Linköping University | Department of Computer and Information Science
Bachelor's thesis, 16 ECTS | Datateknik
2021 | LIU-IDA/LITH-EX-G--21/014--SE

A Model-Based Approach to Hands Overlay for Augmented Reality

Fredrik Adolfsson

Supervisor : Zeinab Ganjei
Examiner : Mikael Asplund

External supervisor : Alojz Milicevic

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

Augmented Reality is a technology where the user sees the environment mixed with a virtual reality containing things such as text, animations, pictures, and videos. Remote guidance is a sub-field of Augmented Reality where guidance is given remotely to identify and solve problems without being there in person. Using hands overlay, the guide can use his or her hand to point and show gestures in real-time. To do this one needs to track the hands and create a video stream that represents them. The video stream of the hands is then overlaid on top of the video from the individual getting help. A solution currently used in the industry is to use image segmentation, which is done by segmenting an image to foreground and background to decide what to include. This requires distinct differences between the pixels that should be included and the ones that should be discarded to work correctly. This thesis instead investigates a model-based approach to hand tracking, where one tracks points of interest on the hands to build a 3D model of them. A model-based solution is based on sensor data, meaning that it would not have the limitations that image segmentation has. A prototype is developed and integrated into the existing solution. The hand modeling is done in a Unity application and then transferred into the existing application. The results show that there is a clear but not too significant overhead, so it can run on a normal computer. The prototype works as a proof of concept and shows the potential of a model-based approach.

Acknowledgments

Firstly, I would like to thank my supervisor Zeinab Ganjei for the constructive comments, help, and support throughout this thesis. It has been greatly appreciated. Secondly, I would like to thank my examiner Mikael Asplund for his input. I would also like to express my gratitude towards XMReality for the opportunity to do this thesis. A special thanks for the input and help from the people at the company, Alojz Milicevic, Lillemor Blom and Alexander Widerberg.

Contents

List of Figures

1 Introduction

Augmented Reality is a technology where the user sees the environment mixed with a virtual reality containing things such as text, animations, pictures, and videos. One of the most famous examples of AR is in the game Pokemon Go [1], where Pokemon's are displayed as if they were in the real world. Another famous example is Snapchat [2] and the way they use their filters to combine the real world image with virtual elements. Remote guidance is a sub-field of Augmented Reality where guidance is given remotely to identify and solve problems without being there in person. Remote guidance can be done on many devices such as smartphones, desktops, or using AR-glasses. This is usually done by having a video session with one person receiving guidance and one or more persons guiding. The receiver can, with the help of video, let the other person see what he or she sees. The person guiding can then show where to focus, instruct what to do with the help of their voice, or add things such as pointers, text, or hands overlay. When using hands overlay, the guide can use his or her hand to point and show gestures in real-time overlaid on top of the video from the individual getting help.

Today an approach to hands overlay actively used in the industry is made based on image segmentation and the works of Poole[1]. It is done by segmenting an image to foreground, the part of the image that shall be included, and background, the part of the image that shall be discarded. The segmentation is done by analyzing pixels and grouping pixels with similar values together. Probability is then used to determine if the pixels should be included. So one tries to find the hand in an image and crop it out. This thesis instead investigates a model-based approach to hand tracking. A model-based approach tracks the position and rotation of points of interest of the hand, such as fingers, palm, and joints. That information is then used to build a model of the hand. A leap motion controller is used to do this. It is a lightweight commercial product which consists of 2 image sensors and three infrared LEDs [3]. Snapshots of those models are then used to implement hands overlay. The leap motion technology has been studied based on great demand in different fields such as stroke rehabilitation[3], sign language recognition[5], and robotic arm controlling [2]. It has also been used in entertainment, 3D modeling, graphics, and manufacturing, while it still got a lot of unexplored potential[8].

[1] https://www.pokemongo.com/
[2] https://www.snapchat.com/
[3] http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/

1.1 Motivation

The demand for doing tasks remotely and working from home keeps growing. With the current events of the corona pandemic, this is more clear than ever. Remote guidance and hands overlay can be a powerful tool to enable this. To solve problems over distance, one needs to be able to identify the problem and give instructions. A reason this is often done in person is the need to show things using hand gestures. Hands overlay makes it possible to show things using hand gestures remotely.

Image segmentation is based on analyzing pixel values or the colors of pixels. The hand is then cropped out of the image being analyzed. The current implementation of this requires distinct differences between the pixels that should be included and the ones that should be discarded to work correctly. Therefore, it works best on a uniform background with a solid color that is different from the hand. Since the alternative solution in this thesis is instead based on sensor data, it would not have these limitations. Using a model instead of a real hand also enables the perspective from which it is seen to be modified effortlessly. This makes it possible to for example see the hand in first-person instead of from the perspective of the camera. Having a first-person perspective is something that can make guiding more intuitive.

1.2 Aim

Hands overlay is done in real-time. For this to be useful, the delay between the real world hands and the hands seen in the application must be sufficiently small. The current implementation uses 25 frames per second and works well, making it a good aim. The computational power should also ideally be low enough, so it can run on the computers currently used in the industry.

