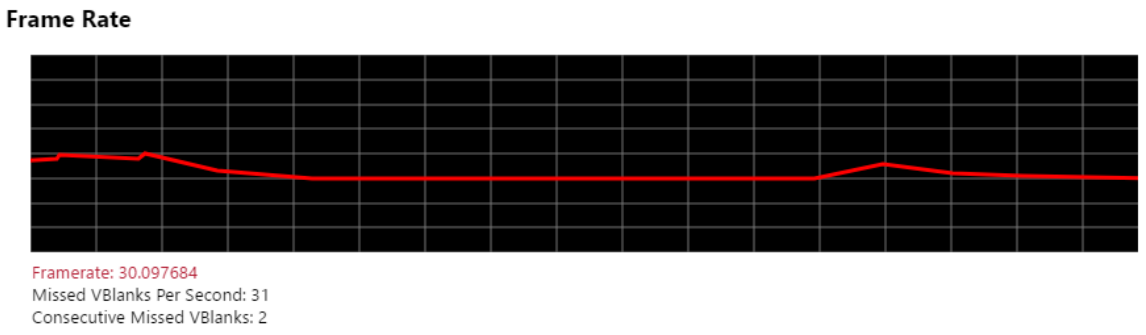
However, Extended Tracking does have some problems on HoloLens. Due to these issues, there are two options for VuMark recognition. One that are best suited for static object detection, and another recognition option for moving objects. The reason for this is that Extended Tracking by Vuforia is performing well when the objects are static. When Extended Tracking is activated, the virtual 3D hand appears to be more stable. However, if the object is moving when Extended Tracking is activated, the time interval between each time the virtual hand is updating its position is too slow. Then it is best to disable Extended Tracking to continuously recognise the VuMark's postilion as often as possible. Chapter 5 shows that the frame rate on moving objects will drop to around 30 FPS. Higher FPS would be and advantage, since there are judder tendencies while tracking a moving object. Judder is the uneven visualisation of a hologram caused by double images [9]. The double images comes because the application is running at 30 FPS which is half of the optimal 60 FPS. The judder tendencies are less visible while Extended Tracking is activated, thus the virtual hand will appear more stable.

6.5 Potential for Occlusion

The task description in section 1.2 explains that one of the issues in this thesis is to occlude the virtual rendered 3D hand behind real world objects. Occlusion in 3D design is defined as the effect of one 3D object blocking another object from view. Since a marker called VuMark is used to track the artificial hand in this, the virtual object will not be rendered when the VuMark is out of view. However, the task description says that if a part of the artificial hand is blocked, then the part of the 3D model representing the blocked areas should not be rendered. This means that if the artificial hand is partly blocked by for example a real hand, then this part of the virtual 3D hand on HoloLens should not be rendered. Partly occluding objects was a goal that was not achieved in this thesis. Nevertheless, some research was done in this field, and this section will explain the potential of partly occluding virtual objects behind real objects.

Due to time constraints only one method for partly occlusion was tried in this thesis. The method used was Spatial Mapping, which is briefly explained in 2.3.2. Microsoft says that occlusion is one of the primary uses of Spatial Mapping [12].

