FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

# ASSIGNMENT OF BACHELOR'S THESIS

| | |
|---|---|
| **Title:** | HoloCopy - 3D copy with Hololens |
| **Student:** | Anna Zderadičková |
| **Supervisor:** | doc. Ing. Tomáš Pajdla, Ph.D. |
| **Study Programme:** | Informatics |
| **Study Branch:** | Web and Software Engineering |
| **Department:** | Department of Software Engineering |
| **Validity:** | Until the end of summer semester 2019/20 |

## Instructions

1) Study literature about Hololens and 3D model visualization, e.g. [1,2,3,4] and references therein as well as about 3D reconstruction from photographs, e.g. [5,6,7].
2) Propose and implement a system for automatic image acquisition by Hololens, 3D model computation on a server, communication between the server and Hololens, model visualization and Hololens user interface for acquisition control, visualization and object manipulation. Study approaches for efficient model representation and visualization and implement an approach suitable for Hololens.
3) Demonstrate the system on a real device.

## References


[1] M Garon et al.: Real-time High Resolution 3D Data on the HoloLens. 2016
[2] M Joachimczak et al.: Real-time mixed-reality telepresence via 3D reconstruction with HoloLens and commodity depth sensors. ICMI 2017 DOI:10.1145/3136755.3143031.
[3] S Orts-Escolano et al. Holoportation: Virtual 3D Teleportation in Real-time. UIST '16, 2016
[4] S Dong, et al.: Real-Time Re-textured Geometry Modeling Using Microsoft HoloLens.
[5] JL Schonberger et al.: Structure-from-motion revisited. CVPR 2016
[6] A Locher et al.: Progressive 3D Modeling All the Way. 3DV 2016
[7] AliceVision. https://alicevision.github.io/


Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 16, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

# HoloCopy - 3D copy with Hololens

***Anna Zderadičková***

Department of Software Engineering
Supervisor: doc. Ing. Tomáš Pajdla, Ph.D.

May 15, 2019

# Acknowledgements

I would like to thank my supervisor, doc. Ing. Tomáš Pajdla, Ph.D., for giving me the opportunity to work on this interesting topic.

# Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on May 15, 2019 .....

Czech Technical University in Prague
Faculty of Information Technology


### Citation of this thesis

Zderadičková, Anna. *HoloCopy - 3D copy with Hololens.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

# Abstract

This thesis concentrates on creating a system for a 3D model reconstruction of an object in augmented reality. The aim of this work is to create an application for Microsoft HoloLens for automatic acquisition of images and visualization of a reconstructed model and a server program that communicates with HoloLens and mediates the reconstruction. Unity Engine was used for the creation of the application because it supports the development for virtual and augmented reality. The server is created using the Python Flask framework. The 3D reconstruction is done in the COLMAP pipeline. The final system is useful for Microsoft HoloLens users that are interested in creating 3D models from a real world.

**Keywords** Microsoft HoloLens, 3D reconstruction, 3D models, AR, augmented reality, Unity

# Abstrakt

Tato práce se zabývá vytvořením systému pro 3D rekonstrukci objektu v rozšířené realitě. Cílem práce je vytvořit aplikaci na Microsoft HoloLens pro automatické pořízení fotografií a vizualizaci rekonstrukce modelu a server, který po síti komunikuje s HoloLens a zprostředkuje rekonstrukci. Pro vývoj aplikace byl zvolen herní engine Unity, který podporuje vývoj na virtualní a rozšířenou realitu. Pro server je použit Python framework Flask. 3D rekonstrukce z fotografií je provedena pomocí programu COLMAP. Výsledný systém je prospěšný pro uživatele Microsoft HoloLens zajímající se o tvorbu 3D modelů z reálného světa.

**Klíčová slova** Microsoft HoloLens, 3D rekonstrukce, 3D modely, AR, rozšířená realita, Unity

# Contents

# List of Figures

# List of code demonstration

# Introduction

Augmented and virtual reality is becoming more and more popular during the latest years. It is slowly getting more available to the public. Augmented reality is mostly used on phones, but the restrictions of the small screen of the phone are breaking the immersion. That is why Microsoft HoloLens headset is the ideal device for augmented reality. Having the headset on allows the user to move freely without breaking the immersion.

With virtual environments comes the issue of virtual objects. 3D models are nowadays used almost everywhere. Movies, games, images, commercials... However, 3D modeling is a time consuming task, users spend hours and hours of work to recreate objects in a virtual world.

The topic of this thesis is a system for 3D reconstruction of objects using HoloLens device.

The result of the work will be beneficial for users of HoloLens wanting to automatically recreate objects from their surroundings. That way, they won't have to manually model those object themselves.

I chose this topic because this technology is still evolving, and if used correctly, it will help create 3D models effectively and fast.

In this thesis, I look into the problems of 3D reconstruction from photographs and development on HoloLens. I analyze possible implementations, then design and implement the system for automatic 3D model reconstruction using HoloLens. For the HoloLens application, I used Unity Engine with Microsoft's Mixed Reality Toolkit. The server part is done in Python and reconstruction from photographs is done using COLMAP [1] [2] [3].

The structure of the thesis is as follows. First, this work will focus on the theory behind the system. Next will be described the analysis and design, where users preferences, similar applications, and possible ways of development of the system will be discussed. Then continues the realization part where the implementation of the system will be demonstrated. User testing is included in the realization chapter. After the realization comes the experimental validation, where all experiments that had to be done during the development will be described. The final chapter is about installing and running the system.

Chapter 1

# Goal

The goal of this bachelor thesis is divided into two following categories.

The first category is the theoretical part. The main goal is to study literature about HoloLens, 3D model visualization and 3D reconstruction from photographs, then analyze possible ways of implementation of previously mentioned problems.

The second category is the practical part. There, the system for 3D reconstruction for HoloLens will be proposed and implemented. First, the system will automatically acquire images made by HoloLens. HoloLens will communicate with a server, which will perform a 3D model computation and a reconstruction. And lastly, the 3D model will be visualized on HoloLens and manipulated by the user via the user interface. The system will be demonstrated on a real HoloLens device.

Chapter 2

# State-of-the-art

This chapter explains the technology that this thesis is working with. The first topic to be explained is augmented reality, then HoloLens will be described and a final topic to cover is photogrammetry.

## 2.1 Augmented reality

Augmented reality (also called AR) is *"an enhanced version of reality where live direct or indirect views of physical real-world environments are augmented with superimposed computer-generated images over a user's view of the real-world, thus enhancing one's current perception of reality"* [4]. The most widely used AR is on mobile phones, for example, being games, photography filters or planning and measuring applications. Other most known AR devices are Google Glasses [5] and Microsoft HoloLens [6]. Both devices are hands-free in the form of glasses that are operated using voice or gestures.

Augmented reality is often mistaken for virtual reality (VR). The main difference is that VR places the user in a whole new and fully virtual world, where the user can't see the real world. On the contrary, in AR users can see the real world or the virtual environment is mapped onto the real world.