The objective of this thesis is to create a prototype that uses a leap motion controller to model a hand. This prototype is then integrated into the XM-application, which is described in Section 2.2, and hands overlay is implemented with the help of it. The prototype and the evaluation of it should be helpful to decide if it is a good idea to develop this further and potentially take it to production.

1.3 Research Questions

This section contains the research questions that this thesis aims to answer.

- Can hands overlay be done using a leap motion camera and the leap motion software?
- How are the frame rate and computational needs compared to the old solution?
- How can it be integrated into the XM-application?

1.4 Delimitations

Hands overlay is only implemented for Windows OS. The thesis aims only for a proof of concept and does not need to be fully integrated into the XM-application.

2 Theory

The following chapter describes the necessary information needed to solve the problem and to help design an appropriate method. It first explains how hand tracking is done and then describe the programming languages and tools needed to implement a prototype. The programming languages are briefly explained to allow the user to understand the structure of the application.

2.1 Related work

Although there exists both studies in hand tracking and remote guidance, the two combined is not very explored. In an article by Mather et al [4] both hands overlay and remote guidance is studied together from a user experience perspective. The study found that subjects got a positive experience and that the guidance was as helpful as it would have been if the teacher was there in person. This further show why this thesis could be useful, by adding to the technical aspects. In a article by Oda et al [6] they describes a system for remote guidance where an expert is guiding a subject to do complex tasks. The expert shows what to do by manipulating 3D replicas of real world objects. They compare a 3D augmented reality system to a 2D drawing system and found that subjects were able to preform the task faster when getting guidance with the augmented reality system. The system implemented in their study is similar to what was done in this thesis, but focusing on modeling hands instead of real world objects.

2.2 Current Implementation of Hands Overlay

XMReality AB[1] develops and sells a remote guidance solution that allows users to understand and solve problems over vast distances quickly. The solution that XMreality offers, henceforth called XM-application, works on many platforms. The Windows version of this application was used as a starting point, and then modified and extending in this thesis. It can be used both in iOS, Android, Windows-desktop, and on the web. The flowchart for the app can be seen in Figure 2.2. The XM-application has three layers. To ease the development for cross-platform, they have a library written in C++ that executes most heavy calculations

[1] https://xmreality.com

such as generating, merging and manipulating frames. The library also handles transfer of frames from one device to another. The library is the same regardless of platform. Connections between users are established with the help of a server, using WebRTC[2]. All video and sound streams are sent directly between users using web sockets. With this comes a platform-specific front end client which is what the users see and interact with. For Windows, which is the focus of this thesis, this is written in C#. To link the front end client with the library, XM-Reality have implemented a platform-specific bridge layer which handles all communication between the two.

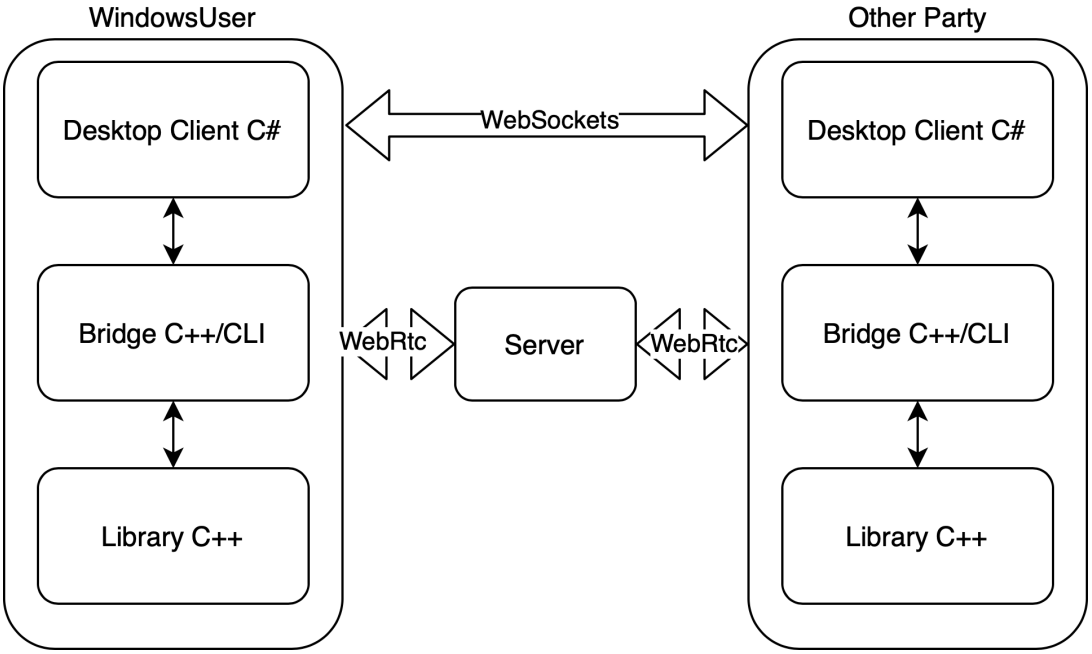


Figure 2.1: Flowchart of how the XM-Application works

[2]https://webrtc.org/

2.3 Hand Tracking

Hand tracking has been researched quite extensively. What technology used is often determined by cost and how flexible hardware setup one needs. Some technologies require multiple cameras and used to be really expensive. The method used in this thesis is model-based hand tracking and is described below.

