RWTH AACHEN UNIVERSITY

The present work was submitted to Visual Computing Institute

Bachelor Thesis
Wintersemester 2016 - 2017

HoloConnector: A framework enabling remote 3D projection definitions in the context of CNC Machine Simulation with Microsoft Hololens

Bachelor Thesis in Computer Science by

Pascal Zenker
Student ID Number: 333025

Prof. Dr. Torsten W. Kuhlen
Virtual Reality & Immersive Visualization

Supervisor: Dr. Meysam Minoufekr

Aachen, February 15, 2017

Contents

Acknowledgements

At this particular point I want to thank everyone that supported and motivated me during the writing of this thesis. It has benefited greatly from their support.

Some of them I want to name explicitly. To begin with, I want to thank Prof. Torsten Kuhlen and Prof. Jan Borchers for taking the time to examine this work as first and second reviewer. I want to thank my supervisor Dr. Meysam Minoufekr for his guidance and his shared expertise. He also encouraged me to write about such an interesting and forward-looking topic. Furthermore I owe thanks to my second supervisor Philipp Wacker for both his valuable input and for providing me with all the necessary facilities for development. Also I am thanking all my dear fellow students, who helped and supported me through my whole bachelor studies. Especially Tobias for proofreading this thesis.

In addition, I want to thank my girlfriend for her patience and encouragement during the phases, in which I hang my head.
Last but not least I want to express my deepest gratitude towards my parents who always had my back and still give me the opportunity to live my live as I have always wanted to. Thank you very much.

Abstract

With the ongoing development of both augmented and virtual reality, new important paths open up for the use of computer aided manufacturing. Microsoft's new mixed reality device, the HoloLens, bridges the gap between reality and media by injecting holograms into the user's field of view. This new way of showcasing digital data allows developers to think whole new fields to develop for. In this thesis the use case of working with a Hololens in productional engineering will be examined and the controlling of the HoloLens improved. This work will describe and introduce a framework which enables users to transfer data or 3D models via an interface from any computer architecture to the HoloLens in context of a CNC machine simulation. Machine models can be picked on a remote computer and be loaded into the Hololens as hologram. Through the framework they can be simulated and the machine work observed before the actual process starts. This paper especially concentrates on the requirements for hard and software for a project of this kind. Furthermore, the connectivity from a remote computer to the HoloLens is discussed. Lastly, the visualization system for CNC machine simulations, explicitly the kinematic chain inside a holographic environment, is discussed. The goal is to develop a framework, which can be extended and transferred easily to other similar areas.

Überblick

Mit der immer fortwährenden Entwicklung von Augmented und Virtual Reality öffnen sich neue wichtige Wege für die Nutzung des Computers als Hilsfmittel zur machinellen Produktion. Microsofts neues "mixed reality" Gerät, die Hololens, überbrückt die Schlucht zwischen Realität und Medien, indem sie Hologramme in das reale Blickfeld des Nutzers einbindet. Diese neue Art der Einbindung von Medien wird es Entwicklern erlauben ganz neue Bereiche zu erkunden. In dieser Arbeit wird untersucht, wie ein Use Case für die Nutzung der Hololens in einer industriellen Fertigung aussehen könnte und gesteuert werden würde. Diese Arbeit stellt ein Framework vor, welches dem Nutzer erlaubt Daten oder 3D Modelle mit Hilfe eines entfernt gelegenen Interfaces auf die Hololens zu übertragen. Dabei sollen die Computer Architektur oder das Betriebssystem keine Rolle spielen. Als Andwendungsbeispiel wird die CNC Produktion genutzt. CNC Machinen Modelle sollen auf einem entfernt gelegenen Rechner ausgewählt werden können, auf die Hololens übertragen und als Hologramm geladen werden. Mit Hilfe des Frameworks soll dann ein CNC Fertigungsprozess simuliert und als Hologramm begutachtet werden können, bevor der eigentliche Prozess startet. Diese Arbeit wird sich dabei insbesondere auf die Anforderungen an solch ein System in Hinblick auf Hard- und Software konzentrieren. Des Weiteren wird die Verbindung zwischen entferntem System und der Hololens begutachtet und entwickelt. Außerdem wird ein System zur kinematischen Darstellung von CNC Machinen, insbesondere der kinematischen in einer Holografischen Umgebung ausgearbeitet. Ziel ist es ein Framework zu entwickeln, das einfach erweiterbar ist und später auch in ähnlichen Anwendungsgebieten genutzt werden könnte.

1 Introduction

Augmented and virtual reality are on their way to change human's interaction with computers as drastically as the step from command line interfaces to graphical user interfaces (GUI) with the announcement of the Apple Lisa in 1983 [Perkins et al., 1997]. They offer new ways in which humans can interact with a computer intuitively and how computers can be integrated in one's everyday and working life. Especially augmented reality (AR) enables the user to visualize computer controlled information while being fully aware of her surroundings. The recent success of the mobile application "Pokemon GO" shows that mainstream is ready for these new technologies. Now it is time to transfer the influences of this new technology and search for fields where AR enhances current work-flows.

In the private sector, wearable AR devices could possibly replace any kind of display. TVs or computer displays, might not be necessary anymore at a certain point in the future. This could create whole new kinds of games or multimedia applications. E-Commerce could offer virtual fitting rooms, interior design could be tested right in the room [Azuma, 1997].

Arguably, AR also has great potential for the industrial sector. Production workers could gain more process relevant content without the need to access a computer. Shared working processes between robots and humans can be coordinated more intuitively. Design processes could be loosened from a desktop experience to a fully immerse AR representation. This does not only allow the creator to have a better understanding of his working piece but also make rapid prototyping for visual feedback unnecessary. These are examples of how this new technologies could impact into the modern computer work.

This thesis showcases a framework that can be used to simulate CNC machine productions through augmented reality features using the Microsoft Hololens.
In chapter 2 the state of the art of AR, VR and MR is examined. This chapter will explain modern "Game Engines" and clarify some of the terms which are used during this thesis. Furthermore it will present some of the work that has already been combining CNC machines and AR.
The next chapter outlines the motivation and objective of the thesis. It frames the main hypothesis on which this thesis is based: A software and hardware solution could be built that enables a production worker to better work and visualize digital data. Later on, it will introduce the three main research issues that are based on this hypothesis.
For a better in-depth understanding of CNC machines, chapter 4 is explaining CNC machines and their use in the industry. It presents the work-flow while using a CNC

machine and showcases how CNC machine simulation is done. In the following chapter, the requirements for the HoloConnector framework are listed.
The thesis continues by discussing different wearable devices that could be used to visualize digital data. After a fitting device is found, the software systems which can run on the device are compared and evaluated. Chapter 6 describes different approaches to build a remote connection to the framework and presents the chosen solution. Afterwards, in the next chapter, the visualization is explained. It is shown how complex mechanisms of the CNC machine can be represented with a computer. Then, the implementation of these mechanisms is explained in depth.
The framework is evaluated in the next chapter. First off, the working system is presented. Furthermore, it is examined if the research issues are properly answered. In the end, the overall speed of the framework is evaluated.
The last chapter is a conclusion and summarizes the thesis. Additionally, it will give a small outlook onto the future.

2 State of the Art

This chapter gives an overview on the differences between the existing techniques that allow users to discover virtual content in a real life experience, namely virtual reality (VR), augmented reality (AR) and the new term mixed reality (MR), which was created by Microsoft and defines a special case of augmented reality. Later on, different devices that can be used for this kind of visualization are presented shortly and compared to each other. Last but not least, game engines, which can be used to visualize data and objects on these devices, are evaluated.

2.1 AR, VR, and MR

To the modern day there are two major extensions for the human perception of reality, that have the ability to change our way how we interact with computers.

In the 19th century, first, the term Virtual Reality was used. Here it described the fact that theater can create an illusion which gives the feeling to be in a virtual or artificial reality. Since then, the term Virtual Reality or VR has changed a little and now describes all methods that change the user's field of view from the real life to a digital representation. In this alternate environments, he often can move and interact with digital content. It should arouse the feeling of being physically present in a virtual world. This is often supported through sensors, that allow users to look around and hear or move in these worlds [Rheingold, 1991].
As early as 1901, the author L. Frank Baum imagined the first kind of Augmented Reality. He described a device that overlays data onto the real life. Our current understanding of augmented reality does not differ that much. Azuma describes AR as the integration of 3D virtual objects into our real 3D environment. Virtual data is processed and can be used or interacted with in real time [Azuma, 1997]. With this definition in mind you can compare VR and AR easily. While VR completely relies on a virtual representation of the world, AR is the step between VR and real life, where digital and real objects are mixed together in real time.

On 30th of March 2016 Microsoft started shipping their own AR device: The HoloLens. Microsoft describes their technique not as AR but as an expansion the Mixed Reality (MR). MR means that there exists not only the real life world, but also a virtual pendant of it. Mixed Reality has the advantages of virtual content to interact with real life content and therefore allows new technologies, which inject digital data in a very natural feeling way into our everyday life [Ohta and Tamura, 2014]. In the Hololens this digital copy of the real world is created by using sensory tech-

niques, which can scan the real world and transfer it into a mesh representation. Holograms that are projected into the real world can use the data of the virtual pendant. This allows the device to mimic interaction from holograms with real world objects [Microsoft, 2016b]. This work will concentrate on the Hololens and therefore act in the range of this specific MR or generally AR.
In contrast to Microsofts definition of MR, which is basically a special kind of AR that can interact with the real environment, the term Mixed Reality was also used by [Milgram et al., 1995]. For them, MR describes the whole continuum that exists between the real environment and a virtual world, as seen in 2.1. They include the term Augmented Virtuality (AV), which basically is the opposite to AR. It describes to include real world things, for example a hand or an arm, into a virtual reality. Through this immerse object you can manipulate the virtual world. Today, this kind of technique is mostly set under the term Virtual Reality and not so much a category of its own.