According to [4] there are four types of augmented reality:

- Marker-based AR
- Markerless AR
- Projection-based AR
- Superimposition-based AR

The marker-based AR requires special visual markers. These markers indicate where the virtual object should be placed and from what position is the AR device looking at them. This is, for example, a QR code, that will display a virtual model, when a user points the AR device on it.

The markerless AR uses sensors like GPS, gyroscope, accelerometer... The AR device is tracking the position. It doesn't need any markers to display a virtual object and knows the position from which is the device looking at it. This is, for example, AR on mobile phones or HoloLens

The projection-based AR is projecting virtual objects to physical objects. Users sometimes may even interact with the projection.

The superimposition-based AR changes the original viewed scene. It either fully replaces it with a virtual scene, that is mapped to the real environment, or it places new objects into the environment.

## 2.2 Microsoft HoloLens

HoloLens is an AR head-mounted display device in the form of smart-glasses created by Microsoft.

Figure 2.1: Microsoft HoloLens 1st generation [7]

The first generation of HoloLens was released in March 2016 [7]. HoloLens 1st generation offers a markerless AR. It tracks the headset position using an IMU (inertial measurement unit), four environment understanding cameras and a depth camera. The visualization of a virtual object is done on see-through holographic lenses (also called waveguides) [8].

User controls the device using hand gestures, voice commands, and a gaze. The gaze refers to the way the user is pointing their head and it indicates with which object the user wants to interact.

The second generation of HoloLens was announced in February 2019 and it is said to be released in coming months [9]. One of the main changes was moving the center of the mass from the front to the center of the device

by placing some of the hardware in the back of the headset. Another great change is increasing the field of view for better immersion. Microsoft also added another way of controlling the device and that's by eye tracking. The movement of eyes is tracked by two IR cameras [10].

The system is developed on the HoloLens 1st generation device.

## 2.3 Photogrammetry

Photogrammetry is a science of calculating geometric information from images. In other words, photogrammetry is extracting 3D information from a set of photographs, most frequently resulting in a 3D reconstruction of the object from the photographs.

According to [11], the most used way to classify types of photogrammetry is based on camera location while taking pictures. There are two main types: aerial photogrammetry and terrestrial photogrammetry.

As the name indicates, in aerial photogrammetry, images are taken in mid-air. The camera is mounted to an aircraft, usually a drone. This method is mostly used for terrain models or a building 3D reconstructions.

In terrestrial photogrammetry (also called close-range) the images are taken on the ground. The camera is either held by the user or mounted on a tripod. It usually results in 3D model reconstruction or measuring of distances, sizes, etc... This is the type of photogrammetry that is used in this thesis.

Chapter 3

# Analysis and design

This chapter focuses on the analysis of similar applications and possible solutions. There is an overview of possible technologies to be used for the system as well as an analysis of user expectations. Requirements for the system are to create an application on HoloLens for image acquisition and model visualization and a server for 3D model reconstruction. Possible solutions for these requirements are discussed in this chapter.

## 3.1 Analysis of similar applications and systems

This part analyses the applications similar to the goal of this thesis. This analysis consists of three main parts. For the first part, an analysis of applications available on the official Microsoft App Store [12] is performed. The second part focuses on similar applications that aren't available at the store. The third part inspects similar applications available on different platforms.

Since HoloLens is a Microsoft product, most of HoloLens applications are available at Microsoft App Store [12]. After further inspection, there weren't found any applications matching the assignment of this thesis.

There are two types of applications as similar as possible available for HoloLens.

The first type is applications that visualize already pre-made models, some of them allowing the user to view and modify them. This is, for example, the Bridge [13]. This application lets users choose a 3D model to visualize and offers the option of moving, scaling and rotating the model. Another example is MR Studio Immerse [14] which displays 3D models and 3D scans in HoloLens. MR Studio Immerse has same functions as Bridge, and in addition it provides an option of aligning models with the real world and annotating them.

The second type is applications where the user can create a model. This is, for example, 3DBear Holo [15]. This application is used for creating models out of simple shapes, moving, scaling and rotating them. Second example is

3dDraw [16]. User can create a model by drawing lines using their hand and placing blocks.

But not all HoloLens applications are on the Microsoft Store. Applications that are much closer to the goal of this thesis aren't available there. Next part discusses the most similar HoloLens applications that were to be found.

One of these applications worth mentioning is the Real-time High Resolution 3D Data visualization system from M. Garone et al. [17]. This system uses an Intel RealSense RGBD camera mounted to the HoloLens to capture a depth map to reconstruct the current scene. The data obtained from the RealSense camera is sent to a computer, where a reconstruction is calculated and then sent to HoloLens for visualization.

Another to be discussed is the system from Real-Time Mixed-Reality Telepresence via 3D Reconstruction with HoloLens and Commodity Depth Sensors article created by M. Joachimczak et al. [18]. This system focuses on the real-time 3D reconstruction of people to be used for remote communication. The system consists of two computers that have Kinect 2 sensors attached to them. These computers stream depth and RGB data from Kinect 2 cameras to a 3D reconstruction server that is launched on another computer. The server processes the data and creates a 3D reconstruction. The 3D reconstruction is then compressed and sent to a HoloLens device where it is decompressed and visualized.

Another point of interest from the similar applications analysis is Holoportation. Holoportation is a system created by S. Orts-Escolano et al. [19] that real-time reconstructs people, objects and space in augmented and virtual reality. These reconstructions can be transmitted to another remote device, allowing the user to see and hear the 3D reconstructed person and interact with them as if they were present. The data for 3D reconstruction is acquired through 8 camera pods with 2 Near Infra-Red cameras and a color camera mounted on top. This setup collects color and depth data. The depth information is estimated using the data from previously mentioned camera pods and a diffractive optical element with a laser that is projecting a pattern to the scene. The scene is then reconstructed on a computer, compressed and sent to an AR or VR device where it is visualized.

The last existing solution to be considered is from an article called Real-Time Re-textured Geometry Modeling Using Microsoft HoloLens written by S. Dong and T. Höllerer [20]. This application concentrates on a 3D reconstruction of an environment in real-time. Unlike all previously mentioned solutions, this consists only of HoloLens, not using any additional sensors or a server. The application is using the depth data from HoloLens, drawing the data and applying a color texture to the drawn triangles. The visualized 3D reconstruction is dynamically changing as the user is scanning the environment. The texture is size limited, so the larger the scanned environment is, the smaller level of detail will the user get.

From the applications on other platforms, the most widely used are mobile

applications.

One of the most used ones was 123D Catch from Autodesk. The user took several pictures using this application, the application would then upload taken images to the cloud and that would make a 3D reconstruction. After several minutes the reconstruction was available to download, the user could view it and slightly edit the model. However 123D Catch was discontinued [21] and replaced with another product from Autodesk, called ReCap [22]. ReCap works the same way as 123D Catch but in addition, it allows users to lint the application with a laser scanner for even better reconstructions. Both applications let the user view the final 3D reconstruction only on the device, not using augmented reality.