Model-based Hand Tracking

Rehg and Kanade [7] describe a model-based hand tracking system. In their work, they track the position, rotation, and orientation of many points of interest or states, such as the palm, fingers, or finger joints. They have divided the palm into seven such points of interest, each finger into four and the thumb into five. The points are also connected. The tip of the index finger is connected to the joint closest to it, for example. So a hand pose is represented by 28 states. They then show how this can be used to build a model of the hand. These states are estimated by extracting line and point features from unmarked images of hands. They do this from one or two viewpoints. When trying to find the points of the hand, they use information about the current state generated by previous frames to get a good starting point. Their approach then searches the image along a line close to the starting point, where the previous angle of the finger determines the angle of the line. So to avoid pixel processing, they try to reduce the size of the image search. Searching a smaller part of the image allows one to use a higher sampling rate, which in turn gives more accurate starting points. When having multiple views, partial observations can be used. Points of interest found in one view but missing in another because of it being blocked can still be incorporated, giving a more accurate representation.

2.4 Leap Motion Controller

A leap motion controller is a commercial product developed by Ultraleap [3] and is used for hand tracking. It consists of two image sensors and three infrared LEDs.

Figure 2.2: A leap motion controller

Similar to Rehg and Kanade [7] they use a model-based tracking system. Each finger, including the thumb, has four tracking points. It tracks the base of the finger, the joints, and the tip. The palm has seven points and the wrist five. These points, excluding some for the wrist, can be seen in Figure 2.3.

The controller is limited to a reach of about 80 cm due to decreasing intensity from the IR LED when increasing distance. Data is streamed from the controller to a computer via USB. The data that the controller streams are two grayscale images of the near-infrared light

[3] https://www.ultraleap.com/
[4] http://blog.leapmotion.com/getting-started-leap-motion-sdk/hand-hierarchy/

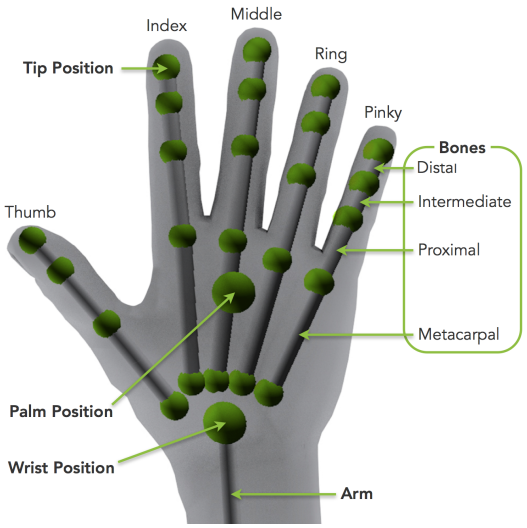


Figure 2.3: Points being tracked by leap motion controller[4]

spectrum (one for each camera). This is what is used by the Leap Motion software to get the tracking points and model the hand.

Weichert et al.[9] shows that the Leap Motion Controller has excellent accuracy with the difference between the desired 3D position and the measured position being below 0.2 mm with a static setup and under 2.5 mm when moving the reference point. As a baseline, the accuracy of a human hand is said to be around 0.4 mm.

2.5 Used technologies

This section explains the technologies used in the thesis.

C++

C++ is a high-level, general-purpose programming language. It was first created as an extension to C or C with classes. It has expanded steadily over time, and today it supports object-oriented, generic, and functional programming. It also allows for low-level memory manipulation. C++ is a compiled language and is known for being fast. C++ does not have any garbage collector, so one has to manage the memory used. The speed makes it suitable for heavy calculations in image processing.

C#

C# is a strongly typed object-oriented programming language. Microsoft developed it and released it in 2001. It has a garbage collector that takes care of memory clean up and also supports exception handling. Microsoft's strong support for this language and the control that comes with that often makes this the language of choice when doing front-end in a Windows environment.

C++/CLI

C++/CLI (Common Language Infrastructure) is a language specification created by Microsoft. It is used for achieving interoperability between C++ and C# applications. With it, one can have both unmanaged classes in C++ style and managed classes in C# style in the same program. It is essential if one wants to link a C++ library with a front-end built with C#.

TCP and UDP

Both TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are protocols used for sending data over an IP network.

A TCP-connection is made with a three-way handshake, first initiating and then acknowledging the connection. After a request has been made each packet is numbered to ensure them being in order and the receiving end must confirm that each packet has been delivered. If the confirmation is missing (if the packet was lost or an error occurred) the packet will be resent. A congestion control is in place to not overflow the receiver if it can handle the number of packets being sent. TCP is all about being reliable and because of that is a bit slower compared to UDP.

UDP does not have a connection. The packets are sent without any checks. They are unnumbered so they can get lost, be out of order, or sent multiple times. If an error occurs that packet is discarded. UDP is all about speed and is often used for streaming.