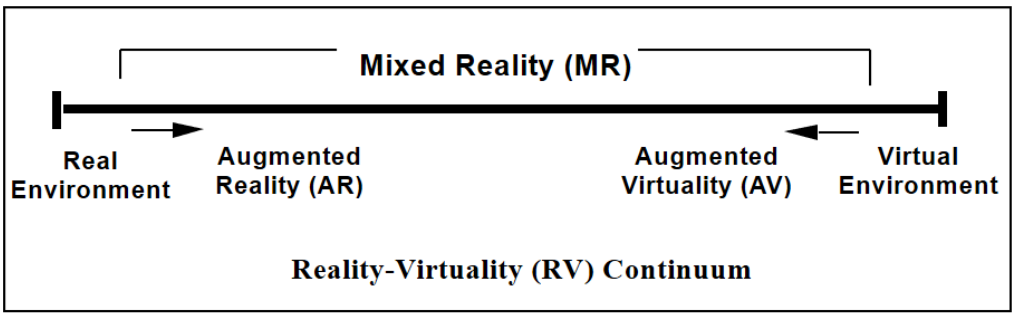

Figure 2.1: Mixed Reality continuum taken from [Milgram et al., 1995]

2.2 Holographic representations

Due to the usage of the word *hologram* in this thesis, there need to be some clarifications. All devices and technologies that will be presented in this thesis are not producing actual holograms. A hologram is "a three-dimensional image reproduced from a pattern of interference produced by a split coherent beam of radiation" [Merriam-Webster, 2017]. This is seen without the aid of any device. Due to this technical specification, none of the presented devices are actually producing holograms. They rather perform perspective projections with the help of lenses or other techniques that produce a holographic experience. Through transforming objects in a way, that these seem to be at a certain position.

In order to ease the understanding of this thesis in later passages, the term *hologram* will be used, even if the described experiences are perspective projections.

2.3 Game Engines

To the current date, there are multiple different game engines on the market. Game engines are used to simplify graphical representation in terms of development time and cost. Modern engines include several important mechanics for 3D development. For example transformation of objects or texturing of 3D meshes. They usually are targeted to run on multiple devices, for instance mobile phones as well as computers. By this, the developers should be able to target a great audience without the need to port a game from one platform to another.
They also allow fast development cycles because they release developers from the need to have deeper computer graphics (CG) knowledge. The engines wrap complex software solutions as OpenGL or DirectX and therefore enable small developers to start programming content without the need to program a graphics engine. Another major point that makes game engines so valuable for the modern computer graphics market are the included physics and collision systems. These make complex physical development obsolete. Most engines include a lot of presets and are easily extendable with plugins as well as own script code. In the modern area of AR and VR development, the engine developers try to implement easy solutions for both.

2.4 TCP

The *Transmission Control Protocol* (TCP) is a network protocol. Mostly every modern computer architecture can use TCP. It defines in which way data is transferred between two network components. Its major competitor is the "User Datagram Protocol" (UDP). But TCP has some advantages. Its reliable and therefore guarantees to deliver a message or send an error if it failed. Furthermore, it settles its connection between two endpoints, the sockets, and allows data transfer in both directions through this. A connection which is working in both directions is called full-duplex in networking. Another benefit is,that TCP guarantees ordered delivery. All of these are major points which are important for correct and realiable data transfer [Tanenbaum et al., 2003].

2.5 CNC Machines and AR

Computer Numeric Control or CNC is the electronic procedure of controlling machine tools. Machine tools are machines that work with metal or other rigid materials. They can for example cut complex forms out of metal plates or transform these by bending or shearing. CNC processes are normally planned and simulated before the machines are started. The work pieces can be created inside CAD tools and afterwards this model can be transformed with a CAM software to G-code which can be read by the machine tools.

The question is how AR and CNC machines can be combined. An intuitive approach is taken in [Olwal et al., 2008], which tries to emerge digital data into the field of vision of a production worker. They describe their tool as a possibility to give a production worker a better understanding of what is currently happening in terms of data. For example they inject the state of the machine like temperature or workload into the users sight. This should enable the worker to get a better understanding of complex mechanisms inside of the machine, while being able to spare any additional displays or tools around him, which he would have to consult.

In [Kiswanto and Ariansyah, 2013] the basic idea behind combining AR and CNC machining is to simplify the simulation process. They claim that VR has sit-backs in terms of measure units and correct machine behavior. Therefore, they want to combine the real machine with AR input to get a realistic as possible simulation, while not having to concern with the problems VR brings with it. Which are claimed to be motion sickness or disorientation.

A similar work is [Zhang et al., 2010]. Their intention is to allow trainees to work with a real machine and adjust parameters on it, while not having to worry about breaking the machine or work tools. This is done in the way, that you control a real machine, but the actual working pieces are just holographic projections. So while the controlling is real, the actual work process is done as a simulation in the AR environment. This would allow very specific and complex training, that would be very cost efficient as well.

2.6 Machine Kinematics in Computer Graphics

In general, machine kinematics or a kinematic chain is a series of rigid bodies that are connected with joints. Each machine part's movement is restricted by the parent part's movements. The position of the different parts can be changed, which is called transformation. Transformation in machine kinematics are either translations, a movement, or rotations. When a parent part is transformed, its children will undertake an according transformation, so that the relative position to its parent part is still the same.

All transformations are mathematically described with matrices. In the next sections, this thesis showcases these transformations. Nearly all of the machines that are used in the industry, use the *Cartesian Coordinate System*. This coordinate system has three dimensions, x, y, and z. In computer graphics, for the same transformation it is easier to use extended coordinates. Extended coordinates have 4 entries and the resulting matrices are 4x4 matrices. This allows to define rotation together with translation inside one matrix [Bohez, 2002].

2.6.1 Translation

If you want to translate and therefore move a point *pos* about the amount of a vector *tra*,

$$pos = \begin{pmatrix} a \\ b \\ c \end{pmatrix} tra = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

this is done by multiplying the vector with the following matrix:

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} a \\ b \\ c \\ 1 \end{pmatrix} = \begin{pmatrix} a+x \\ b+y \\ c+z \\ 1 \end{pmatrix}$$

2.6.2 Rotation

The following matrix does a rotation for an object around the z-axis. The angle α determines the degree by which the object is turned.

$$\begin{pmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

To rotate an object not around the origin but around a normalized vector,

$$rot = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

it is possible to calculate a matrix that transforms the coordinate system so that r is the new x-axis. This can be done by calculating the orthonormal base and write its vectors into a new matrix T. After applying this transformation the former vector *rot* now is the new x-axis. Then the object can be rotated around the x-axis as mentioned above. Afterwards the transformation can be undone by multiplying with the inverse of T: $T^{-}1$. For rotation matrices it is set that $T^{-}1 = T^{T}$. This leads to the following matrix for a rotation around the r-axis:

$$R_{rot} = R * R_{\alpha}(x) * R^{T}$$

2.6.3 Object Transformation

Now, after introducing these two basic mechanics, the advantages of extended coordinates can be seen more easily. The inner 3x3 matrix of a homogenous matrix

allows to rotate the object while the fourth row allows to apply a simultanoues translation. For example this matrix rotates an object around the above mentioned axis R_z, translates it about tra and additionally scales it with factor s [Paul, 1981].

$$\begin{pmatrix} s*cos\alpha & -sin\alpha & 0 & x \\ sin\alpha & s*cos\alpha & 0 & y \\ 0 & 0 & s & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Because it is possible to multiply matrices of the same dimensions, we can now stack different transformations, which results in a complete transformation M as this: $[M] = [X] * [Y] * [Z] * ....$

Using this technique, in one calculation step, multiple complex transformations can be done. Nevertheless, it is a little bit more complicated than that when working with robots or machine kinematics. The condition for the correctness of the matrix multiplication is that every object uses the same coordinate system. While this is a minor problem in computer graphics, it is a very important topic for machine kinematics. Usually every part of the machine has its own coordinate system, which is based on its origin. This is because the movements of the single machine parts are always relatively to their parent. For example, the rotation table B in figure 4.5, always rotates around its own y-axis, while its position and orientation can change when B is rotating. The problem is explained in figure 2.2.

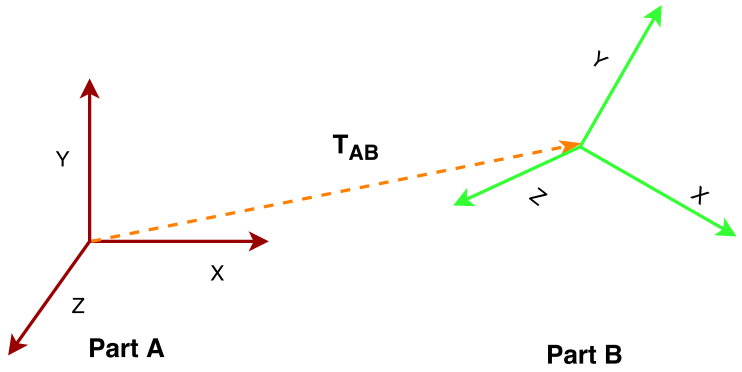

Figure 2.2: Two parts and their individual coordinate systems

The next step is to find the homogeneous matrix that describes these different coordinate systems and can move the individual parts together. Basically, its sufficient to translate every coordinate system, so that it is based inside the global machine coordinate system [Bohez, 2002]. Integrated into the matrix above, the result is the following:

$$\begin{pmatrix} s*cos\alpha & -sin\alpha & 0 & x+T_{AB}.x \\ sin\alpha & s*cos\alpha & 0 & y+T_{AB}.y \\ 0 & 0 & s & z+T_{AB}.z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

With this action, all parts of the kinematic chain can be transformed each on its own as different transformations of the rigid bodies, which are later multiplied to one complete transformation of the whole machine. The end matrix M describe the complete transformation of one step of the robot for its individual part, be it transformation or rotation or both [Uicker et al., 2011].

To calculate the kinematics itself is another difficult topic, which can be reduced to two big problems. The "Forward Kinematic", which can calculate the position of each point of the robot or machine with the given length of each link and the angle of each joint.
The other topic is the "Inverse Kinematic", which wants to find the angles of each joint, while having the length of each link and the position of some point. Basically it is the conversion from angle to position and backwards [Paul, 1981].
But because this thesis is not about the kinematics of robot itself but only their simulation, it is not necessary to examine this topic further. For additional information refer to [Uicker et al., 2011].

3 Objective of the Thesis

The first section of this chapter is going to specify the motivation which leads to this thesis as well as the objective of the framework which should be developed. Afterwards, a course of action will be designed, which fits to the motivation and objective.

3.1 Motivation & Objective

The modern production heavily relies on CNC machines. These are controlled by different data sets that are created in a postprocess. In this process often simulations are run beforehand to ensure working processes will lead to the result that is wanted. While CNC machines are a huge assistance for modern production chains, for the production worker themselves these complex technical constructions are mostly black boxes. It is very hard to understand what is happening inside the machine, as well as it is very difficult to combine simulation data with real machine working steps inside. Due to this lack of knowledge, it is hard to recognize errors or malfunctions of a CNC machine.