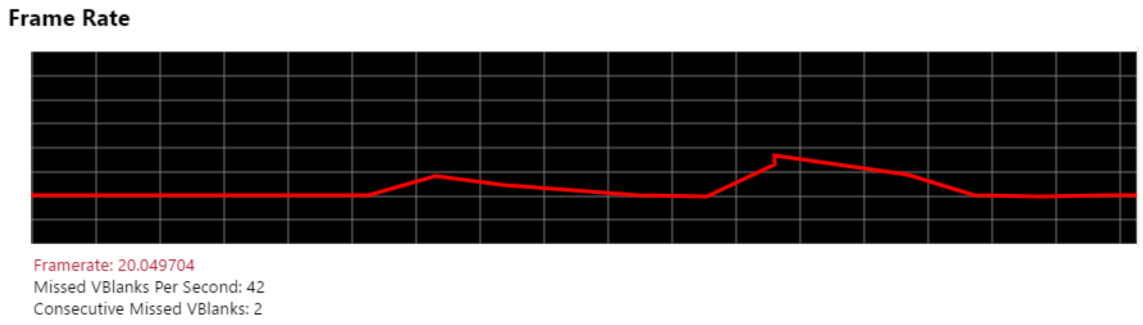
A Spatial Mapping component was implemented from the HoloToolkit and added to the Unity project. By adding a script to the Spatial Mapping component, the users can specify how often the environment is going to be scanned, how many triangles per cubic meter the rendering mesh shall create, and what material that are going to be used in environment scanning. For virtual object to occlude behind real world objects, which can be both static and moving, the update interval of the scanning have to be continuous. Different updating intervals was tried to get the best result. When it comes to how many triangles per cubic meter the mesh shall render on a new scan, this will also have to be fairly high. To detect "smaller" object, such as a moving hand blocking the virtual objects, will require a highly detailed mesh. Some different materials was also tried, but the standard material that HoloToolkit has included is very well optimised for Spatial Mapping. The standard material is called "wireframe", and provides a detailed representation of the surrounding, and was thereby not changed.



Framerate: 30.097684
Missed VBlanks Per Second: 31
Consecutive Missed VBlanks: 2

Figure 25: HoloLens frame rate with Spatial Mapping enabled. Focusing on the virtual menu.

In chapter 5 the performance of the HoloLens while running the HoloLens application is analysed. Figure 25 shows correspondingly frame rate while looking at the same menu as used for performance testing in chapter 5.



Framerate: 20.049704
Missed VBlanks Per Second: 42
Consecutive Missed VBlanks: 2

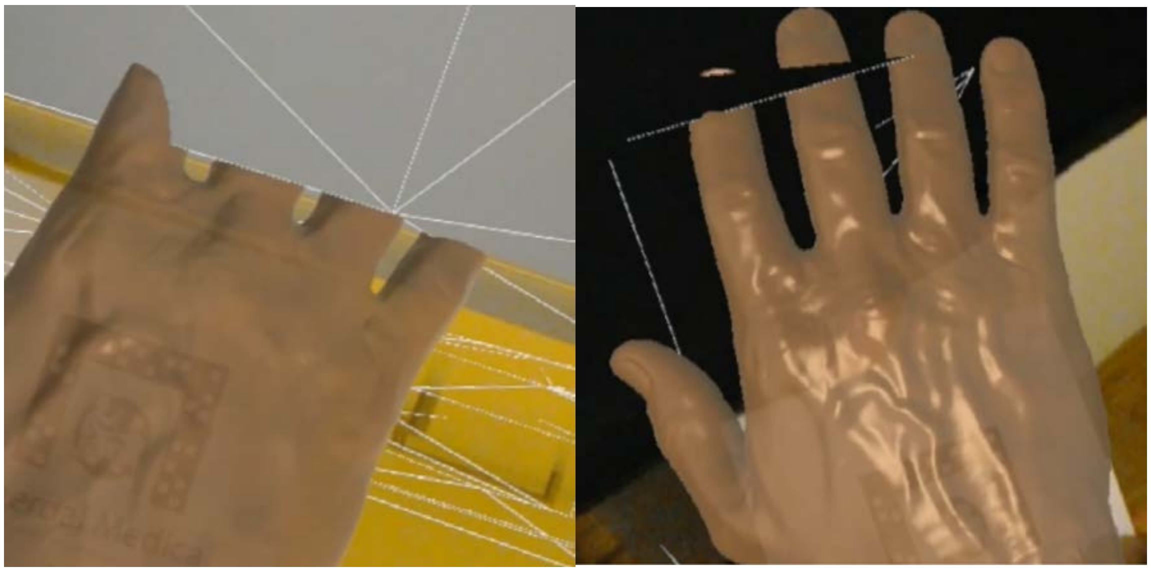
Figure 26: HoloLens frame rate with Spatial Mapping enabled. Focusing on the VuMark.