Unity

Unity [5] is a cross-platform game engine developed by Unity Technologies. It can be used to create 2D, 3D, Augmented reality, or Virtual reality games. It can also be used to create simulations, films/animations, or in engineering. The scripting API for Unity uses C#. Unity can ease the use of textures, rendering, and modeling a lot. One can also import and use many finished assets from various sources. One such example is the Software development kit that exists for leap motion in Unity [6]. With it, one can get finished models, shaders, and example scripts that can be used as a stepping stone.

Image Format BGRA32

Image formats describe how pixel data is stored in memory. BGRA stands for blue, green, red, and alpha, where alpha is the transparency. Each of those values is stored in a byte, which together takes 32 bits to represent a pixel. In Unity, the programmer can specify what format to be used by textures. In this project, BGRA32 is used.

[5] https://unity.com/
[6] https://developer.leapmotion.com/unity#5436356

3 Method

The following chapter includes details on how the prototype was developed. Firstly describing how the images of the hands are generated. Secondly, how they are transferred and integrated into the XM-application. Finally, the evaluation of the prototype is described.

3.1 Overview

The prototype used the XM-application, explained in Section 2.2, as a starting point. The structure of the prototype can be seen in Figure 3.1. The client takes all the input from the user and is what the user sees. The library does all the heavy calculations and transfer data between users, while the bridge handles communication between the library and the client. The bridge was modified and integrated into the existing application. The client and library was not modified. Hand Modeler is a Unity-application made from scratch. It takes input from the leap motion controller, converts it to a 3D-model, converts that model to a byte array, and then finally send that byte array to the XM-application using a TCP-connection. Different approaches were investigated to find out in which part of the XM-application the frames that are sent from Hand Modeler should be received. If it was put into the client layer, one would not be able to use functions from the library and if it instead was put directly into the library it would be hard to know when hands overlay should be activated or deactivated. In the bridge layer, one can communicate both with the client layer and the library layer, which was essential when merging frames from Hand Modeler and the streamed video from user cameras. It was therefore put in the bridge.

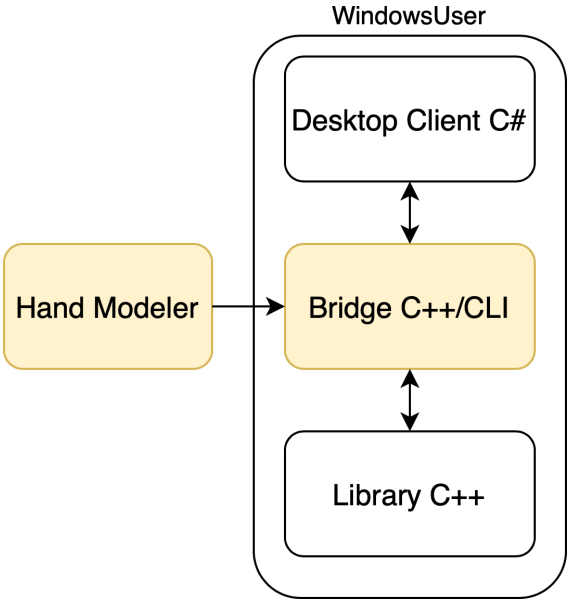


Figure 3.1: The three layers of the XM-application and where frames from Hand Modeler is sent and received.

3.2 Building the Hand Model

When creating a 3D model a leap motion controller and its software was used to do the hand tracking. The output was then used in Hand Modeler. Figure 3.2 describes the information flow. The leap motion controller feeds the leap motion software with data that is converted to points of interest representing a model of a hand, as described in Section 2.4. The leap motion SDK[1] (Software Development Kit) comes with an API (Application Programming Interface) which can be used when making applications in Unity. It was used to convert the models made by the leap motion software to actual 3D models. The solution used the example project included in the Leap Motion SDK as a starting point and used the two scripts that came with it. The first script is the leap service provider, which handles the communication between the leap motion software and Hand Modeler. It checks if a controller is connected and extracts the data, if any is sent. The second script is for handling the hand models.

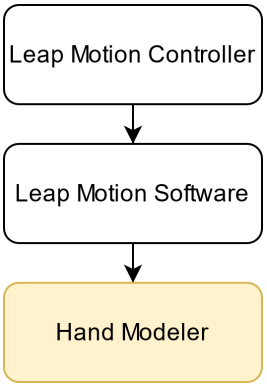


Figure 3.2: Information flow from the Leap Motion Controller into Hand Modeler

[1]https://developer.leapmotion.com/unity

3.3 Convert Model to Image Data

Hand Modeler renders the 3D representation of the hands in a scene. However, the models are not meant to be seen in the Hand Modeler application but instead meant to be transmitted to the XM-application and displayed there. To do this, we create a Texture2D[2] which represents an image. The width, breadth, and image format was set of this texture to 640, 480, and BGRA32, to match what is used in the XM-application. Pixels are then read from the scene and saved into that texture. The raw pixel data can then be extracted and stored in a byte array. This byte array has a size of 640 * 480 * 4, and are now ready to be transferred to the XM-application.