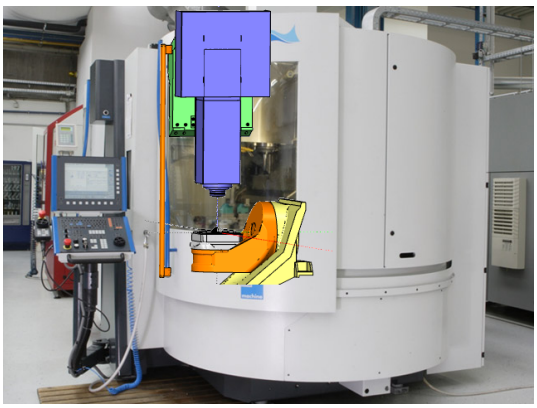

Figure 3.1: Expanding workers vision through AR

To compensate this missing information, production workers must often decide between observation at the machine or looking at computer optimized data that tells the worker what is happening. There is no way to decide which information is more important because both information, the real machine as well as the computer optimized status information of the machine are equally important. While this were two sets of information which had to be strictly separated in the past, modern technologies arise that can help to bridge this gap. Mixed Reality devices and applications are getting cheaper and more reliable. Wearable devices like the Microsoft Hololens were a huge surprise for the industry. Now, it is time to take the potential these technologies offer and insert data provided by computer into an actual work flow. With this, production workers' perception, and interaction possibilities can be en-

hanced, while still giving them full control an awareness of their environment.

This kind of problem is a very good example for a major modern task for the industry, to inject virtual data in the everyday working life as smoothly and simply to access as possible [Regenbrecht et al., 2005]. Therefore, this thesis should not only target this specific issue, but try to build an example framework of how these problems can be targeted in general as well as for this specific issue.

3.2 Course of Action

In this section, the course of action is developed. A major hypothesis, which describes the above mentioned problem in one sentence will be given. Furthermore, the main research issues that arise of the hypothesis are named and will be tackled in the later chapters of this thesis. Later on, the hypothesis and the evoked research issues can be evaluated. The hypothesis is:

Production workers for CNC Machine need additional data of CNC simulation that enables them to work in a more productive and intuitive way with digital content while being fully aware of their surroundings considering the industrial configuration of the production field (staff security, awareness of dangers,...)

Based on this hypothesis, there are three main research issues, which can be derived from it. These are:

- How can process relevant data understandably be provided to a worker, while he is fully aware of his/her surroundings?
- How can data be provided to a remote device?
- How can CNC simulation and its kinematic chains be represented?

The first question is mainly a question about which hardware and software architectures should be used to present data to a user. The second question targets the *how*. How will the data go from the computer where it is stored to the hardware and software that is chosen. How can this be done in a way, that it is simple to use while the aspect to be easily included in running systems still remains. The last research issue is an aspect regarding the use case of the above mentioned scenario. How can be CNC simulation done with the help of AR or VR and more specific, what is a kinematic chain and how do you mimic the kinematic chain that every CNC machine inhabits to be represented visually. The next chapters will each be dedicated to one of the research issues.
This thesis is going to answer these issues in a separate chapter each and will develop a framework of the results.

4 Kinematics of and Usability of CNC Machines

This thesis tries to show the value of augmented reality and wants to showcase its features in the use of Computer Numeric Control (CNC) machines. This chapter gives a short overview about CNC machines and how they are used in modern production.

CNC machines are machine tools which have the capability of producing high precision work pieces through the use of modern control techniques. In a CNC machine, a track for edition of the work piece is set. The machine rapidly checks the relative movement between work piece and working tool and continuously controls the relative movement to correct mistakes or gaps between the actual value and the should be value.

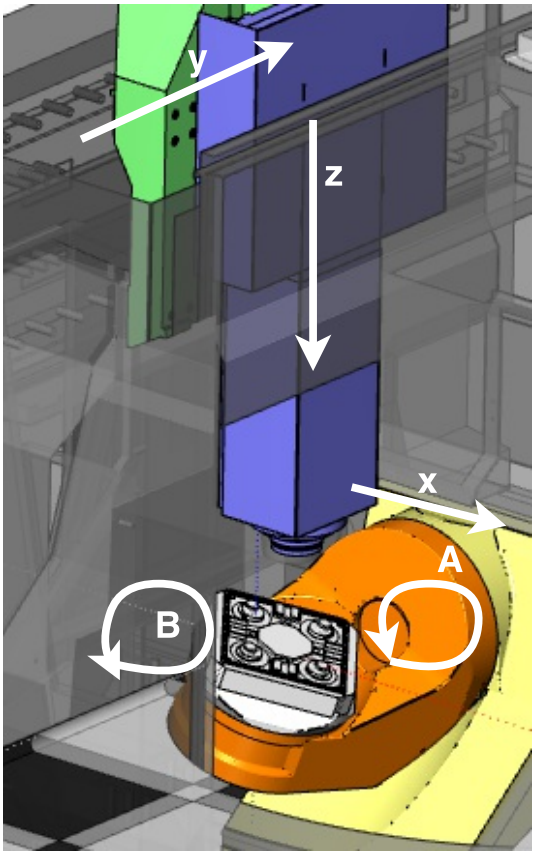

Figure 4.1

CNC machines allow the creation of fast and high quality work pieces that cannot be done equally good manually. Some work pieces that can be produced are not even possible to be created with a handcraft tool machines without adjusting the work piece multiple times. This is because these miss the great freedom grade most CNC machines have. Higher grade CNC machines do not need work pieces to be rearranged and even allow under-cut parts. Different types of CNC machines have different axis configurations or amount of axis and therefore freedome grades [Altintas, 2012]. For example some machines work with head rotation, while other work with a rotating table.

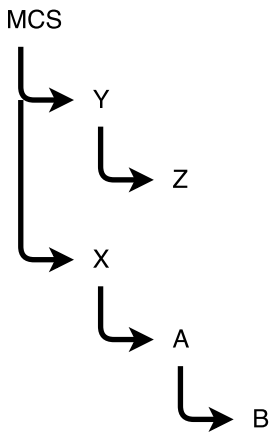

Figure 4.2: Model of a CNC machine with its axis and its according coordinate system

In figure 4.1, you can see a 5-axis machine. The different colored parts can move along their axis into the according axis direction, namely X, Y and Z. There are two more axis, A and B. These are rotation axis and allow the machine to rotate the according part of the machine. The relevant coordinate system to the machine is presented in 4.2. The machine coordinate system (MCS) is root of the linked coordinate systems. Every work tool of the machine has its own coordinate system and can have children. The children are always moving relatively to their position in their parents' coordinate system. Through this, a chain is created. If axis X is moved, the table, which consists of axis A and B, moves together with it. This is called a kinematic chain.

4.1 Work Flow

CNC machine production has a very straight forward work flow. The work piece that will be created is designed as a 3D model in a *Computer Aided Design* (CAD) software. Afterwards the designed 3D model can be used inside a *Computer Aided Manufacturing* (CAM) software. This CAM software translates the model data into different working steps and paths the machine will take. Afterwards it compiles these steps into so called G-Code. G-Code was developed in the 1950s and is one of the major implementations for CNC machine code. It consists of path control information that contain the path itself as well as the speed of the machine tool for each step.
This specific code can be read by the CNC machines. Most modern CAM software simulate the working process beforehand. With this simulation the user can obtain information about the actual work of the machine. A benefit of this is, that it makes error detection easier and prevents malformed models. The simulation should be done in an AR or VR environment with framework which is developed in this thesis.

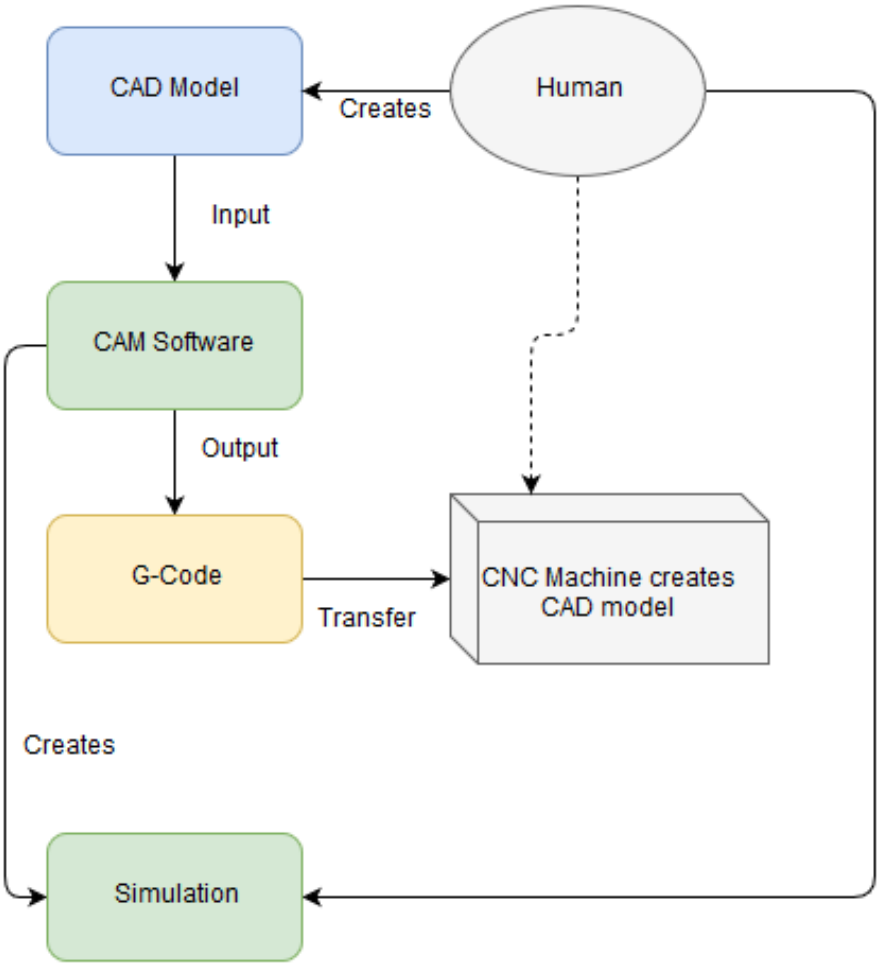

Figure 4.3: From a user created CAD model to the machine process

4.2 Simulation

What is probably the most used component regarding CAM simulation is currently developed by the company ModuleWorks. A simulated machine is seen in 4.5. While simulation can be done in different ways and input data can be chosen differently, due to its status as nearly industry standard, this thesis will show how simulation with ModuleWorks is working. As input data you chose a so called machine definition, which consists of a XML file, multiple 3D models and textures. This XML file contains all the different axis of the CNC machine. They are already stacked as parents and children and therefore inhabit the kinematic chain. Every part of the machine includes its type, if it can move, how it can move (rotation or translation), the boundaries of the machine part and more. The part also references the model file which represents a 3D model of the machine part as seen in 4.4.