Figure 26 shows correspondingly frame rate as used to analyse performance in chapter 5, while looking at the VuMark. The frame rate in both figure 25 and 26 shows that Spatial Mapping have a high demand of computing power. Even though the FPS without Spatial Mapping can be justified, the FPS on this application with Spatial Mapping activated is more critical. The HoloLens has a passive cooling system, and if the internal temperatures spikes too high, the application running at that time will shutdown. Creating an app that uses to much computing powers, and thereby rising the temperatures, will not only create an unpleasant user experience, but also risking a shutdown.

Spatial Mapping (SM)	Focus Menu	Focus Hand
SM Off	55	30
SM On	30	20

Table 1: Comparison of frame rate with and without Spatial Mapping

Table 1 compares the result with and without Spatial Mapping to see how it affects the frame rate. The result shows as mentioned that Spatial Mapping requires a lot of computing power, and thereby lowering the FPS. The main goal of implementing Spatial Mapping in this HoloLens applications is as mentioned to occlude object while focusing on the VuMark. The table shows that while focusing at the VuMark and rendering the virtual 3D hand, the frame rate falls to 20 FPS.



(a) Fast Spatial Mapping (b) Slow Spatial Mapping

Figure 27: Occlusion under desks

Figure 27 shows the occlusion result of the Spatial Mapping test. In figure 27a, the hand is partly under a white desk, and in 27b, the hand is partly under a black desk. All the white stripes is the triangle mesh, which is set to be visual in this illustration. This is done to better display how the mesh is drawn to get a representation of the real-world surfaces. The figure shows how the virtual hand is partly occluded below a desk. Although none of the images in the figure showed perfect results, it was clearly that the HoloLens managed to render higher details of triangles meshes while the scanning intervals was slow. Faster occlusion would be beneficial to faster occlude virtual object behind real life object. However, under faster scanning intervals, the HoloLens barely managed to render any meshes, as seen in figure 27b. Constantly requesting new scans had a major impact on the performance, resulting in an unpleasant user experience.

A case study of how to improve Spatial Mapping capability is already done by Evertt [6]. In the study techniques to improve performance and visualisation is showed. It refers to a technique called plane finding, used to transform Spatial Mapping data into planes. And then classify those planes into floors, walls or ceiling. The problem with this study was that it could only classify flat surfaces, and not surfaces that was not flat. To also detect and classify moving object would require a lot more research.

Spatial Mapping could not provide the occlusion results needed for this thesis. Optimising the Spatial Mapping further could be one way to get the desired result. Another way of getting better results can be through more computing power. In the future, a headset device such as HoloLens will have more computing power available. In the present, a remote computer, which communicates with HoloLens wirelessly can be one of the solutions. There is already an application on Windows Store named "Holographic Remoting Player" that can stream holographic content from PC to HoloLens, using Wi-Fi.

6.6 Significance

Testing real life situations through simulations is a widely used technique in medical training. There are multiple article about the effect of medical simulation [30]. In 2010 Lateef released an article [7], explaining how simulation-based training is a technique to replace and amplify real experiences with guided ones. The articles explains how early adopters of virtual reality learning will see the potential of investing time and energy in a simulated virtual environment. Lateef concludes that the cost-effectiveness of expensive simulation-based training should be examined further, and that health care system thereby could be viewed as more accountable and ethical. The article was released in 2010, when virtual reality showed great potential. A few years later, augmented reality is also showing great potential, and it could thereby offer the same economical advantages as virtual realty.

The main benefit of augmented reality compared to virtual realty, is that the users can interact with real world elements while interacting with virtual elements. This could potentially create a more realistic environment, and only some of the elements need to be rendered on an augmented reality device, while other elements could be real items. One of the benefits of virtual reality, is that it is able to transpose the users, by bringing them some place else, like for example a hospital. Then the users can be exactly where they want to be, but this might require more computing power.

Both techniques have potential to make future simulations more immersive and realistic in the future.

6.7 Further Work

The section provides some suggestions for further work that could optimise and improve the capillary refill simulation setup. There is a wide variety of possibilities of features that can be implemented.