3.4 Transfer Image Data to XM-Client

The image data is sent using a TCP-connection. When working with an image, it is essential to know the format. Although somewhat slower than a UDP-connection, TCP enables one not to worry about packet loss and the programmer can assume that each frame has the same size. It is a trade-off between speed and reliability. In the worst case, package losses on the client side could cause segmentation errors that would crash the application.

The application is, on a separate thread, listening for new connections. Once a connection is established, the frames are streamed as long as the connection is active. The frame rate that these are streamed at can easily be modified and is limited by the available computing power. A problem that occurred was that the receiving side got the contents in the byte array representing the image in reversed order. This resulted in the image being mirrored. To maintain the pixel format a workaround for this was to first reverse the image vertically and then horizontally.

3.5 Integrating Hand Frames with XM-Application

In this project, the goal was to get a proof of concept and a demonstration. One could display the hand models in the client layer but would then not fully integrate into the XM-application. It would allow one to see how good the leap motion controller works but not compare it to the old solution. The old solution was used as a starting point to integrate into the XM-application as smoothly as possible. The way it worked is that it took frames from a camera, and applied an image segmentation algorithm. The output of this algorithm was a frame with the hand, and the rest of the image was masked with a solid background (black or white depending on which settings are used). This new frame is then sent to the library, where it is merged with the current video. The XM-library also sends it to the other user if in a call. The approach used here was to replace the image that previously came from a camera and fed to the image segmentation algorithm with the frames coming from Hand Modeler. The frames from Hand Modeler have a solid background and do not need to be manipulated before being sent to the library. If this is done correctly, the old pipeline can still work the way it did and continue to work on all platforms.

[2]https://docs.unity3d.com/ScriptReference/Texture2D.html

3.6 Evaluating the Prototype Based on FPS, CPU, and Memory Usage

The XM-application can run on both mobile devices and on desktop. Since it runs on mobile devices, power usage is of interest. It was therefore decided that the visual gains from having a frame rate of more than 30 was not worth the cost in power usage. As a baseline, the old solution has a frame rate of 25. This project tried to get the highest possible frame rate up to a maximum of 30. The frame rate is dependant on how fast images can be sent from Hand Modeler and then received in the XM-application. If the XM-application is unable to keep up, the frames will start stacking up, and the delay starts increasing. This makes it easy to notice and test. It was achieved, so a frame rate of 30 was used for testing.

Performance-wise, two things are interesting, delay and computational needs. Since the goal frame rate was achieved, delay was not a problem. Because of this only the computational needs, CPU and memory usage, was recorded and evaluated. The use cases this thesis is affecting is in-call with hands overlay and is what was evaluated. Hands overlay was tested with or without hands present since this affects the leap motion controller's power usage. This was compared to the old version with hands overlay. For the purpose of having a baseline, both the new prototype and the old solution was recorded while being idle. The measurements were done on a system with an Intel Core i7-6700HQ processor, 16 GB ram, a Geforce GTX 960m graphics card and running 64-bit Windows 10. The measurements were done using Windows Performance Monitor. Each experiment ran for at least 10 minutes with measurements taken every five seconds.

4 Results

This chapter presents the results relative to the research questions being investigated. Firstly the prototype is described, and then the evaluation of it is presented.

4.1 Hand Modeler

In the final version of the product, Hand Modeler just runs in the background and does not display anything. However, for testing purposes and if one wants to adjust anything, it can still be useful to know how it looks. In Figure 4.1 an example image of the hand models is shown. The models are put onto a solid black background. The models can be with or without the wrist and have a light or dark skin tone. If one adds more 3D models or textures, choices could be further extended. The perspective chosen is first-person. Therefore, the hands added to the video have roughly the same position as one's own hands. The perspective is easily modified by changing the position and angle of the camera in Hand Modeler. The frame rate is set to 30 frames per second in order to limit CPU-usage.

Figure 4.1: Virtual hands seen from Hand Modeler.

4.2 XM-Application

After the hand image is transferred from Hand Modeler to the XM-application, the background is masked away. Examples of resulting images can be seen in 4.2. Small parts of the background at the edges of the hand can still be seen in screenshots. However, this is not as visible when the video is streamed and with the hands possibly moving. The refresh rate is limited by how many frames are sent per second by Hand Modeler.