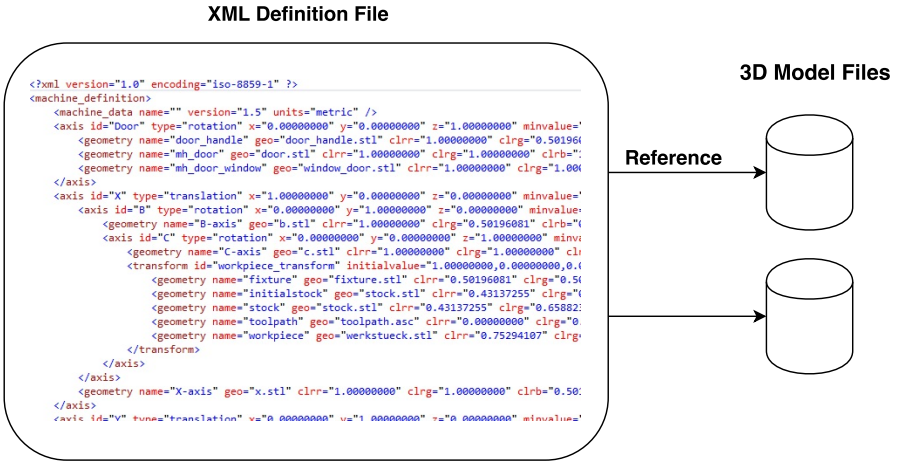

Figure 4.4: The machine definiton

Every referenced machine part is a unique 3D model of one part of the machine like figure 4.5. These models are saved as STL files, which is a simple container format for 3D objects. STL files are widely used for 3D printing and is the standard for CAM as well. For the fact that STL files only describe the geometry of an object, textures or colors for the machine parts are defined in the machine definition XML. Furthermore STL files do not hold any information about the scale of the models. This must be taken from the machine definition, too. For the simulation the XML is parsed by the simulation software, the axis loaded accordingly and the 3D models positioned at the named points with their assigned color.

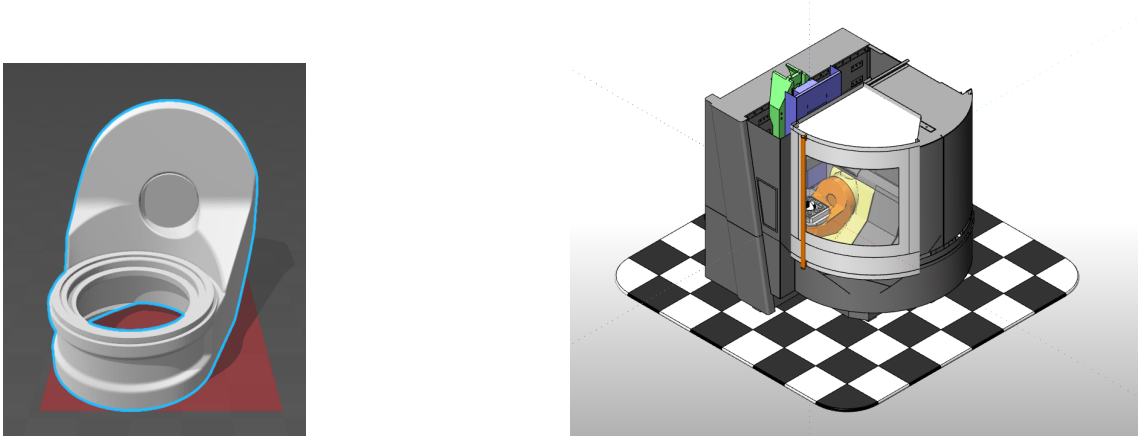

Figure 4.5: A single machine part and the complete machine definition parsed and loaded

5 System Specifications

5.1 Requirements

In this chapter, the basic requirements for the framework are defined. First off, it is good to visualize how the framework that is developed in thesis could be used. For this purpose, some use cases can be derived from the hypothesis above. These will help to define the requirements and needs for the framework, in the general direction as well as the specific defined for CNC machine production simulation.

The production simulation can be done before the actual work process, no matter where and when. No complex beamer infrastructure should be needed, but the simulation is visualized everywhere where a worker wants to visualize it to grant a maximum grade of freedom. The visualization process will give workers better knowledge of complex working processes inside a CNC machine, beforehand as well as during the working process. Through this the user can "take a look inside the machine" without further knowledge of visualization or holographic devices. The simulation can be played in a small size as well as in real size. Real size will leave an even better immersion for the observing person. The users can transfer simulation data as well as files or information to the device.

Based on this story, with the additional information collected in the previous chapter, the requirements for the core framework were defined. Furthermore, the specific needs for CNC machines were developed. Due to limitations and time restrictions of a bachelor thesis, some notes were included that restrict the desired requirements in a way that it is still possible to do it in the time frame of a bachelor thesis. Later on, the framework can be developed further.

Due to the time restrictions mentioned above, there is the need for rapid prototyping regarding the visualization. Rapid Prototyping is a method from software development, which leads to fast usable results of a software. While this software does not need to be bug free or ready for production, this is used to prove the feasibility of a targeted software solution. With this software based as development core, even in the early stages of a software development can problems and requirements be examined. These can be translated to a final solution. This technique is often used in agile programming methods [Connell and Shafer, 1989]. While the framework itself should not be a prototype, it is still important that it can be avoided to write a separate engine for visualizing 3D models.

Non-functional Requirements	Notes
Architecture Independent	At least Linux, Mac, Windows supported
Fast, responsive, and reliable	As fast as possible data and information transfer, information should not be lost
Software should be easy to extend	Modular core structure which can be extended and used for similar projects
Easy to use	Without further CG knowledge & well documentated
Awareness of surroundings	Big area, for this thesis it should be sufficient to minimize the GUI overlay
FOV must not be restricted	User must be able to act without restrictions (hands free, ..)
Complex visualization	Avoid simplifications in terms of graphical complexity
Rapid Prototyping for CG part	Due to time restrictions of a bachelor thesis,the focus is on the framework not on the visualization itself
Inclusion with environment	Holograms should not be floating but anchored. Therefore, some kind of spatial mapping is necessary.

Functional Requirements	Notes
Parse Machine Definitions	XML to graphical representation
Load 3D Models	For this case only STL models
Allow basic control of loaded meshes inside the framework	Color, Alpha, Name, ...
Load boundaries of translation and rotation axis	These need to be parsed of the xml and used in the framework to limit the meshes movement
Control specifc axis after creation	Allow to move or rotate a specific axis of the machine, from the server and the framework
Move and re size CNC Machine model	Allow user to move the model in a simple manner
Transform kinematic chain to cg representation	Allow complex kinematic chains to be modeled inside the framework

Table 5.1: Requirement definition for the Software System

5.2 Wearable Devices

In this section, the choice of the device is discussed. To this date there are different wearable devices on the market, which can inject digital data into the field of view (FOV) from a user. They have different capabilities regarding their mechanics that present content to the user. While there exist not only wearable devices, these are the only ones, which are compared because of the specific requirement that a production worker needs to work without restrictions. This basically means he cannot be tied to a specific area or place.
First thing that comes to mind are glasses. Due to their nature, all glasses are usable without restrictions. Therefore the "Google Glass', "Microsoft Hololens" and the "HTC Vive" are taken into account for this selection.

VR devices mostly have big graphical capabilities but suffer from a major problem for this work, which is the missing interaction with environment. Although the environment can be used inside the virtual world, as the HTC Vive presents with its sensor technology but no direct visual feedback from the environment is given. These missing anchor points make it hard to have a feeling for scale and tend to make the users experience motion sickness. Additionally it would not be safe for the use in mechanical production because it overwrites the users FOV and takes away their awareness of surroundings. This can be a great danger in mechanical production areas. Another big concern is that the Vive can be only used in combination with an additional computer. It must be plugged in and therefore limits the movement radius. A major advantage of the HTC Vive it that it can populate the complete field of vision with digital content.

The Google Glass has very good usability and in contrast to the HTC Vive it does not replace the natural FOV but allows to add additional content to it. It is an AR device. Due to its lightweight construction type, the Google Glass the Glass is very restricted it its graphics power. Furthermore, the data its provides is only a data overlay, which is not integrated into the real environment. Therefore interactions with the real world are not possible. What should not be overlooked, is that the screen size of the Google Glass is very small and would make watching a simulation very exhausting.

The Microsoft (MS) Hololens bridges this gap. It allows complete inclusion of graphics into the environment because it is the only chosen competitor that supports spatial mapping. Spatial Mapping is the process of scanning the environment and creating a polygonal mesh that can be used to be interacted with. Therefore the interaction with environment is very direct and it seems as if the holograms actually can interact with the real world. Furthermore, the Hololens supports gesture recognition, which can be used perfectly for moving and scaling holograms. The MS Hololens does not restrict the FOV. Nevertheless, holographic content can only be visualized in a predefined space of the FOV and not in the full natural field of vision.

While the screen is smaller then the whole FOV, the user can still see through parts of the screen which cannot be animated.

In regards to inclusion, only mobile devices could compete with the Hololens. They are also capable of producing high quality graphics content and special AR frameworks for Android or IOS allow some reduced form of mapping as well. Due to their missing sensor techniques, the spatial mapping can only use data from one camera, which makes the scanning not very reliable and often dimensions are wrong. By using a mobile devices the FOV naturally is restricted as well, and the most problematic thing is that at least one hand is occupied with moving the device. Therefore mobile devices seem to be missing some major points the Hololens has as characteristics, as well.
It should be mentioned that there are lots of promising devices announced to this date, but they are not available yet. Therefore, for this work, only published devices can be used;

Requirements	Google Glass	HTC Vive	Microsoft Hololens	Mobile Devices
Unrestricted FOV	+	-	+	o
Screen Size	-	+	o	-
Awareness	+	-	+	o
Restrictionless	+	-	+	-
Complexity	-	+	+	+
Spatial Mapping	-	-	+	o

Table 5.2: Comparison Devices

5.3 Software Systems

Next to the fitting hardware it is very important to have a fitting software system that handles the visualization. There are different systems that provide the complex 3D capabilities necessary for this project, while they are still able to work with AR or VR special features like spatial mapping. Some of the existing software will be compared now and checked for the requirements listed earlier.