Another application available on mobile devices is Scandy [23]. Scandy is very similar to 123D Catch. In addition to all features that 123D Catch has, it offers a live preview of the 3D reconstruction during the capture. The final 3D model can be uploaded to the internet. The application does not use augmented reality. However, it is advertised that the models can be used as filters in AR and that this feature will be added [24].

Last application to be mentioned is 3D Creator by Sony [25]. 3D Creator mainly focuses on 3D reconstruction of faces allowing the user to create a virtual avatar, that can be viewed in AR, used as an "emoji" in a chat and shared on the internet.

In conclusion, exactly the same system hasn't been yet created. The most similar applications available on HoloLens are visualizing already existing models or mostly using additional depth cameras for the reconstruction. Similar applications on mobile devices focus on reconstruction and don't visualize the 3D reconstruction using augmented reality.

## 3.2 User analysis

Before the designing of the system, a user analysis was performed in order to understand, what would users want and need from such a system. Selected users to be questioned are all interested in AR technologies or 3D modeling, so their answers give an overview of what real users of this system want.

The first question was how long they expect the 3D reconstruction to take. All asked users answered in minutes. The fastest answer was half a minute and the longest was 20 minutes. The most common answer turned out to be 5 minutes and the mean of all responses is 5.68 minutes. This determines that the final time of the 3D reconstruction should be around 5 minutes at most.

Next, they were asked wherever they would want the data created during the 3D reconstruction (captured images and the 3D reconstructed model) to be saved somewhere on the server. 90% of questioned users answered they want the data from the reconstruction to be saved. In conclusion, all the data

created during the 3D reconstruction will be saved in a file on the server so that users could view, edit or use the reconstruction on a computer.

Another question was if it is important for the user to know, where images were taken during the reconstruction and if there should be somehow marked the position where the image was taken. Half of the users said yes and the other half no. It was decided to mark the positions in which images were captured and then based on user testing determine wherever users found it obstructing or helpful.

And a final series of questions was concerning word commands for model manipulation. Users were asked what word in their opinion are best suitable for moving the model, rotating the model and changing the size of the model. For moving the model, 90% of questioned answered with the word "move". For rotating, 90% of users said "rotate". Changing the size of the model got the most varied answers. 50% answered "scale", 25% answered "zoom" and the rest said "resize". From these answers are based chosen keywords for operations with the 3D model and the chosen commands are "move" for moving the model, "rotate" for rotating the model and "scale" for changing the size of the model.

The user analysis provided valuable information and it is projected into the design of the system.

## 3.3 Application on HoloLens

According to Microsoft documentation [26], applications can be created directly using DirectX or developed using the Unity engine.

DirectX allows to directly develop the application using the Windows Mixed Reality APIs. However, that might require for the user to create their own framework. It is said [27] that the development in DirectX is better for the performance of the application, but it is more complicated and time-consuming for the developer.

Unity game engine is recommended for fast and easy development of applications. *"The fastest path to building a mixed reality app is with Unity."* [28] Unity offers a holographic emulator for testing the application without having HoloLens. In addition, Microsoft produced a Mixed Reality Toolkit [29], which is a package of scripts and assets of UI and UX elements.

The application on HoloLens will be developed in Unity.

The UI of the application will be drawn in front of the user so it needs to be minimal to not obstruct the user's view. The designed UI is demonstrated on a picture taken with HoloLens in the figure 3.1. In the middle of the screen is a round cursor so the user would know where they are looking and where is the center of the screen. Another important information for the user is the number of images taken. This number is to be found in the right upper corner and it will be updated every time the user takes a picture. Last information to

be displayed is what is currently happening in the application and instruction for users. This information will be shown in the left upper corner.


Figure 3.1: Wireframe of HoloLens application UI

## 3.4 Server

Calculations performed on the server require operations with matrices. Because of this fact, it was decided to use Python with the Numpy library for matrix operations.

The server will be an HTTP server. An HTTP server can be created just using basic Python libraries but it is rather complicated. There are libraries and frameworks available that make the creation of an HTTP server easier.

The most known frameworks for web applications are Django [30], Flask [31] and Pyramid [32]. Flask framework is said [33] to be ideal for fast and easy development of web applications so it was chosen to be used for the system server.

## 3.5 Communication between the application and the server

The application will communicate with the server as it is designed in the figure 3.2. The application will send captured images to the server, which will save them. When the user stops the image capture, the application will send all camera positions from image capture. The server will then run a 3D reconstruction and calculate the transformation of the model using the camera

positions sent by the application. The transformation needs to be calculated in order to place the reconstruction over the reconstructed object. The server will send the calculated transformation to the application. Application will then request the 3D reconstructed model. The server will decimate the model if needed and then sent it to HoloLens to be visualized.

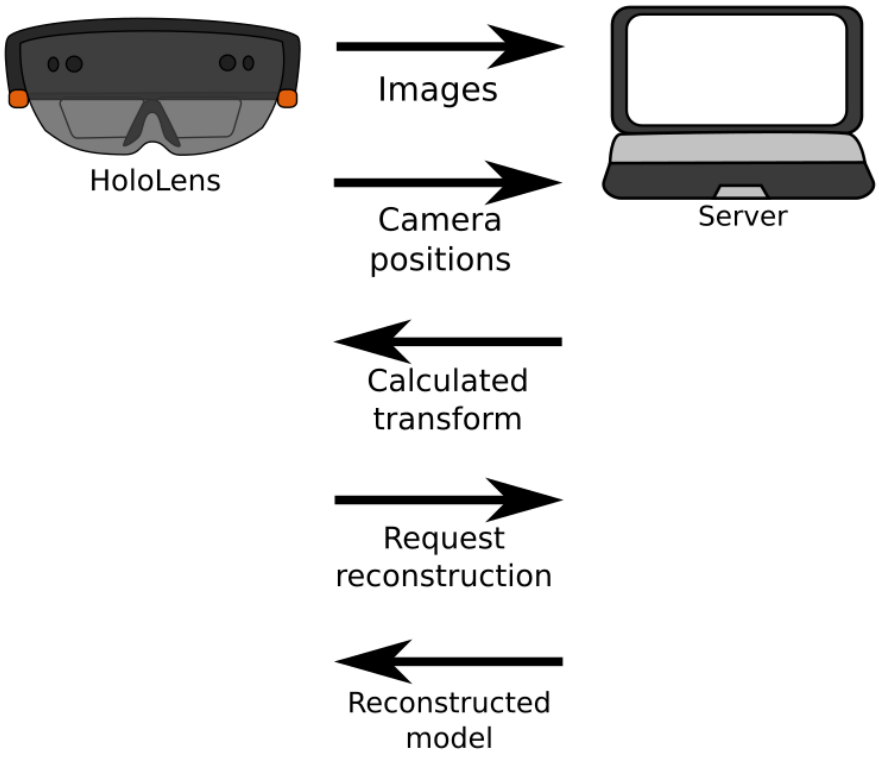


Figure 3.2: Graph of communication between HoloLens application and server

## 3.6 Model reconstruction

The 3D reconstruction will be done using data acquired from HoloLens. All calculations will be done on the server.