6.7.1 Occlusion

Section 6.5 is discussing the potential for occlusion. This might also be an area to explore further. It can be done by optimising the Spatial Mapping capabilities. And also by investigating the possibility for another computer to do some of the heavy computations. If external computer is not desired, cloud computing might be another alternative.

6.7.2 Bluetooth module on Arduino

Optimising the project setup can be done by eliminating the need to an external computer. To do this, while still having a sensor module inside the artificial hand, Bluetooth is need on the Arduino. Arduino does not have Bluetooth by default, but it is possible to connect a Bluetooth module. It is worth mentioning that Bluetooth on Windows 10 (running on HoloLens) is known to be unreliable. To get a reliable Bluetooth connection in this project, the PC and the HoloLens had to be ”paired” in advance. Even if Bluetooth is added directly to the Arduino, an UWP application could still be useful to for example spectate the HoloLens user under simulation.

6.7.3 Visuals

Both optimising and adding more visuals could improve the impression of the HoloLens application. Currently the visual graphics are mostly a proof of concept, and colours and details have not been highly prioritised. Adding more and better visuals could create an more immersive experience for the user, making the simulation more realistic.

Other visual improvements could be to add an advanced animation. For example by adding a talking animation of a person, explaining how capillary refill works. Then the users would not require an external sensor to explain them what to do.

The possibilities within visuals are almost unlimited.

6.7.4 New simulations

Eventually an expansion to other simulations would be the main goal. To have more simulations available on a single device, by using the same manikin would be both economically and efficient.

7 Conclusion

This thesis presented an approach to simulate capillary refill with augmented reality. Except for general programming skills, there are no prior knowledge needed to implemented recognition capabilities on a Microsoft HoloLens. However, expertise within image processing would be helpful to find other solutions than those presented in this thesis.

Vuforia SDK provides a good approach to image tracking through their bar code like tags called VuMark. Other recognition methods by Vuforia require more adjustments to make them a viable solution.

Using the cameras on the HoloLens to detect when the user touch the fingertip on the artificial hand seemed to be very complicated, or maybe even an impossible task. Therefor a sensor was placed inside the artificial hand to detect finger press from the user.

As expected, the hardware limitations of HoloLens brought some limitations when it comes to performance and field of view. Especially Spatial Mapping, which is one of the new revolutionary core technologies that comes with HoloLens, had a major impact on the performance. With more computing power available, Spatial Mapping could potentially be used to occlude real life elements behind virtual object through an augmented reality device like HoloLens.

The results shows that simulating capillary refill with an augmented reality headset such as HoloLens is possible with some tweaks and modifications. Recognising an artificial hand with a marker, such as a VuMark from the Vuforia library, makes it possible to track the artificial hand and render a virtual 3D model over the marker attached to the artificial hand. The 3D model is changing its nail colour, and thereby simulating capillary refill, when the sensor inside the artificial hand detects user interaction. Due to performance issues, occluding objects behind the virtual object did not provide good enough results to be implemented final solution.

References

[1] Adafruit. Using an fsr. https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr/. [Accessed 15-April-2017].

[2] Anonymous. Capillary nail refill test. https://medlineplus.gov/ency/article/003394.htm/. [Accessed 02-February-2017].

[3] Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators and virtual environments*, 6(4):355–385, 1997.

[4] Yusuf Abdullahi Badamasi. The working principle of an arduino. In *Electronics, Computer and Computation (ICECCO), 2014 11th International Conference on*, pages 1–4. IEEE, 2014.

[5] Meta Company. Meta. https://www.metavision.com/. [Accessed 12-May-2017].

[6] Jeff Evertt. Case study - expanding the spatial mapping capabilities of hololens. https://developer.microsoft.com/en-us/windows/mixed-reality/case_study_-_expanding_the_spatial_mapping_capabilities_of_hololens. [Accessed 01-February-2017].

[7] Fatimah Lateef et al. Simulation-based learning: Just like the real thing. *Journal of emergencies, trauma, and shock*, 3(4):348, 2010.

[8] Magic Leap. Magic leap. https://www.magicleap.com/. [Accessed 13-May-2017].