Regarding the visualization, there are different options to take. This chapter will begin with a deeper look into modern game engines. While these engines originally were developed to allow fast and multi-architecture game development, they also allow easy and simple visualization. Their implemented physics and graphics engines allow rapid prototyping. Although you cannot modify the engines directly, they have a big range of script support which allows complex programming. Besides they

often have plugin support to expand their native features. They are easy to start with and wrap most complex computer graphic parts, e.g. shaders. There exist a lot of different engines but for this thesis two of the main competitors were chosen: *Unity* and the *Unreal Engine*.
Both cover all advantages from game engines mentioned above and are widely used in modern game development. Nevertheless, the Unreal Engine does not work on every platform. Currently, it is not working with the Universal Windows Platform (UWP) run-time environment. Unity has some additional features for working with AR. Spatial mapping support and gesture control work on the fly. The Unreal Engine misses these features. Besides the Unity engine has natural native support for the Hololens, which is a big plus point for Unity.

If you want to dive deeper into the graphical aspects of development with AR, you could take a more grass root development approach. This could be done with plain OpenGL or DirectX. OpenGL is a application programming interface, that works across different platforms and programming languages. It is extensively used in a variety of computer software aspects, as for example video games or CAD. Due to its nature it gives the developer full control over every aspect of visualization. It is probably the fastest solution because a lot of overhead can be skipped. Because of its complex usage, lots of implementation and maintenance is necessary, which needs a lot of time, before visualization is possible [Shreiner et al., 2009]. OpenGL has no implemented AR features, but a very complex software architecture. There exist a lot of open-source AR-frameworks. A plain OpenGL application can be extended with these frameworks to gain AR capabilites. These frameworks have a wide range of functions, but currently have no native support for the Hololens [Wagner et al., 2003]. Even if they allow to use the wide range of OpenGL features and can be optimized for the developers wishes, they still cannot offer the native spatial mapping support that Unity allows.

Requirements	Unity	AR Frameworks	Unreal	OpenGL
Rapid CG Prototyping	+	o	+	-
Complexity	+	+	+	+
Simplicity	+	o	+	-
AR Features	+	+	o	-

Table 5.3: Comparison Software

5.4 Conclusion

In this chapter a comparison of different hardware and software which were able to be used for this thesis is made. The Microsoft Hololens is currently the best usable device. Even if it has small sit-backs like its narrow FOV for holograms, its wide range of features make it the best candidate. It can be hoped that the FOV will increase within the next years. In most of the important categories, the Hololens is practically unbeaten. No other device has an equally good scanning system for the environment and its very intuitive usability gesture recognition is a big asset.

Due to the focus of this work, which wants to solve the above mentioned questions, it is very important that developing a basic graphics engine does not need too much time. With this in mind, only game engines and their potential for rapid prototyping leave room for decision. The comparison between Unreal and the Unity was clearly in favor of Unity. Their native support for the Hololens alone gives Unity a big advantage. Through the support, it allows to access features like spatial mapping or gesture control, which makes it a perfect match.

All in all, the Hololens paired with the Unity engine seems to be the current state of the art of AR development and is a good base to tackle the hypothesis defined in chapter 3.

6 Connectivity & Inclusion

In this chapter the part of the framework which connects the Hololens to a remote system is presented. Problems that appeared during development will be highlighted and an insight into the final implementation is given.

6.1 Remote Connection on MS HoloLens

During early stages of development, several implementation ideas for a connection and transfer system to the Hololens were discussed. Some of these played major roles, these were:

- Server system that runs directly on the Hololens (asp.net or Appache)
- Messaging middle-ware like DDS
- Rest interface
- Direct socket communication with own protocol

During development, it appeared that most ideas needed to be scrapped. While the Hololens runs Windows 10, nevertheless, applications that are developed for the Hololens are restricted to using the Universal Windows Platform [UWP]. UWP is a sandboxed run-time environment that only runs applications written in C# [Garofalo et al., 2013]. UWP has access to most functions of the .net software framework by Microsoft but is partially restricted. Due to the sandbox restrictions of the UWP, there are currently no server systems or messaging systems available that run on the UWP. Therefore, these ideas could not be realized.

However, there exist some REST interfaces for UWP. REST is an interface which allows to post and get data from a remote device, with an URL based access system. While this seems to be a good solution, it is necessary to mention that these also have some restrictions. They only support post data in form of strings and no direct binary upload. Because of this binary would need to be transformed into some format which can be put into a string. This can be done with a serialization as for example the Base64 encoding. This would make a lot of conversions necessary and make the framework slow and hard to be included into other systems. Therefore the final decision was made to use a direct socket communication. Some more problems evoked when it got clear that the normal .net APIs for a simple socket

networking are not working in UWP. A solution could be found by taking a special UWP socket stream. This reduced the allowed data to be transferred to byte arrays [Microsoft, 2017].

Therefore it was necessary to develop a protocol that allows to transfer data and information and manages the packages.

6.2 Protocol

In this chapter, the protocol is explained step by step. The protocol should have the following features:

- A small overhead, for fast transfer
- At least two sending modes for transferring data into memory or onto the hard disk drive [HDD]
- A command for toggling Hololens data transfer
- Commands should be extendable
- Protocol should be able to be implemented on every architecture

As base for the protocol the "Transmission Control Protocol" [TCP] was chosen. The protocol should consist of different parts in the header. Instead of relying on a slow string base message approach, integers will be used to encode the commands. Due to the use of TCP sockets, a stream is used to write data in. From the other side of the stream, data can be read out. Because two sided connection could be useful later on, when data from the Hololens should be sent to the remote computer, two different modes should be implemented. One mode is for sending files and messages, the other one for receiving files and messages on the server side. The other one sends a starting signal to the HoloConnector to send its data to the server.
The server part decides whether the Hololens is allowed to send its data or should queue up its message and send them later on.

The first and only mandatory part of the protocol header is the command part. The sender decodes the command for the device with help of a shared header file. Currently, the framework only needs three commands, one for sending a message, which is stored in the memory. Another one for sending files which are stored on the HDD directly. The last one is for switching the mode from only receiving Hololens to sending and therefore allows the Hololens to transfer its queued data. While this implementation uses only two commands, the header file can be extended just by writing new commands into it. When the protocol header is parsed, a callback with the according command is called, where you can simply switch case over your own implemented commands. To make a fast switch between sending and not sending

mode of the HoloConnector possible, the header is finished if the command is "SND" or "RCV". There is no need to write additional information in the header.

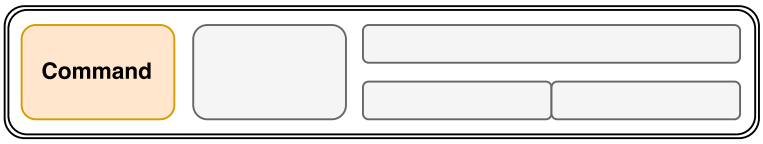

Figure 6.1: The first part of the header, manages the commands

Because every message and every file has different sizes, the next part of the header defines the size of the attached data. This part of the header is only used if the command is a command that indicates a data transfer. Otherwise the header ends here.

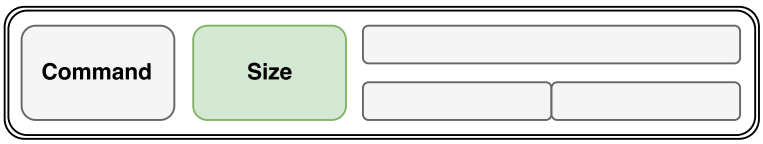

Figure 6.2: The second part of the header, defines the packet size

The third part of the header contains data. If the data consists only of a message, it can be read straight away and stored into memory. No further information is necessary. In the other case when the command, which stores data on the HDD, is transmitted, the header needs to be extended. It will be extended with two new parts, the first one being the file name size of the file which is transferred and the second one the file name. Through this the file can be stored under correct name on the Hololens hard drive.

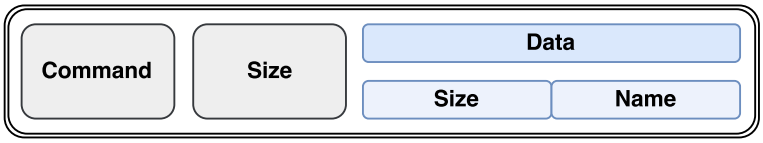

Figure 6.3: The data part of the header can contain additional information

While the whole header is very small, it could be optimized further. In chapter 8, possible optimization will be explained. Figure 6.4 shows the complete header file.

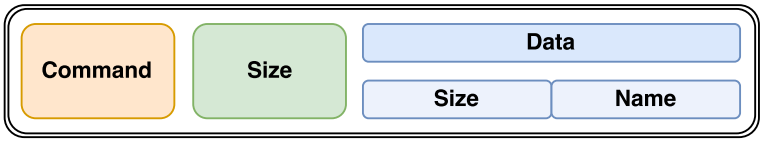

Figure 6.4: The whole protocol header

6.3 Implementation

The theoretical approach for the protocol header above was implemented on the Hololens as well as a server for testing the functionality. On the server side, the connection is not very complicated. The server opens up a socket through the .net class TcpClient, which handles the TCP specific connection setup. The follwing code fragments are part of the class TcpServer.cs, which is part of the HoloConnector framework.

```
1 TcpClient tcp_socket = new TcpClient(ip.Trim(), port);  //Connect to server
2 netstream = tcp_socket.GetStream();                     //Get the stream
3 //Start the protocol header
4 [...]
5 // write post or info data command
6 if(isData) netstream.Write(BitConverter.GetBytes(POST), 0, size_16);
7 else netstream.Write(BitConverter.GetBytes(INFO), 0, size_16);
8 [...]                                                  //Protocol header is written
9 netstream.Write(data, 0, data.Length);                 //Write the data
```

The harder part is to receive the connection on the device side, the Hololens. Because the idea was that any connection from any endpoint to the Hololens is accepted, a socket listener needed to be implemented. This listener waits for connection attempts on a predefined port. As already stated earlier, the Hololens only supports UWP applications [Microsoft, 2017]. Therefore, the normal way of .net programming could not be used. Luckily for UWP applications, there is a provided class which is called "StreamSocketListener", which can handle incoming connections. After the listener registers a connection attempt, it fires a callback and gives this callback to the socket which can be used to read data from the regarding stream. Once the callback is received, a reader for the created socket can be used to wait for incoming data. For the implementation refer to the next two code segments.

```
1 public void StartServer(int port, uint buffer_size)
2 {
3         //get the local ip and resolve as hostname
4         HostName h = new HostName(GetIP());
5         //get a listener
6         StreamSocketListener server = new StreamSocketListener();
7         //set the listener to call the callback "Connection" on setup connection
8         server.ConnectionReceived += (s, e) => Connection(e.Socket);
9         //bind the listener to local ip and a specific port
10        IAsyncAction outstandingAction = server.BindEndpointAsync(h, port);
11 }
```

```
1 private void Connection(StreamSocket socket)
2 {
3         //create new data reader
4         reader = new DataReader(socket.InputStream);
5         //only wait for available data
6         reader.InputStreamOptions = InputStreamOptions.Partial;
7         [...]
8         ReadFromStream(); //Read from stream
9 }
```

The function ReadFromStream waits for available data and parses the data accordingly to the protocol. This happens asynchronously which means the call of the function is not blocking but happens in a separate thread in the background.

```
1  public async void ReadFromStream()
2  {
3          //Start parsing protocol
4          //block until the first byte is received, then read command from stream
5          await reader.LoadAsync(size_16);
6          int command = reader.ReadUInt16();
7          //determine size
8          await reader.LoadAsync(size_32);
9          UInt32 size = reader.ReadUInt32();
10         switch (command)
11         {
12         case POST:
13             string path = await Receive_File(size);
14             callback_post(size, path);
15             break;
16         case INFO:
17             byte[] data2 = await Receive_Data(size);
18             callback_info(size, data2);
19             break;
20         case RCV:
21             callback_rcv();
22             break;
23         }
24         //After one message was read, start reading the next packet
25         ReadFromStream();
26 }
```

Inside this function, three different callbacks can be called. Any class can extend TcpServer.cs and overwrite the callbacks, to manage the received data or to decide how data should be sent.