There are two options for the 3D reconstruction process available. The first option is using the depth data from HoloLens to reconstruct the geometry and using the HoloLens camera for color information. However as it has been shown in the article about the Real-time High Resolution 3D Data visualization from M. Garone et al. [17], this option is not very effective, since the depth data obtained from HoloLens is not suitable for an object reconstruction. A comparison of HoloLens depth data and a real scene can be seen in figures 3.3 and 3.4.

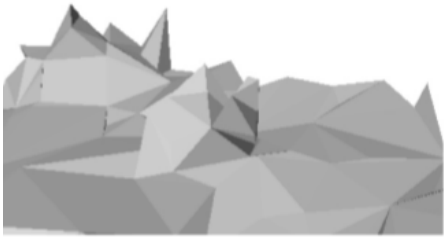

Figure 3.3: HoloLens depth data of a scene from the Real-time High Resolution 3D Data article [17]

Figure 3.4: The scene from the Real-time High Resolution 3D Data article [17]

The second option for 3D reconstruction is reconstruction from images captured by HoloLens. These images will be processed by a pipeline for image reconstruction. This pipeline will be called by the server using the command line to run the reconstruction. Possible pipelines are analyzed in the following part.

First software to be considered is ReCap by Autodesk [34]. ReCap creates 3D reconstructions from images or laser scans. The reconstruction is in the form of a mesh or a point-cloud. The software allows editing of the mesh and exporting it into any 3D format. ReCap includes a command-line tool for reconstruction called DeCap [35]. This command-line tool, however, doesn't provide the user with many options to direct the reconstruction, it creates a ReCap project and it doesn't support export in another format. One of the options DeCap is offering is a model decimation which would be useful, but the low choice of directing the reconstruction doesn't make it suitable for this system.

Issues that made ReCap not suitable are well resolved in Capturing reality [36]. Capturing reality is a software that generates 3D reconstructions from images and laser scans. It offers a command-line interface that allows the user to do almost everything they can do in the application. This would be an ideal solution for this system, but it won't be used because it is not open source so the user would have to buy the license if they would want to use this system.

Another examined software is Meshroom by AliceVision [37]. Meshroom is an open source 3D reconstruction software that processes reconstructions from images. Meshroom offers the reconstructed model to be a point-cloud or a mesh. A reconstruction can be run from a command-line. However the official documentation [38] doesn't fully cover that. Some users have stated [39] [40] [41] that the reconstruction can take from 5 minutes to over an

hour. Since the user analysis showed that users prefer faster reconstruction, Meshroom will not be used in this system.

Last pipeline and the one that will be used is COLMAP created by Johannes L. Schoenberger [1] [2] [3]. COLMAP is a "general-purpose Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipeline " [1] that recreates 3D models from images. The command line interface is available and allows the user a lot of control over the reconstruction. The result of a 3D reconstruction is a mesh and a point-cloud. The mesh is created from the point-cloud using either a Poisson or a Delaunay method for meshing the point-cloud.

## 3.7 Model decimation

The reconstructed model can be too large for the HoloLens application to run smoothly. In this section will be discussed possible solutions for this problem. There are two main possible solutions: implementing a model decimation on the server or using an application.

The implementation of the model decimation is an extensive subject. Model decimation has been implemented several times before. To create a model decimating script, that would be fast and would result in a good quality model, would take a lot of time so it was decided to utilize a software that would mediate the decimation.

The software suitable for the decimation would be the one that offers a command line interface and either can be very precisely directed (letting the user define how much to decimate, the format of the model, ...) or that can be scripted.

These requirements match Blender [42]. Blender is an open source software for 3D modeling. Blender supports being run from a command line and can be scripted using Python.

Chapter 4

# Realization

This chapter is about the realization of the system.

There are two main parts: application on HoloLens and server. Each part will describe each important step of the whole process of 3D object reconstruction.

## 4.1 Used technologies

The HoloLens part of the system is developed using Unity [43] and .NET backend. The server part is realized in Python using Flask framework [31].

## 4.2 Application on HoloLens

The HoloLens application is developed in Unity version 2018.1.3f1 using .NET scripting backend. Some settings need to be done at the beginning of the project for it to run on HoloLens without any problems. First, the camera had to be set to the origin of the scene (coordinates [0, 0, 0]). The camera background was changed from Skybox to Solid color and set the color to be RGBA (0, 0, 0, 0). Then for the application to run well without any lags, the quality setting was set to very low. The smooth functioning of the application is very important as large and often lags could cause the user to feel uneasy. And finally, the target SDK (software development kit) had to be specified to Windows 10 with Windows Mixed reality enabled.

### 4.2.1 Mixed reality toolkit

The application uses some of the scripts and prefabs from Mixed reality toolkit for the UI. Mixed reality toolkit is a Microsoft open source project that contains some of the building blocks for a HoloLens application. It is used because it *"showcases UX best practices with UI controls that match Windows Mixed Reality and HoloLens Shell."* [29].

### 4.2.2 User interface

In the middle of the screen is the cursor that follows the user's gaze and reacts when the user is using gestures such as the air-tap. Prefabs and scripts for this part of the UI are from the Mixed reality toolkit [29].

In the upper right corner of the screen is located the image counter. It displays the number of images that have been already captured since the start of the application. It is a simple text mesh, that is updated every frame and drawn on a canvas in front of the camera. The image capture is started with an air-tap gesture and it is stopped again with and air-tap, that also triggers the model reconstruction.

In the upper left corner is the information panel, which informs the user about everything that is happening. As well as the image counter, it is a simple text mesh on the same canvas. It is displayed using the DebugWindow.cs script that can be seen in the code demonstration 1. The text is sent to the Unity console using the function `Debug.Log()` this triggers the `Application.logMessageReceived` event and the text are then added to the text mesh and drawn.

```
void OnEnable()
{
    Application.logMessageReceived += LogMessage;
}

.
.
.

public void LogMessage(string message,
                       string stackTrace, LogType type)
{
        textMesh.text = message + "\n";

}
```

Code demonstration 1: The information text code

### 4.2.3 Image acquisition

The image acquisition is located in the CameraCapture.cs script it is processed by the function `TakePhoto`.