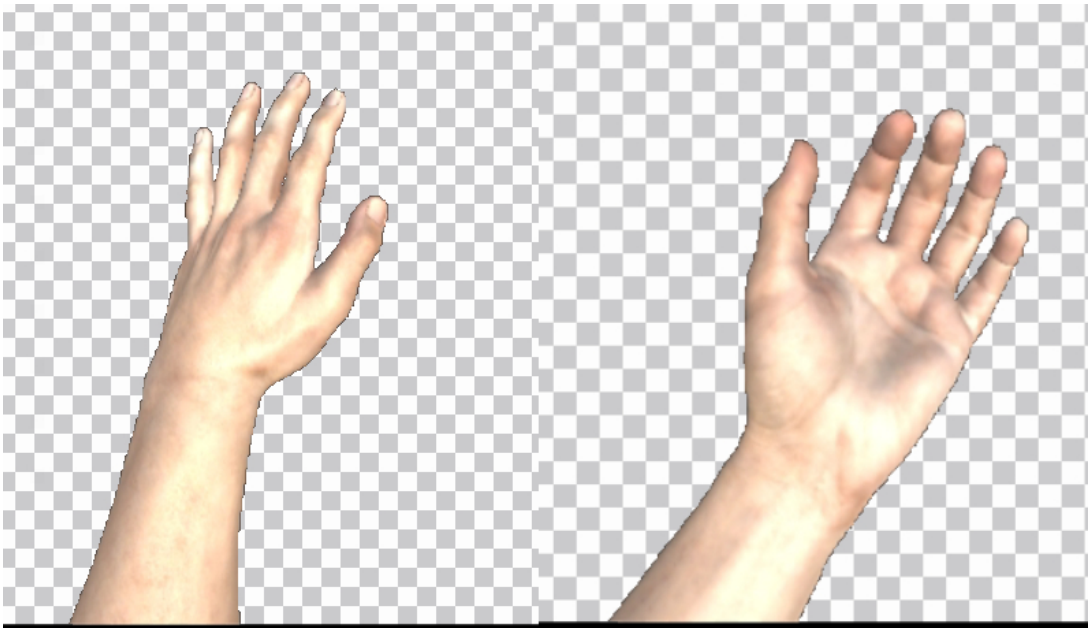


Figure 4.2: The virtual hands after being received in the XM-application

4.3 XM-Call

The prototype presented in this thesis is only available for windows. However, it can still communicate with all types of devices supported by the original solution. One can still make calls to mobile devices, web, or desktops. Compared to the current solution, nothing is changed for the receiving end except what the image coming in looks like and has the same memory and CPU usage as before. Figure 4.3 shows how it can look during a video session. The screenshots are from a call to an IOS device.

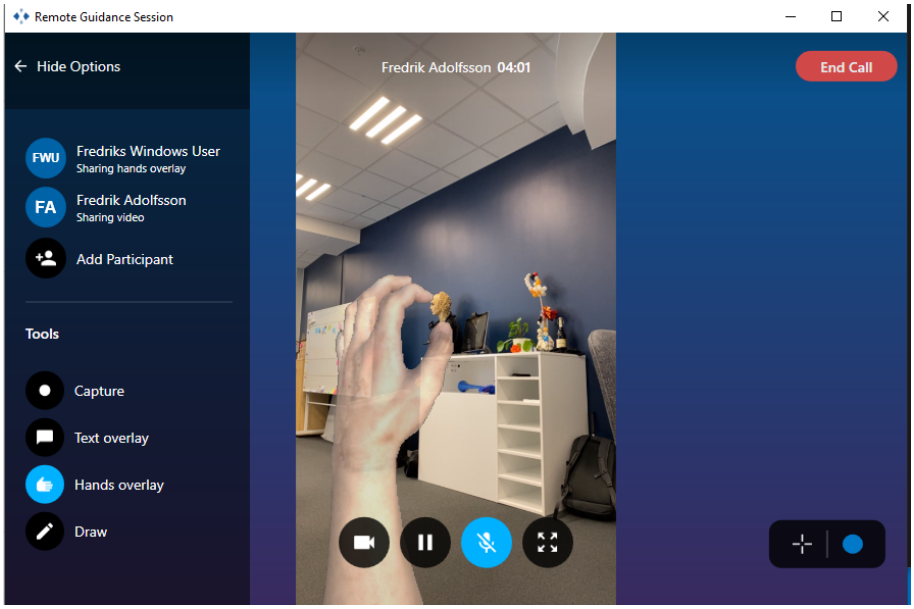


Figure 4.3: Hands overlay in a video session using the prototype

4.4 Evaluation

The prototype was evaluated based on CPU and memory usage. The results for CPU-usage can be seen in Figure 4.4. The use-cases measured were the app being idle (blue line), being in a call but not having any hands present (yellow line), and being in call with the hands (green line). The old solution was used as a benchmark for comparison (red line). Both versions draw about the same when idle and use less than ten percent of the processor power. In call, there is about a ten percentage points increase without the hands and another ten percentage points with hands. The solution can be divided into three parts: XM-application, Hand Modeler, and Leap motion software. Hand Modeler is not affected if one has hands present or not and explains the difference between the old solution and no hands. The leap motion software used to track the hands accounts for the remaining increase.