6.4 Connectivity to existing Systems

A major requirement for the framework was that it is reachable from every computer architecture. Due to the benefits of a raw socket communication, it is extendable to nearly every system. The system which wants to connect to the Hololens just needs to implement the protocol explained above and can communicate via TCP. To show how this could look like, an example implementation was done for a server system running asp.net. The example system allows to upload files to a server with a web interface and then transfer these files to the Hololens with the protocol. The overall interface, which is very basic and just for showcase purposes is seen in 6.5. In (1) you insert the IP of the Hololens device to which you want to connect. Then a connection is established on port 13000. If you choose the browse button (2) you can upload multiple files at once, so a whole machine definition including its models and textures can be transferred and processed at once. Afterwards the size of the loaded hologram can be changed. Besides there exist some control options regarding the kinematic chain.

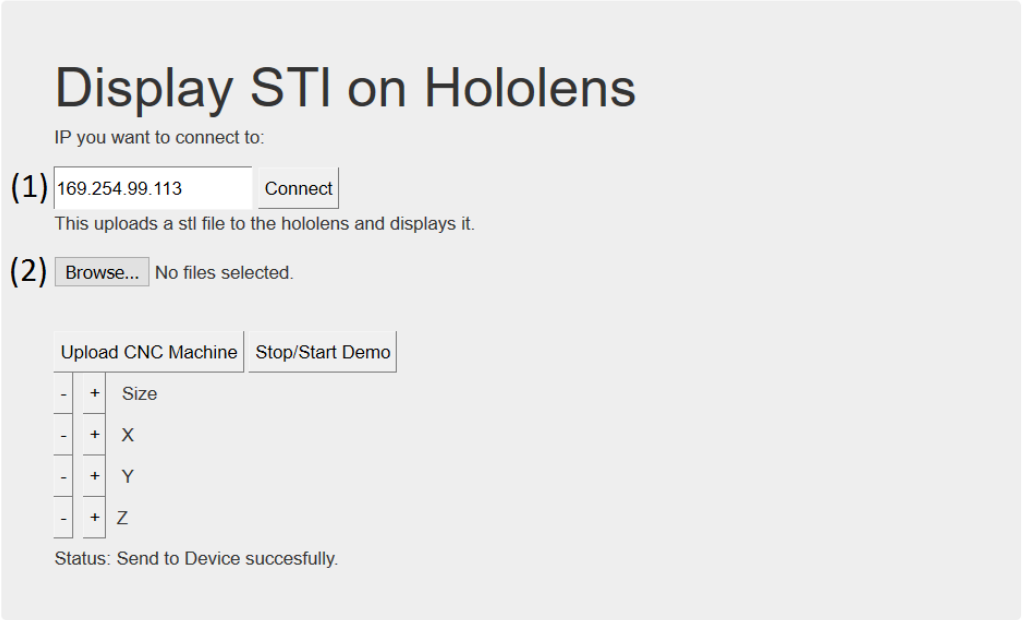

Figure 6.5: Server GUI

7 Visualization

In this chapter, the following research question is examined: *How can CNC simulation and its kinematic chains be represented?*. The question asked here has multiple answers. The chapter *Related Work* already explained basic transformation for machine kinematics. In this chapter the actual implementation and transfer from a machine definition to Unity is presented and the problems that arose with it explained.

7.1 Implementation

For the kinematic chain which is set in the machine definition, the task was now to find a fitting implementation for Unity. To this thesis benefit, the Unity engine already has a structure on which the kinematic chain can easily be based upon. In Unity, every scene is populated with so-called "GameObjects". These are hierarchically structured and can be compared to a scene graph. In a scene graph every object can have children or parents. Every object holds its own transformation matrix. If you manipulate an object, all the child objects of the object are transformed with the same matrix as well [Burns and Osfield, 2004]. In Unity these objects can be the GameObjects.

Unity also makes some functions to translate and rotate GameObjects available. Translations can be done with regard to the local coordinate system of each GameObject or to the global coordinate system. Rotations can be done in regards to the local or the world system, while they either rotate around the center of the model or a predefined point. This scene graph structure takes away some of the struggles mentioned in the last chapter. It is not necessary to calculate specific matrices or translate objects into its parents coordinate system. To understand the structure of the machine, the framework parses the machine definition recursively:

```
1  private GameObject load_model()
2      {
3          //Create a gameobject that holds the complete machine, for later use
4          GameObject machine = new GameObject("root");
5          //Load all axis
6          foreach (machine_definitionAxis a in m.axis)
7          {
8              load_axis(a, machine);
9          }
10         //Now load all single models into unity
```

```
11          foreach (machine_definitionGeometry g in m.geometry)
12          {
13              load_geo(g, machine);
14          }
15          //return the complete parsed cinematic chain
16          return machine;
17      }
18
19  private void load_axis(machine_definitionAxis a, GameObject father)
20      {
21                  //create an own gameobject for this axis
22          GameObject axis = new GameObject(a.id);
23          //attach it to the parents transform
24          axis.transform.parent = father.transform;
25          //attach axis script to it
26                  [...]
27          //load all children recursively
28          foreach (var child in a.Items)
29          {
30              if (child is machine_definitionAxis) load_axis(child, axis);
31              else if (child is machine_definitionGeometry) load_geo(child, axis);
32          }
33      }
```

The basic structure of the kinematic chain is established in that way. Every child's transformation matrix is linked to the parents transformation matrix, while the whole machine has one root object. If the parent is transformed, similar to a scene graph, the child is transformed accordingly. In figure 7.3, the transformation from machine definition to Unity is visualized.

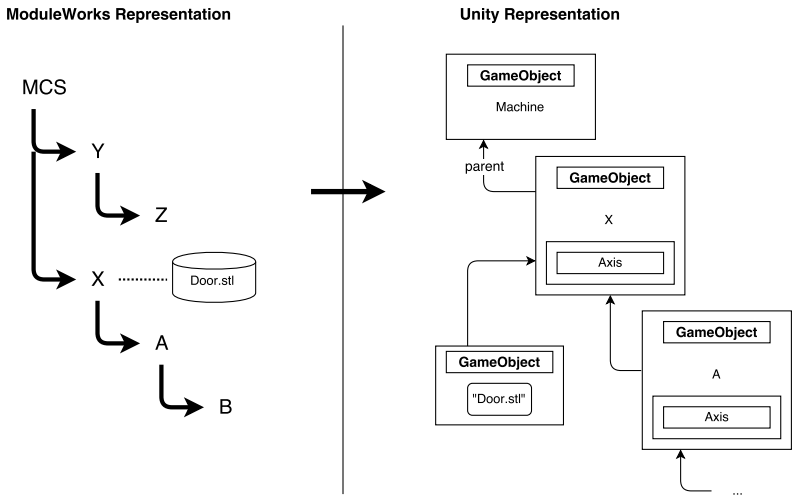

Figure 7.1: Transfer of the kinematic chain

While this works perfectly with translation commands, there did occur some strange behavior regarding rotation of machine parts.
The problem is caused by the input data. Normally, when creating a 3D model, it is centered in the origin $(0, 0, 0)$. This is handled slightly different from ModuleWorks. They avoid the need to re-position and store the unique position of the individual

part models. To do so, each model is not modeled in the origin but at the position it would have if the whole machine is set in the same coordinate system, compare figure 7.2.

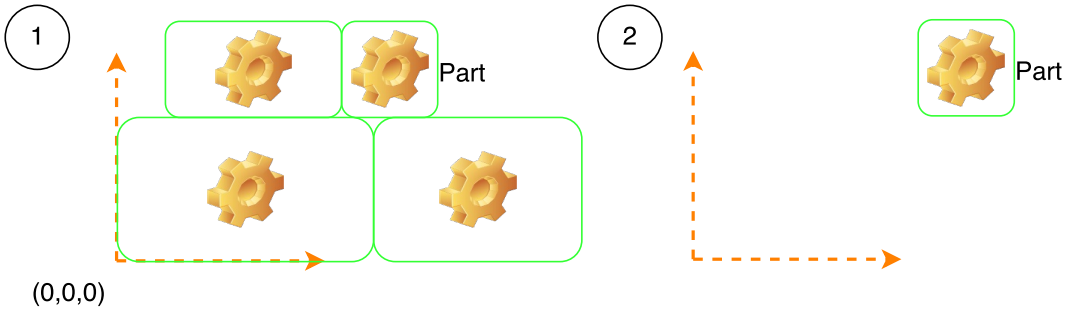

Figure 7.2: How the weird placement happens

Nevertheless this makes one more information necessary to be stored in the machine definition, the rotation axis. Normally in CG, if you rotate an object, you either rotate it around the world origin or translate it into the origin to rotate it around its center. But most of the machine parts should not rotate around the origin but a specific rotation axis. This can be seen in figure 7.3.