The automatic image acquisition is triggered with an air-tap gesture. With every image being captured, the application plays a sound of the camera clicking to let the user know an image has been taken. Once the acquisition is

started a photograph is taken every time the user moves his head 30 centimeters. The head location is checked and updated every frame in the `Update()` function in the CameraCapture.cs script as it is shown in code demonstration 2. Both the `newHeadPosition` and the `startHeadPosition` has been initialized during the start of the script. The variable `newHeadPosition` contains current head position and `startHeadPosition` is the position when the last image was taken.

```
void Update()
{
    newHeadPosition = Camera.main.transform.position;
    double headPositionDistance = Math.Sqrt(
            (newHeadPosition.x - startHeadPosition.x) *
            (newHeadPosition.x - startHeadPosition.x) +
            (newHeadPosition.y - startHeadPosition.y) *
            (newHeadPosition.y - startHeadPosition.y) +
            (newHeadPosition.z - startHeadPosition.z) *
            (newHeadPosition.z - startHeadPosition.z));
    textMesh.text = photoCount.ToString();

    if (headPositionDistance >= distance)
    {
        startHeadPosition = newHeadPosition;
        if (cameraCaptureOn)
        {
            TakePhoto();
        }
    }

}
```

Code demonstration 2: The headset location update

The image capture is accomplished using Unity class PhotoCapture [44]. The image data is stored as a list of bytes that can be obtained by method `photoCaptureFrame.CopyRawImageDataIntoBuffer` as it is in code demonstration 3. This data is sent to the server to be saved as an image. The figure 4.1 demonstrates an example of images taken by HoloLens during the run of the system.

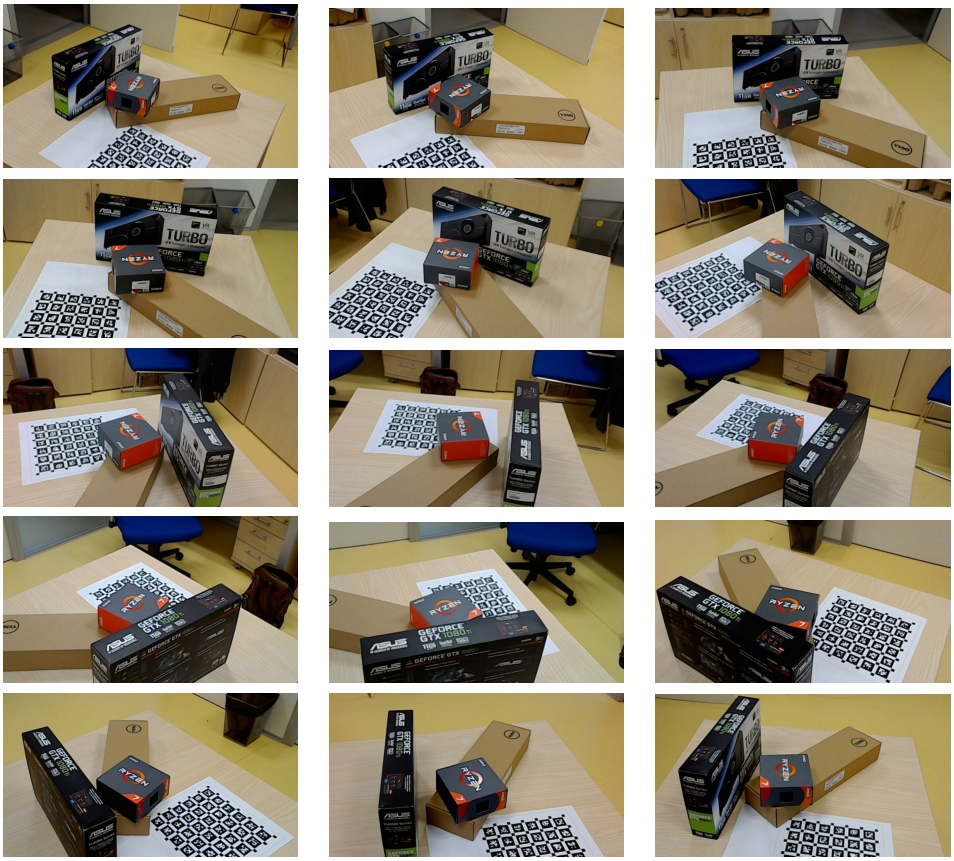

Figure 4.1: Example of acquired images

```
List<byte> imageBufferList = new List<byte>();
photoCaptureFrame.CopyRawImageDataIntoBuffer(imageBufferList);

UploadImage(imageBufferList.ToArray());
```

Code demonstration 3: Image data

To let the user know, where they took the images, there are cameras being drawn in the exact position as when the image was taken. This is achieved by getting the current head position from the moment of the image acquisition. The cameras are then drawn as red pyramids in the DrawCameras.cs script. The marked camera positions are visible in the figure 4.2.


Figure 4.2: Cameras marking positions of image captures

### 4.2.4 Communication with server

Communication with the server is accomplished using a Unity Networking library. This library contains the UnityWebRequest class [45] that is processing the communication. This class builds an HTTP request of any type that is needed and sends it to a specified URL. All the communication with the server is done in the CameraCapture.cs script. In the code demonstration 4 is an example of a request for downloading the reconstruction. A get request is sent to the server at a route /api/cv/query_reconstruction. The `HandeReconstructionDownload` function is a handler, waiting for a response from the server with the reconstruction data.

```
 void DownloadReconstruction()
{
    string url = BASEURL + "/api/cv/query_reconstruction/";
    UnityWebRequest request = UnityWebRequest.Get(url);
    UnityWebRequestAsyncOperation op = request.SendWebRequest();

    op.completed += HandleReconstructionDownload;
}

//handler for http request, draws reconstruction
void HandleReconstructionDownload(AsyncOperation op)
{
    UnityWebRequestAsyncOperation aop =
                        (UnityWebRequestAsyncOperation)op;
    byte[] result = aop.webRequest.downloadHandler.data;
    .
    .
    .
}
```

Code demonstration 4: Communication with the server

The whole process of communication of the application with the server during one reconstruction is described by the figure 4.3. The application is sending each image to the server, the server saves the image and replies with the name of the image. The name is used as a camera position ID for pairing the camera positions with camera positions reconstructed in COLMAP for later calculations of transformation. After the image acquisition is complete, the application sends all camera positions from when images were taken. The server calculates a transformation between these positions and the positions of cameras that were reconstructed and sends a reply containing the transformation. Then the application sends a request for the 3D reconstructed model and server replies with the data of the decimated and reconstructed model.

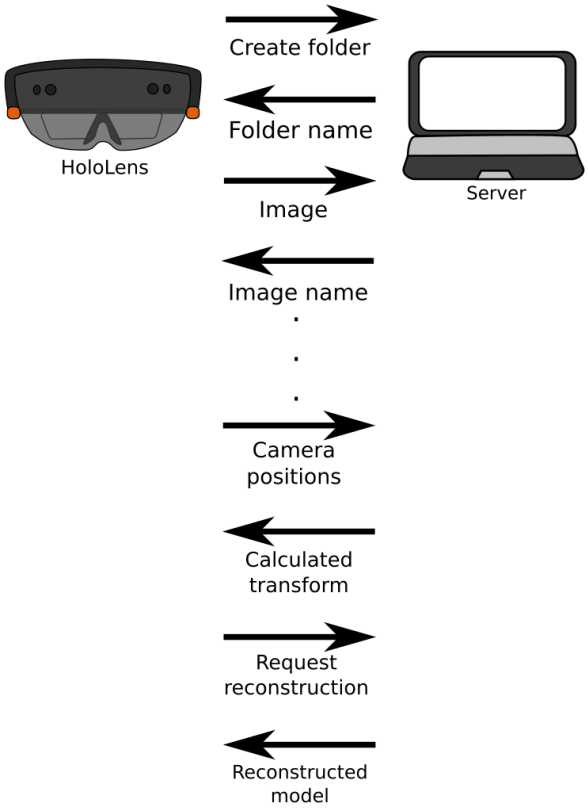


Figure 4.3: Communication between the application and the server

### 4.2.5 Mesh visualization

After the reconstruction, the server sends a JSON containing the model data. The data is composed of vertices, colors for each vertex and faces in the form of vertex indents. That data is then assigned to the mesh filter of the visualization object in the function `HandleReconstructionDownload` from the CameraCapture.cs script. The mesh filter passes the mesh to mesh renderer, where the mesh is drawn using the set material. This material's color is assigned in a simple shader called VertexShader.shader (code demonstration 5). The shader assigns each vertex a color that's specified in the mesh variable color. The visualization of the 3D reconstruction can be seen in figures 4.4 and 4.5.