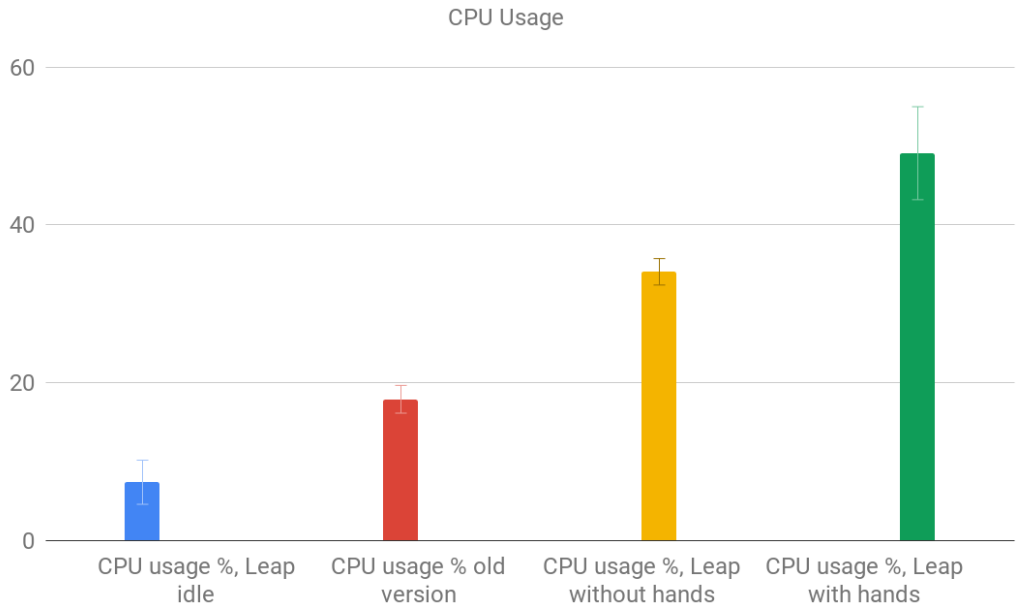


Figure 4.4: Diagram showing CPU usage for different use cases

Memory usage was evaluated when in call, with or without hands present, and compared to the old solution. An increase of about ten percent can be seen when using the new solution and having hands present. There is also a small increase even without the hands. The memory usage does not change over time.

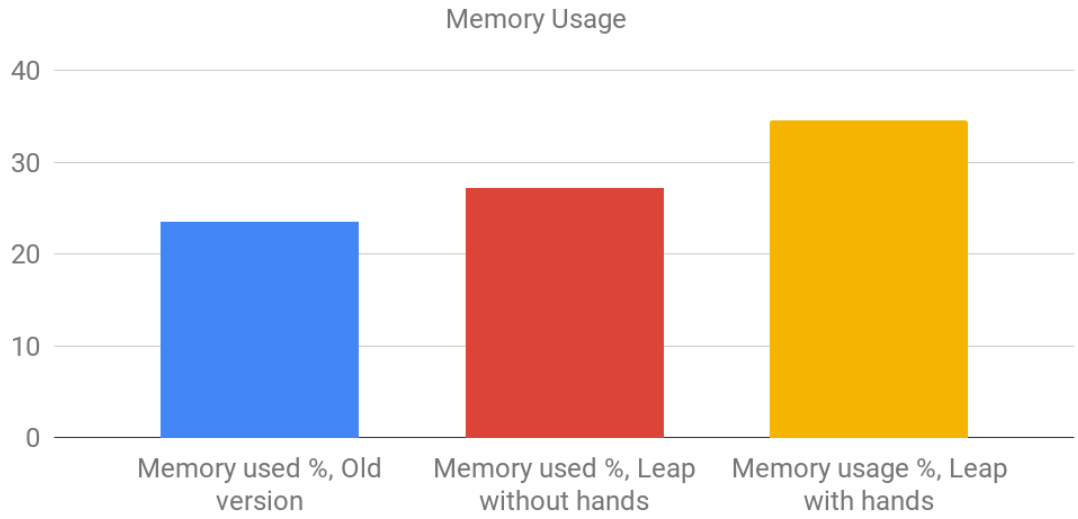


Figure 4.5: Diagram showing Memory usage for different use cases

5 Discussion

This chapter discusses, comments on, and criticizes the result and method. It also discusses the thesis in a larger context.

5.1 Result

The prototype allows one to use hands overlay with the leap motion controller. There is a clear overhead compared to the old solution. The memory overhead is quite small, about 10 percent. This should not be an issue for users. However, there is a significant increase in CPU usage. Creating a 3D model requires more computation than just using a 2D image. There is also an overhead when converting the model to an image and then sending that from Unity into the application. The overhead only affects Windows users though. When calling to another device, the other side has no decreased performance but can still see the new hands overlay. Power drain is mainly a factor on mobile devices, and as long as the application can run on a regular computer, the overhead should not be crucial for windows users. This prototype allows one to choose perspective, choose what model to use, and all one need is a small device that can easily be transported. It works with any background as opposed to the old solution working only with uniform ones. A drawback is that only the hands can be tracked. Tools, for example, can not be seen.

The flexibility that the new solution offers is most likely hard to increase, even if a different approach was used. However, when implementing software many choices are made that can affect performance. If one could do the modeling directly in the XM-app instead of in Hand Modeler, it would probably have lower CPU usage. Nevertheless, it is Hand Modeler that lets one change models or perspectives. It is a trade-off between utility versus performance. When integrating the prototype with the XM-app as much as possible of the old solution was left untouched, modifications could probably yield performance gains.

5.2 Method

The method worked and led to good results. The prototype works as a proof of concept and shows the potential of the leap motion controller. However, the method could be improved in some aspects.

This thesis shows one way that hands overlay can be done. The prototype reuses a lot of the old solution, and it would be interesting to see what the performance would be if one did everything from scratch. Since images transferred from Hand Modeler have a perfect solid background, one could use a more simple way of image segmentation. This would probably reduce the overhead. Unity was chosen for doing the hand modeling, mainly because of ease of development. It is neat and flexible. However, it would most likely require less CPU if it was done directly in the XM-application. There is an overhead doing the modeling in Hand Modeler and then transferring it into the app.