[9] Microsoft. Hologram stability. https://developer.microsoft.com/en-us/windows/mixed-reality/hologram_stability. [Accessed 03-April-2017].

[10] Microsoft. Hololens hardware details. https://developer.microsoft.com/en-us/windows/mixed-reality/hololens_hardware_details. [Accessed 15-March-2017].

[11] Microsoft. Microsoft hololens. https://www.microsoft.com/microsoft-hololens/en-us. [Accessed 03-February-2017].

[12] Microsoft. Spatial mapping. https://developer.microsoft.com/en-us/windows/holographic/spatial_mapping. [Accessed 03-February-2017].

[13] Microsoft. What's a universal windows platform (uwp) app? `https://docs.microsoft.com/nb-no/windows/uwp/get-started/whats-a-uwp`. [Accessed 25-May-2017].

[14] Microsoft. Holotoolkit-unity. `https://github.com/Microsoft/HoloToolkit-Unity`, 2017. Accessed 15-February-2017.

[15] Amelia Pickard, Walter Karlen, and J Mark Ansermino. Capillary refill time: is it still a useful clinical sign? *Anesthesia & Analgesia*, 113(1):120–123, 2011.

[16] KIRI ROWAN. What is adobe illustrator used for? understanding vector images. `https://blog.udemy.com/what-is-adobe-illustrator-used-for/`. [Accessed 12-February-2017].

[17] Gianni Saporiti. Force sensitive resistors. `https://developer.mbed.org/users/saporiti/notebook/force-sensitive-resistors/`. [Accessed 05-May-2017].

[18] Utkarsh Sinha. Why opencv? `http://aishack.in/tutorials/opencv/`. [Accessed 29-January-2017].

[19] Mike Taulty. Windows 10, uwp, hololens & a simple two-way socket library. `https://mtaulty.com/2017/02/21/windows-10-uwp-hololens-a-simple-two-way-socket-library/`. [Accessed 10-March-2017].

[20] Mike Taulty. Simpleuwptwowaycomms. `https://github.com/mtaulty/SimpleUwpTwoWayComms`, 2017. Accessed 27-February-2017.

[21] Vuforia. Designing a vumark in adobe illustrator. `https://library.vuforia.com/articles/Solution/Designing-a-VuMark-in-Adobe-Illustrator`. [Accessed 10-May-2017].

[22] Vuforia. Developing vuforia apps for hololens. `https://library.vuforia.com/articles/Training/Developing-Vuforia-Apps-for-HoloLens`. [Accessed 13-February-2017].

[23] Vuforia. Extended tracking. `https://library.vuforia.com/articles/Training/Extended-Tracking`. [Accessed 13-February-2017].

[24] Vuforia. How to work with vumark databases. https://library.vuforia.com/articles/Solution/How-To-Work-with-VuMark-Databases. [Accessed 05-May-2017].

[25] Vuforia. Image targets. https://library.vuforia.com/articles/Training/Image-Target-Guide. [Accessed 30-January-2017].

[26] Vuforia. Object recognition. https://library.vuforia.com/articles/Training/Object-Recognition. [Accessed 25-February-2017].

[27] Vuforia. Vufoira developer portal. https://developer.vuforia.com/. [Accessed 20-April-2017].

[28] Vuforia. Vumark. https://library.vuforia.com/articles/Training/VuMark. [Accessed 05-May-2017].

[29] Vuforia. Vumark design guide. https://library.vuforia.com/articles/Training/VuMark-Design-Guide. [Accessed 10-May-2017].

[30] Hsin-Kai Wu, Silvia Wen-Yu Lee, Hsin-Yi Chang, and Jyh-Chong Liang. Current status, opportunities and challenges of augmented reality in education. *Computers & Education*, 62:41–49, 2013.

A Appendix

A.1 Demonstration video

This is a short demonstration video of capillary refill using HoloLens. The video demonstrates how to connect a computer with a HoloLens, and how the fingernail will change colour when pressure is added to the fingertip.

The video is available at: `https://youtu.be/p4IDE5TMSQM`

A.2 Source Code

The source code is embedded in the PDF document.