Figure 4.4: Examples of several visualizations of the 3D reconstruction


Figure 4.5: Examples of several visualizations of the 3D reconstruction

```
    Shader "Custom/VertexShader" {
SubShader{
Pass{
CGPROGRAM
#pragma vertex vert
#pragma fragment frag
#include "UnityCG.cginc"

// vertex input: position, color
struct appdata {
float4 vertex : POSITION;
fixed4 color : COLOR;
};

struct v2f {
float4 pos : SV_POSITION;
fixed4 color : COLOR;
};

v2f vert(appdata v) {
v2f o;
o.pos = UnityObjectToClipPos(v.vertex);
o.color = v.color;
return o;
}

fixed4 frag(v2f i) : SV_Target{ return i.color; }
ENDCG
}
}
}
```

Code demonstration 5: Mesh shader

### 4.2.6 Model manipulation

Manipulation with the model is achieved using the Mixed Reality Toolkit Input scripts for voice and gesture recognition. The script requires keywords and methods that should be run when a certain keyword is recognized.

The reconstructed model doesn't appear as a solid object in the application which means that the user can't interact with it. This is solved by inserting an invisible cube in the center of the reconstruction, that is the same size as the reconstructed model. This cube has a Box Collider set, which means that it collides with the user's gaze and results in detecting when the user is interacting with it. The user interaction then triggers the manipulation with the reconstructed model.

The current model manipulation supported is rotating the model around the Y axis, moving the model and changing the size of the model. User can switch between these types of manipulation by voice command "rotate", "move" or "scale" and perform the manipulation by air-tapping and dragging their hand. The code demonstration 6 shows an example of how the rotation is accomplished. The model manipulation is implemented in the GestureAction.cs script. The `RotationSensitivity` is set during the initialization of the script and it represents the sensitivity of hand movement to the rotation. The parameter  stands for the movement of the hand. The rotation is then calculated by multiplying the `RotationSensitivity` with the position of the hand on X-axis.

```
void INavigationHandler.OnNavigationUpdated(NavigationEventData e)
{
    if (isNavigationEnabled)
    {
        float rotationFactor = e.NormalizedOffset.x
                               * RotationSensitivity;
        transform.Rotate(new Vector3(0, -1 * rotationFactor, 0));
    }
}
```

Code demonstration 6: Rotation of the model

## 4.3 Server

The server is created using Flask framework [31]. The server obtains images and runs a reconstruction. Then it calculates a transformation between the reconstructed model and the real world object from camera positions. Finally, it decimates the model and sends it to the HoloLens application.

### 4.3.1 Image acquisition

Image acquisition on the server is implemented in the `api_save_images` function as can be seen below in code demonstration 7. As the decorator implies, the route to this function is /api/cv/save_images and it expects POST HTTP requests. Once called, this function tries to save obtained data as a jpg image. The server sends the saved image name as a response to the application.

```
    @app.route("/api/cv/save_images/", methods=['POST'])
def api_save_images():
    try:
        for f in request.files:
            img = request.files[f]
            global folder
            start = time.time()
            pimg = Image.open(img.stream)
            path = os.path.dirname(os.path.realpath(__file__)) + "/"
            name = str(start).replace('.', '_') + "_" + str(f) + ".jpg"
            filename = path + folder + "/" + name

            pimg.save(filename);

            response_list = [("file", name)]

            response_dict = dict(response_list)
            return flask.jsonify(response_dict)

    except Exception as err:
        print("ko:", err)

return "ok"
```

Code demonstration 7: Server image acquisition

### 4.3.2 3D reconstruction

Reconstruction is processed by COLMAP [1] that is called by the server via command line in the `api_get_transformation` method. COLMAP is a Structure-from-Motion and Multi-View Stereo pipeline widely used for 3D reconstruction from images. The way to run COLMAP from a command line can be seen in code demonstration 8. The command needs a workspace path that determines, where the reconstruction should be saved, an image path to know which images to use for a reconstruction and an optional parameter is the quality of reconstruction. High quality reconstruction can take a few minutes, so for the whole system to be as fast as possible, the recommended quality for reconstruction is medium.

```
os.system(path +
"COLMAP.bat automatic_reconstructor \ --workspace_path "
+ path + folder + " \ --image_path " + path + folder
+ " --quality " + quality)
```

Code demonstration 8: Calling COLMAP

After the reconstruction is done, the 3D model is very large, containing around hundreds of thousands of vertices. The first concern is that Unity has a limit for the number of vertices in a mesh. This limit is said to be around 65 000 vertices [46]. The second problem is that the higher the number of vertices is, the more time it takes to draw the model. That could result in the application lagging which is not desired. So the model needs to be decimated before sending it to the application.

This is implemented in the `api_download_model` method. The decimation is done using Blender [42]. Blender can be run from a command line and fully supports scripts. The server runs Blender with a script that opens the reconstructed model and decimates it until the number of faces is smaller than 20 000. Then it exports the decimated model to a file. The command for Blender can be seen in the code demonstration 9 and the script in the code demonstration 10. The script needs a path to the decimated model as an argument. The decimated model is saved in the same folder as the original model. The original result of the reconstruction is shown in the figure 4.6 and to compare, the decimated model is in the figure 4.7.

```
os.system('.\Blender\\blender.exe --background --python "'
+ path + 'decimation_script.py" -- "'
+ modelpath +'"')
```

Code demonstration 9: Calling Blender

```
    argv = sys.argv
    argv = argv[argv.index("--") + 1:]
    bpy.ops.import_mesh.ply(filepath=argv[0] + "/meshed-poisson.ply")

    for obj in bpy.data.objects:
        if obj.type == "MESH":
            modDec = obj.modifiers.new("Decimate", type = "DECIMATE")
            bpy.context.scene.update()
            #len(obj.data.vertices)

            while modDec.face_count >= 20000:
                modDec.ratio *= 0.5
                bpy.context.scene.update()


bpy.ops.export_mesh.ply(filepath= argv[0] + "/decimated.ply",
                        use_normals=False, use_uv_coords=False)
```