Unity does provide the earlier mentioned function "RotateAround(Vector3 point, Vector3 axis, float angle)", which allows to rotate around a certain point and a specified axis. Normally it would be sufficient to use this function with "point" being the rotation point provided in the machine definition and "axis" the according axis. There is some special case, where this does not work properly. If a machine part, should be rotated around its gravity center, in the machine definition the rotation axis is missing. This is due to the fact, that the object should rotate around its own gravity center while being relatively to its parent object.

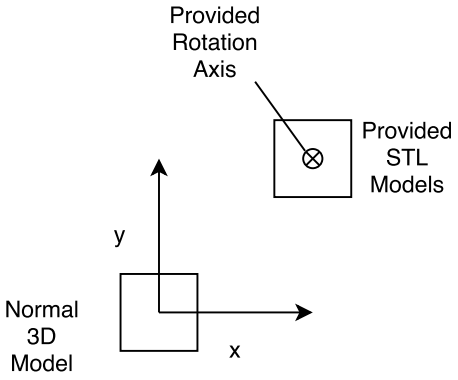

Figure 7.3: Showcase of Model Positioning

For the framework the following solution was developed. In case the rotation point is set as zero, the rotation point, will be the child's calculated gravity center in world coordinates, and the rotation axis is the vector of the machine definition axis, originating from the parent's origin transformed to world space. By this a rotation around the correct point and axis can be done, even if the parent itself is moved or rotated.

8 Evaluation

In this chapter of the thesis, the developed system is evaluated. Furthermore, the resulting system is presented. Concerning aims that were not reached, this chapter discusses why this happened and how this could be so solved in the future.
Last but not least, it discusses if all research question were answered and if any statement about the correctness of hypothesis can be made.

8.1 Working System

In this section, the framework architecture is explained and its major classes be presented. In figure 8.2 you can see a simplified diagram of the complete software framework that was written for this thesis.

There are five main protagonists who are colored in orange and red. The user wants to display and interact with the Hololens to view holograms of CNC machine simulation. The user can upload files onto a server. There the files will be converted into byte which can be transferred via the developed protocol. The classes which transfer and convert the files, can be easily extended and reused.

On the Hololens side Unity is started which takes care of the visualization and loads the different classes of the framework. Inside of Unity, the framework called "HoloConnector" is running in the form of different script classes. The class "TcpServer" is an extendable class which starts a listening TcpServer on side of the Hololens. It takes care of incoming connections. By extending the class, the different callbacks that are called when data is transferred can be overwritten and suited to the individual needs.
"StaticMeshLoader" is a static class, which allows to load all mesh formats that Unity supports, colorize the mesh, position it and if desired, assign the resulting GameObject a parent. To support the loading of STL files, which are by default not supported by Unity, a slightly modified version of the open source script [KARL, 2016] is used.

"TcpHandler" is the core part. It extends "TcpServer" and can parse the XML files of the received machine definitions, to create the correct kinematic chain. It creates GameObjects for the meshes and a handler class called "Axis" for each axis, which handles the axis movements. A script called "TapToPlace" from the HoloToolkit [MICROSOFT, 2016a], is put on the root object of the parsed machine. This enables the user to gaze at the hologram, perform a click gesture in the air and

move the hologram to other positions on the floor.
While Unity performs the visualization, the user is able to move specific axis via the server's web-interface as well as change the size of the simulated machine.

Figure 8.1: The running system

The next sections evaluates how the according system answers the research questions and if the target of the thesis was reached.

8.2 Hard & Software

Research Issue: How can process relevant data understandably be provided to a worker, while he is fully aware of his/her surroundings?

To answer this question, different devices were compared. As already introduced in chapter 2, an AR device was chosen. A modern game engine was selected to be used with the Hololens. For a more individual graphic approach a more grass root approach could be taken by using plain OpenGL or DirectX and implement the AR features on your own. With regards to the time restrictions, it seems as the choice of taking Unity was correct. The framework enables to visualize holograms and with different mechanisms, simulation data too, while the Hololens does not restrict the field of view. It does not restrict the movements or usability of a production worker. Furthermore in chapter 5 requirements for the frameworks were defined. The table 8.1 will evaluate if these requirements were fulfilled.

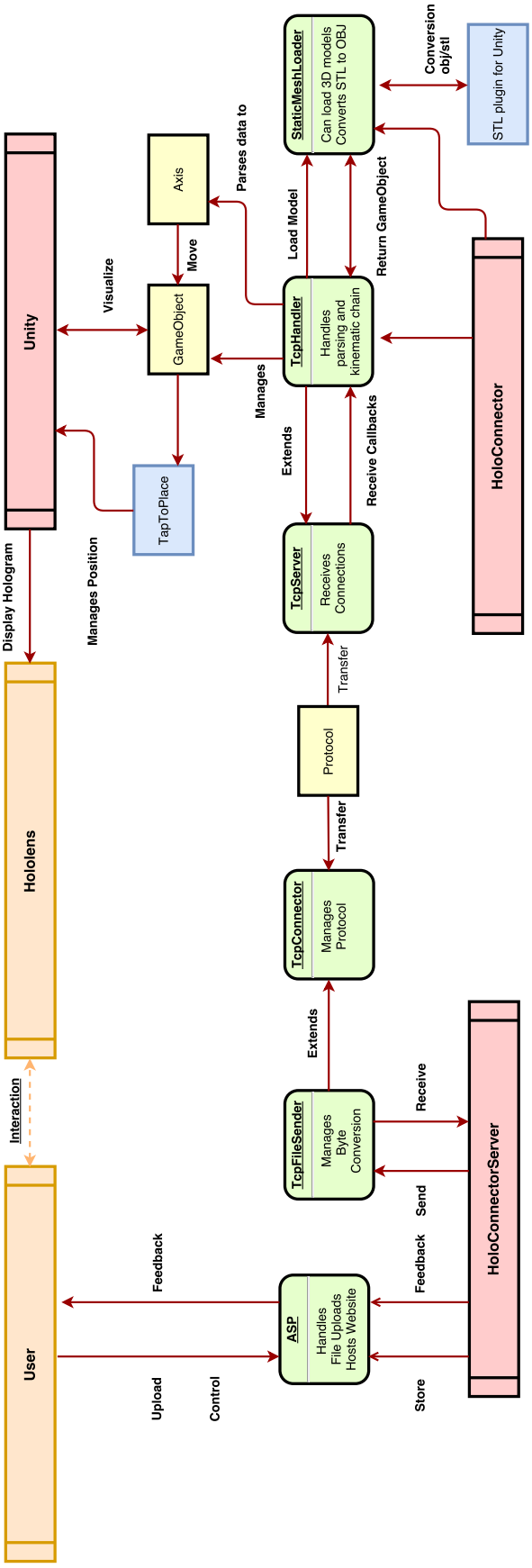

Figure 8.2: The system architecture

Non-functional Requirements	Fulfilled	Explanation
Architecture Independent	Yes	Connection is based on TCP which is architecture independent
Fast, responsive, and reliable	Partially	Due to TCP its reliable, for fast confer section 8.5
Software should be easy to extend	Yes	A modular structure was used; Classes can be extended
Easy to use	Maybe	Needs user tests
Awareness of surroundings	Yes	Hololens is a see-through device and not cable bound
FOV must not be restricted	Yes	See above
Complex visualization	Yes	Unity allows complex visualization
Rapid Prototyping for CG part	Yes	Is granted through Unity
Inclusion with environment	Yes	Anchor and Tap to Place from HoloToolkit are used

Functional Requirements	Fulfilled	Explanation
Parse Machine Definitions	Yes	A recursive parser class was written
Load 3D Models	Yes	*StaticMeshLoader* class can load mostly all mesh files (including STL)
Allow basic control of loaded meshes inside the framework	Yes	*StaticMeshLoader* supports basic control
Load boundaries of translation and rotation axis	Yes	The *Axis* class handles boundaries
Control specific axis after creation	Yes	The *Axis* class can control specific axis
Move and re-size CNC Machine model	Yes	The *TcpHandler* class allows change of the size via the server interface
Transform kinematic chain to cg representation	Yes	Compare chapter 7

Table 8.1: Most of the requirements were fulfilled

8.3 Communication

Research Issue: How can data be provided to a remote device?

In this section different approaches were tested, with the result that most ideas that were thought of during development are not possible due to UWP restrictions. While the existing solution certainly proves that data can be provided to the Hololens as the chosen device, the protocol can certainly be improved. Nevertheless, the advantages of a TCP socket based approach, guarantee that commands and data can be sent from any system that has access to TCP sockets, so basically every computer architecture. In summary you can say that an answer to the research question was found.

In first versions of the protocol was unnecessary header information like IDs or information about the following packages. These could be stripped, because TCP already handles this information. Therefore they were duplicates. An idea to accelerate the protocol in the future, would be to switch from an integer based approach to a bit based solution to reduce the protocol overhead. While this makes parsing more complicated and should only be used for specialized solutions, a smaller header could be achieved. The remaining question is, how fast is the current approach, how reliable the transfer is and if this is sufficient for the use-case. While the transfer of models takes a lot of time, the transfer of messages takes ¿1 second. For testing this is fast enough but for a real time simulation on a remote system with continuous transfer of transformation data, it could be too slow. Section 8.5 dives deeper into this topic.

Due to the use of TCP, the framework benefits from all advantages from TCP. These are reliability, error-check and order as well as full-duplex. This means data can be sent and received simultaneously. Therefore the protocol which sits on top of TCP does not need to handle these critical properties. For that reason the necessary amount of header information is very small and allows a fast transfer. This allows to have minimal maintenance points, while still using a highly functional transfer way.

8.4 Kinematic Chain

Research Issue: How can CNC simulation and its kinematic chains be represented?

Through Unity and a scene graph kind of structure, a way could be found on how kinematic chains can be represented accordingly in a visualization. Little problems that occurred with odd placements could be solved. And a fully functional one to one transition is possible. For this, a parser was written that can handle machine definitions and different classes that can handle axis and their correct movements.

In the end you can say, a way was found to represent kinematic chains and simulate CNC machines. While the simulation itself should not happen on the device but rather on a remote computer that sends the correct movement pattern either beforehand or on the fly over the communication system.

8.5 Speed of the Framework

It is hard to evaluate the speed of such a complex system without writing multiple complex tests or doing time consuming user research. Therefore, the resulting speed can only be approximated. In the requirements, it was stated that the system should be as fast as possible. A good measurement is the availability of real-time computing (RTC). RTC describes time between event and response in regards to a constraint. These constraints are also called dead lines. The system needs to guarantee a response until this time threshold is reached. In simulation this describes the term that the simulation clock is running at the same speed as the actual clock and therefore showcasing a simulation without delay.
Also there exists the term "nearly real-time" (NRT), which describes that a delay is not significant enough to be considered [Liu].