The main purpose of the evaluation is to see if a normal computer can run the program. Nevertheless, the idea of what a normal computer is can differ from person to person. It could be interesting to test and have lower fps and see the difference in computational needs. We argue that the main point is to see the difference between the new and the old solution. Furthermore, one could reduce the fps on the old solution, which should make the relative overhead about the same.

The sources in this thesis are mostly from either published papers or the creator of the tool/product. For example, with sources regarding the Leap motion controller they are from the company itself, what they say should be valid.

5.3 Thesis in a Larger Context

Hands overlay is an important part of remote guidance which allows people to avoid traveling on site. This allows expertise to instead be shared over distance, making it more accessible. As the part of expertise delivered over distance increases the need for traveling decreases, which should have a positive environmental impact, as well as economic, for everyone involved. If this thesis helps improve the experience and popularity of remote guidance, it will increase that effect.

6 Conclusion

This chapter concludes the thesis relative to its research questions.

6.1 Can hands overlay be done using a leap motion camera and software?

A prototype and a proof of concept are introduced in this thesis. It shows that hands overlay can indeed be done using a leap motion controller and the software that comes with it. With the help of the prototype developed, the potential of a model-based approach can be seen. This is helpful when deciding if this is something the field of Remote guidance should continue researching.

6.2 How are the frame rate and computational needs compared to the old solution?

There is a clear increase in CPU-usage compared to the old solution. This is partly because images are transferred from Hand Modeler and then into the XM-application. Interprocess communication comes with some overhead. The rest of the increase comes from the leap motion controller, building a model is more draining than taking a picture. There is a small increase in memory usage. People should still be able to run this on their normal computer. The overhead should, therefore, not be too important. The frame rate is the same compared to the current solution.

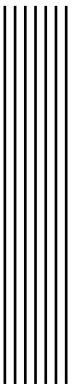
6.3 How can it be integrated into the XM-client?

This thesis proposes an approach where one only changes how the images of hands overlay are generated. That allows one to reuse the old solution as much as possible. The Hand Modeler extension is integrated into the bridge-layer which had to be modified. The lib and front-end is unchanged but still used. The images are generated in Hand Modeler and then transferred into the application via a TCP-connection. This allows the prototype to keep all the functionality from the old solution.

6.4 Future Work

Future work for this thesis could be as follows:

1. Implement hand modeling directly in the XM-application instead of in Hand Modeler.
2. Test and modify the prototype based on user-experience. What hand model and perspective should be used? Maybe allow the user to modify it themselves.
3. Investigate a head-mounted setup instead, how would the performance change?



Bibliography

[1] Poole Alexander. "Real-Time Image Segmentation for Augmented Reality by Combining multi-Channel Thresholds". MA thesis. 2017.

[2] D Bassily, C Georgoulas, J Guettler, Thomas Linner, and T Bock. "Intuitive and adaptive robotic arm manipulation using the leap motion controller". In: *ISR/Robotik 2014; 41st International Symposium on Robotics*. VDE. 2014, pp. 1–7.

[3] Maryam Khademi, Hossein Mousavi Hondori, Alison McKenzie, Lucy Dodakian, Cristina Videira Lopes, and Steven C. Cramer. "Free-Hand Interaction with Leap Motion Controller for Stroke Rehabilitation". In: *CHI '14 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '14. Toronto, Ontario, Canada: Association for Computing Machinery, 2014, pp. 1663–1668. ISBN: 9781450324748. DOI: 10.1145/2559206.2581203.

[4] CA Mather, Tony Barnett, Vlasti Broucek, Annette Saunders, Darren Grattidge, and Weidong Huang. "Helping hands: using augmented reality to provide remote guidance to health professionals". In: *Studies in health technology and informatics* 241 (2017), pp. 57–62.

[5] M. Mohandes, S. Aliyu, and M. Deriche. "Arabic sign language recognition using the leap motion controller". In: *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*. 2014, pp. 960–965. DOI: 10.1109/ISIE.2014.6864742.

[6] Ohan Oda, Carmine Elvezio, Mengu Sukan, Steven Feiner, and Barbara Tversky. "Virtual Replicas for Remote Assistance in Virtual and Augmented Reality". In: (2015). DOI: 10.1145/2807442.2807497.

[7] James M Rehg and Takeo Kanade. *Digit-Eyes: Vision-based human hand tracking*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1993.

[8] Anshul Sharma, Aditya Yadav, Saksham Srivastava, and Ritu Gupta. "Analysis of movement and gesture recognition using Leap Motion Controller". In: *Procedia Computer Science* 132 (2018). International Conference on Computational Intelligence and Data Science, pp. 551–556. ISSN: 1877-0509. DOI: 10.1016/j.procs.2018.05.008.

[9] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. "Analysis of the Accuracy and Robustness of the Leap Motion Controller". In: *Sensors* 13.5 (2013), pp. 6380–6393. ISSN: 1424-8220. DOI: 10.3390/s130506380.