Code demonstration 10: Blender script

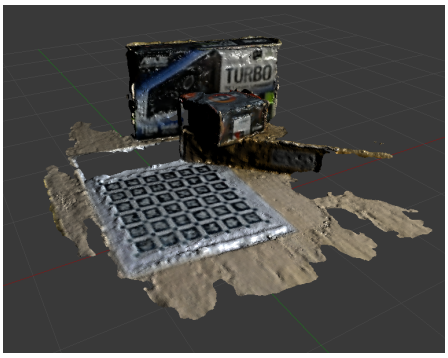

Figure 4.6: 3D reconstructed model containing around 230 thousand vertices

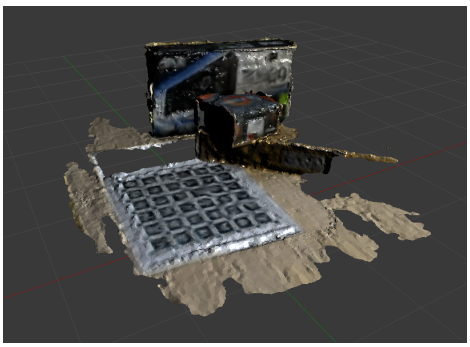

Figure 4.7: Decimated model containing around 25 thousand vertices

The final model is then transformed to be mapped to the reconstructed object and sent by the server as arrays of vertices, faces and colors to the application to be visualized.

### 4.3.3 Model transformation

The final 3D reconstruction needs to be placed on the object that was reconstructed. The translation, rotation, and scale are calculated using HoloLens

camera positions and reconstructed camera positions from COLMAP.

The server obtains HoloLens camera positions from HoloLens application. Then it loads the reconstructed camera positions from a file created by COLMAP. The camera positions from COLMAP are in the form of rotation and a translation, so the center of the camera is calculated by multiplying the translation by the rotation. The loading of the camera positions is done in the method `api_get_transformation` on the server

The calculation of the transform is then done in the `calculate_transform` method. First, to calculate the transformation COLMAP camera positions need to be converted from the coordinate system of COLMAP to the coordinate system of Unity. COLMAP uses a right-handed coordinate system and Unity left-handed coordinate system. The transformation is done by making the Y-axis negative. This is achieved by multiplying COLMAP camera centers with a reflection matrix, as it can be seen in the code demonstration 11).

```
 Q = np.array([[1, 0, 0], [0, -1, 0], [0, 0, 1]])
.
.
.

for i in range(numOfCams):
    colCams[i] = np.matmul(Q, np.array(colCams[i]))
```

Code demonstration 11: Converting camera centers from right-handed to left-handed coordinate system

When both sets of camera centers are in the same coordinate system, the transformation is calculated using the Procrustes analysis [47]. The Procrustes analysis calculates the best transformation (rotation, scale, and translation) to match two sets of points onto each other. An example of the matching can be seen in the figure 4.8. The green dots represent centers of COLMAP cameras that are matched to the centers of HoloLens cameras that are blue. Resulting in transformed centers of COLMAP cameras that are represented by red dots. The distance difference between pairs of cameras before and after the transformation is compared in the figure 4.9

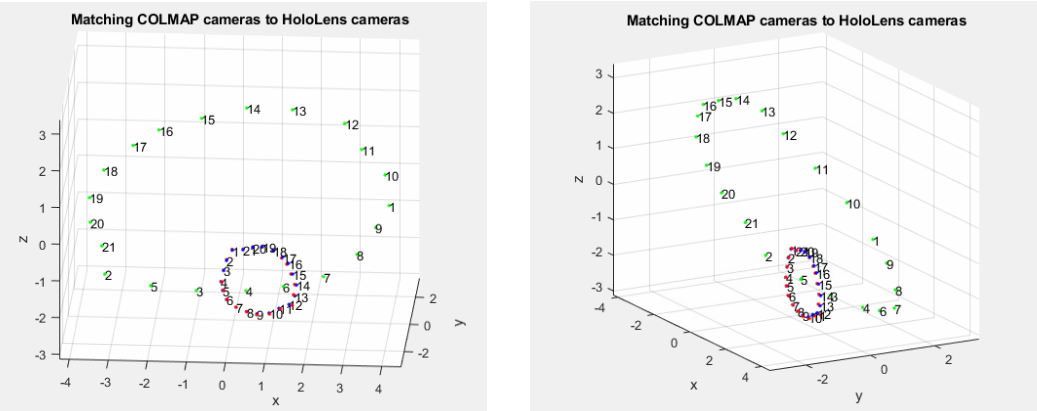


Figure 4.8: Camera positions reconstructed by COLMAP (green dots) that are matched to HoloLens camera positions (blue dots) using Procrustes analysis resulting in the reconstructed camera positions being mapped to HoloLens camera positions (red dots)

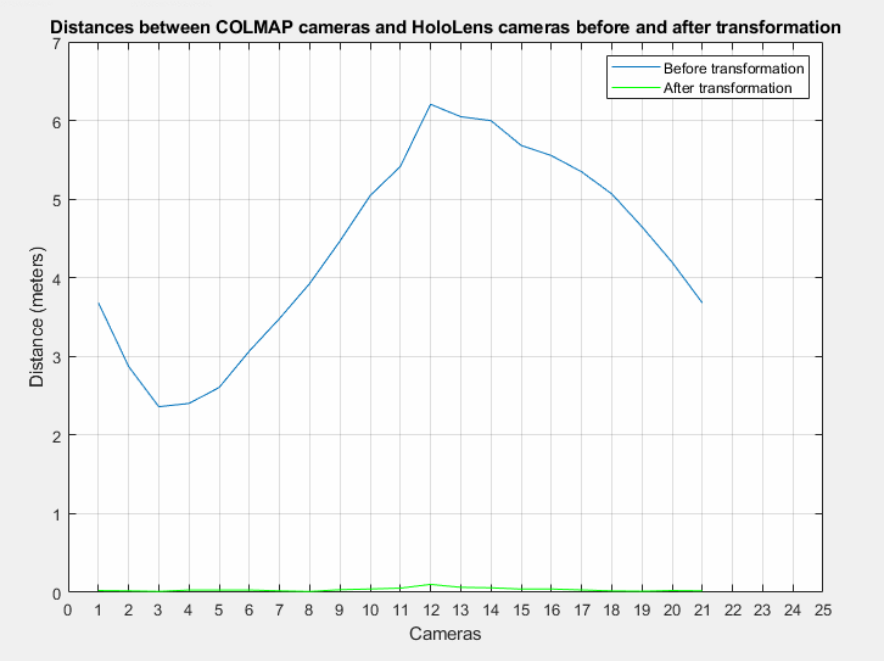


Figure 4.9: Comparing the distance between COLMAP cameras and HoloLens cameras before and after the transformation calculated with Procrustes analysis

The calculated scale, rotation, and translation are stored in global variables to be then applied to the reconstructed model when the application sends a request to download the 3D model. The `api_download_model` method processes the 3D model decimation, the transformation and sends it to the application. The 3D model is stored in a polygon file format (.ply). The file is read line by line, each line being parsed into vertices, colors and faces. The model transformed as it can be seen in the code demonstration 12. Each vertex is first converted into a left-handed coordinate system by negating the Y coordinate. The vertex is moved to the origin of the COLMAP cameras coordinate system. Then the vertex is transformed by the transformation calculated using Procrustes analysis. It is multiplied by rotation and scale and moved by the translation.

```
vertex = np.array([verts[index],
                   -1.0 * (verts[index+1]),
                   verts[index+2]])
vertex = vertex - globmuY
vertex = vertex / globnormY
vertex = scale * np.matmul(vertex, rotation) + translation
```

Code demonstration 12: Transforming the 3D reconstruction to Unity coordinate system

The transformed model is then sent to the HoloLens application as a list of vertex coordinates, colors and face indices.