There are 3 different bottle necks, which have to be looked at when you want to see how fast the framework is:

- The connection
- The simulation
- Loading a CNC machine

The connection is based on TCP. TCP generally is not considered to be RTC ready. Because of its guarantee of delivery and sorting of out-of-order messages, sometimes a relatively long delay can happen. There exist some protocols like the "Real-time Transport Protocol" which is based on UDP and could be used if a RTC connection should be established. Right now, the transmission for small packets could be called NRT because the chance of delay is very minor for them, while bigger packets which make faulty transmissions more likely could take more time.

Regarding the simulation, a pre-compiled simulation which only needs to be played is easily manageable for Unity in RT. Where it gets critical is when the commands for the simulation are transferred via the protocol on-the-fly. From the point where the message has been sent, the actual movement of the part is only done when the global "Update" function is called. Considering the fact that Unity's Update is called every frame, as soon as the frame rate drops, the speed gets slower. Nevertheless, there exists a FixedUpdate which can be used to do updates while ignoring the frame rate. To make the simulation more consistent, it would be necessary to use the FixedUpdate function with a reasonable time value. Therefore, RTC could

be reached for this part.

Because every transmitted CNC machine needs to be parsed recursively, this happens really slow depending on the machine definition size. Loading a CNC machine, therefore, cannot be done in RTC.

All in all, the transmission of CNC machine files is not even NRT, while small messages which are sent to control the machine are probably around NRT.

8.6 Evaluation of the Hypothesis

While the research issues are all answered, it remains to be seen if the hypothesis is correct. For this, further development and especially field testing is necessary.

9 Summary and future work

In this chapter the thesis is summarized and a little outlook will be given on how this framework could be extended into a software solution and what other possible field of use it could have.

The current development in AR and VR makes new kinds of visualization possible. Wearable devices are so far developed to this date that inclusion in production and private environments can realistically be thought of. This thesis started with explaining the current state of the art of AR/VR and MR. Different terminologies and important science topics for this thesis are explained.
The motivation for this work was to optimize the CNC machine production processes in a visual way. Therefore, the following hypothesis was built:

> **Production workers for CNC Machine need additional data of CNC simulation that enables them to work in a more productive and intuitive way with digital content while being fully aware of their surroundings considering the industrial configuration of the production field (staff security, awareness of dangers,...)**

Afterwards, three research issues were extracted from this hypothesis. They were examined in the subsequent chapters and evaluated. To make this thesis easier to understand a concise introduction to CNC machines was given beforehand and their value for modern production facilities underlined.
The first research issue was: *How can process relevant data understandably be provided to a worker, while he is fully aware of his/her surroundings?*. To explore the answer to this question, different wearable devices for visualization were compared. Most of the modern devices either rely on VR and therefore do not guarantee the awareness which is necessary for working in production or their AR features are not far enough developed. The result was that only the Microsoft HoloLens allows the maximum flexibility in terms of heavy weight graphic capability, while the worker still remains with his full field of vision and can be aware of the production field. After choosing the best working hardware, it became relevant to look for the best fitting software architecture. The software comparison was not that clear. While current AR frameworks or plain OpenGL could build a solid base, the decision was to take the Unity Game Engine because of its native HoloLens support that enables spatial mapping. Furthermore, its game engine features and scripting capabilities

allow rapid prototyping in terms of graphic engine.

With the Hololens and Unity as system, the next research issue that was tackled, was how the connectivity from any remote system onto this new System could be realized. It is important to transfer simulation data on the fly, as well as information or transfer complex data, like 3D models to the system. Different approaches were taken into consideration. Most of them failed due to the UWP restrictions. After multiple faulty approaches, lastly a plain TCP socket connection was selected. Because of this decision, a new requirement was added to the list. A protocol which sits on top of TCP needed to be developed. Through the next section of the thesis a protocol was developed that allows file transfer as well as message sending to the Hololens. Due to the properties of TCP, this protocol can be implemented on every system and allows TCP streams from every architecture (Windows, Mac, Linux, Android, iOS,..). At the end of the chapter, a basic implementation for the protocol was presented. A windows asp.net server solution which allows machine model transfer and basic simulation control from a web server.

The last research questions determines the implementation of the kinematic chain. The kinematic chain was shortly explained in chapter 4, this specific chapter tried to solve the task of representing the kinematic chain as computer graphic representation. The first sections explain basic CG transformations like rotation or translation of vectors and matrices. Besides it explains how these can be combined. In the end, the actual implementation of the kinematic chain in Unity is explained. The theoretical approach did not work immediately. Different problems evoked because of the input data of CNC machine production simulation files. These could be solved and a solution was presented.

The last chapter before this summary presented the framework in use and evaluated the resulting software solution. It explored if the research issues are answered in a proper way. Arisen problems are documented. All in all, you can say the evaluation gives the impression that the resulting software is a good answer to the research issues that should be solved. The evaluation points out, that there exist some other solutions and how the current solution could be accelerated, while explaining the difficulties of these improvements. Different examinations regarding the real time capabilities are made, but not enough data is collected to make a statement. Even if the Hololens seems to be the perfect device, it suffers from teething troubles.

In the end, you can say this thesis succeeded in proving that the hypothesis could already be answered today. The resulting software may be a prototype, but it showcases well what potential AR and especially the Hololens have for modern production. This question leaves enough material, to devote a whole lot of theses to it.
There are different ways in how this framework could be used and further developed. It could be transferred to different areas that have use for holographic data visual-

ization from a remote system, too. Some examples would be factory design, where you visit a factory and evaluate moving ways before actually putting the machines into the factory. Or interior design, rooms could be furnished on the fly. Industry design could benefit from an easy approach to visual prototyping, while having the opportunity to save a lot of time and material. The value of these devices for the industry will probably increase a lot when the field of view of wearable devices is made wider. The prototype could be developed into a working software solution. It could be transferred to other systems that support Unity as for example Android devices or computers. There is a given chance that wearable AR devices will revolt the current work flows and data visualization in the industry and be a big part in the new industrial revolution.

Bibliography

[Altintas, 2012] Altintas, Yusuf (2012). *Manufacturing automation: metal cutting mechanics, machine tool vibrations, and CNC design*. Cambridge university press.

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

[Bohez, 2002] Bohez, Erik LJ (2002). *Five-axis milling machine tool kinematic chain design and analysis*. International Journal of Machine Tools and Manufacture, 42(4):505–520.

[Burns and Osfield, 2004] Burns, Don and R. Osfield (2004). *Tutorial: open scene graph A: introduction tutorial: open scene graph B: examples and applications*. In *Virtual Reality, 2004. Proceedings. IEEE*, pp. 265–265. IEEE.

[Connell and Shafer, 1989] Connell, John L and L. Shafer (1989). *Structured rapid prototyping: an evolutionary approach to software development*. Yourdon Press.

[Garofalo et al., 2013] Garofalo, Emanuele, A. Liccardi and M. Aponte (2013). *Windows Runtime Environment*, pp. 31–72. Apress, Berkeley, CA.

[karl, 2016] karl (2016). *pbStl*. GitHub Repository. Retrieved February 15, 2017, from https://github.com/karl-/pbStl.

[Kiswanto and Ariansyah, 2013] Kiswanto, Gandjar and D. Ariansyah (2013). *Development of Augmented Reality (AR) for machining simulation of 3-axis CNC milling*. In *Advanced Computer Science and Information Systems (ICACSIS), 2013 International Conference on*, pp. 143–148. IEEE.

[Liu] Liu, Jane WS. *Real-time systems. 2000*.

[Merriam-Webster, 2017] Merriam-Webster (2017). *Hologram*. Website. Retrieved January 23, 2017, from https://www.merriam-webster.com/dictionary/hologram.

[Microsoft, 2016a] Microsoft (2016a). *HoloToolkit-Unity*. GitHub Repository. Retrieved February 15, 2017, from https://github.com/Microsoft/HoloToolkit-Unity.

[Microsoft, 2016b] Microsoft (2016b). *Microsoft HoloLens*. Website. Retrieved January 23, 2017, from https://www.microsoft.com/microsoft-hololens/en-us.

[Microsoft, 2017] Microsoft (2017). *Sockets*. Website. Retrieved January 23, 2017, from https://msdn.microsoft.com/de-de/windows/uwp/networking/sockets.

[Milgram et al., 1995] Milgram, Paul, H. Takemura, A. Utsumi and F. Kishino (1995). *Augmented reality: A class of displays on the reality-virtuality continuum*. In *Photonics for industrial applications*, pp. 282–292. International Society for Optics and Photonics.

[Ohta and Tamura, 2014] Ohta, Yuichi and H. Tamura (2014). *Mixed Reality: Merging Real and Virtual Worlds*. Springer Publishing Company, Incorporated, 1 ed.

[Olwal et al., 2008] Olwal, Alex, J. Gustafsson and C. Lindfors (2008). *Spatial augmented reality on industrial CNC-machines*. In *Electronic Imaging 2008*, pp. 680409–680409. International Society for Optics and Photonics.

[Paul, 1981] Paul, Richard P (1981). *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul.

[Perkins et al., 1997] Perkins, Roderick, D. S. Keller and F. Ludolph (1997). *Inventing the Lisa User Interface*. interactions, 4(1):40–53.

[Regenbrecht et al., 2005] Regenbrecht, Holger, G. Baratoff and W. Wilke (2005). *Augmented reality projects in the automotive and aerospace industries*. IEEE Computer Graphics and Applications, 25(6):48–56.

[Rheingold, 1991] Rheingold, Howard (1991). *Virtual Reality: Exploring the Brave New Technologies*. Simon & Schuster Adult Publishing Group.

[Shreiner et al., 2009] Shreiner, Dave, B. T. K. O. A. W. Group et al. (2009). *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*. Pearson Education.

[Tanenbaum et al., 2003] Tanenbaum, Andrew S et al. (2003). *Computer networks, 4-th edition*. ed: Prentice Hall.

[Uicker et al., 2011] Uicker, John Joseph, G. R. Pennock, J. E. Shigley et al. (2011). *Theory of machines and mechanisms*, vol. 1. Oxford University Press New York.

[Wagner et al., 2003] Wagner, Daniel, D. Schmalstieg et al. (2003). *First steps towards handheld augmented reality*. In *ISWC*, vol. 3, p. 127.

[Zhang et al., 2010] Zhang, J., S. K. Ong and A. Y. C. Nee (2010). *Development of an AR system achieving in situ machining simulation on a 3-axis CNC machine*. Computer Animation and Virtual Worlds, 21(2):103–115.