## 4.4 Testing

The system was tested on users to determine if the UI is intuitive and if the system is easy to use for new users.

Videos of all the testing can be found on the enclosed CD. There are two videos for every tested user. The first video is of the user using the application. The other video is a view from the application. However, HoloLens does not support two applications accessing the camera, so the in-application videos were taken after the reconstruction was done, showing results of the user's reconstruction.

The system is designed for users of HoloLens. However, some of the tested users have never used HoloLens before. To compensate this fact, a tutorial about operating the device was played to them before the testing. First, before

the testing of the system, tests subjects were asked a series of questions:

- Have you ever used Augmented reality? If yes, on what occasion?
- Have you ever used HoloLens?
- What do you understand under the term "3D Copy"?
- Do you prefer a fast and smoothly running program or a high quality of a model and a lot of details?
- Do you have any experience with 3D modeling?

The initial questioning revealed that only half of users had some experience with Augmented reality and their experiences were mostly with AR on mobile phones. A third of the questioned had used HoloLens. Two types of answers appeared concerning the term "3D Copy". Half of the users said they pictured a 3D printed copy of an object. The other half understood it as a virtual copy of an existing real world object. When asked about their preferences (speed and latency or high quality), about three quarters preferred fast and smoothly running program over a high quality of a model. Most of the questioned users have no experiences with 3D modeling.

After starting the application, users were asked to perform a few tasks to ensure, that the application can be used by users without any struggles.

The first task was to start an image capture. Almost all tested users didn't have any problems figuring out how to start taking images. The only ones with some issues where users with astigmatism that made it impossible for them to read the instructive text.

When the image acquisition was started, users were asked to take as many images of objects on the desk as they thought was suitable for a reconstruction and then to stop the image acquisition. Both, the mean and the median of number of images taken was 21 images. All users took enough images for the reconstruction to run successfully. However, some users struggled with reconstruction because most of the images they took were motion blurred. This was caused because images were taken while the user moved. Most of the users haven't realized that they needed to be steady while taking an image. But after realizing that, all users managed to take images sufficient for a reconstruction. Another issue that was discovered was, that if the user moved too fast some of the images weren't saved and caused an exception in the application. This problem will be solved by using a queue for captured images instead of a single photo capture variable.

When taking images was accomplished, users were asked if they can tell from looking at the UI how many images had been taken. All users answered correctly without any hesitating.

Finally, after the reconstruction was finished, tested users were asked, how would they move, rotate or scale the reconstructed model. Most of them

didn't realize, they could use voice commands to manipulate the device, but after they were reminded, that HoloLens can be operated using voice, they used correct commands.

An example of 3D reconstructions made by tested users can be seen in figure 4.10


Figure 4.10: 3D reconstructions created by tested users


Figure 4.11: Tested users with objects they were reconstructing

After the testing, tested users were asked another set of questions to find out their experience from using the application:

- Did you have any struggles while using the application?
- Did your answer about preferring speed or quality change after using this application?
- Were you satisfied with the final 3D model?
- Did you find the UI intuitive? Did it have all the information you needed?
- Did you find the marked positions of image captures helpful?

The final questioning showed that most of the tested users didn't feel like they struggled during fulfilling given tasks. The minority answered that they struggled with the air-tap. Users, who responded in the initial questioning that they preferred speed and latency over high quality model haven't changed their opinion. However, users preferring the quality of the model said, they give preference to speed and latency. Half of the questioned users liked the reconstructed model. The other half said that the model wasn't detailed enough for their preferences. All users answered that the UI was informative and adequate for everything they were doing. No user had any problem with marked locations of image captures, they all found it helpful.

In conclusion, the change that came out of this testing was heightening the quality of the reconstructed model. The captured photos will be put into a queue to be processed in a case the user moves too fast. As was shown in the testing, marking the positions in which the user took the image was found helpful and it will be kept.

Chapter 5

# Experimental validation

This chapter describes the experimental findings that were done during the development of the system.

### 5.0.1 HoloLens location data

Every image capture contains data about the location where it was taken. This location consists of a rotation and a position. The position could mean either the center of the head or the center of the camera. This information was unavailable in the official Microsoft and Unity documentation.

If the acquired position is the center of the camera, it would mean that there are no alterations to the calculation of transformation between the coordinate system needed. However, if the acquired position would be the center of the head, the HoloLens camera center positions would have to be calculated from the center of the head.

In order to find which of the two possible positions does HoloLens return, there was an experiment performed.

The HoloLens device was placed on an arm attached to a rotation table. Above the rotation table was a bridge-looking construction. From the top of this construction was hanged an object to be photographed and reconstructed. The whole construction can be seen in the image 5.1.

The rotation table then slowly rotated, with HoloLens on it pointing towards the center of the table and taking a picture every 15 centimeters. After the photo capture, all the camera location data was saved to a text file and further inspected using MATLAB. There were four measurements in total, each time at a different measured distance from the center of the rotation table.

Figure 5.1: The set up of the experiment

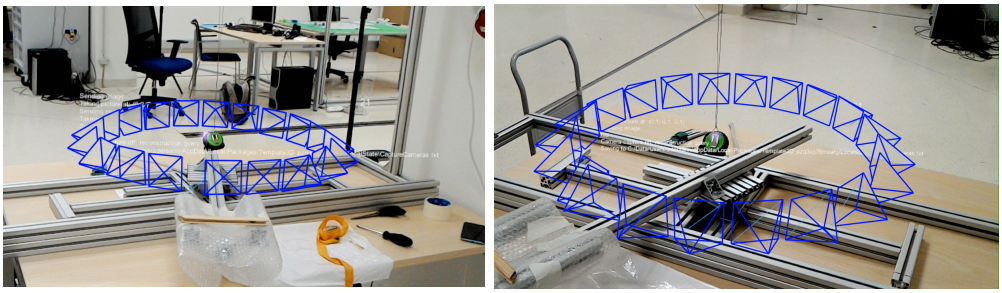

Figure 5.2: The set up of the experiment with marked locations of image capture

The inspection of the camera positions consisted of calculating the center of all the camera positions for each measurement and then calculating the deviation of HoloLens camera positions from the distance between the center of the rotation table and the HoloLens camera. The histogram of all deviations in the figure 5.3.

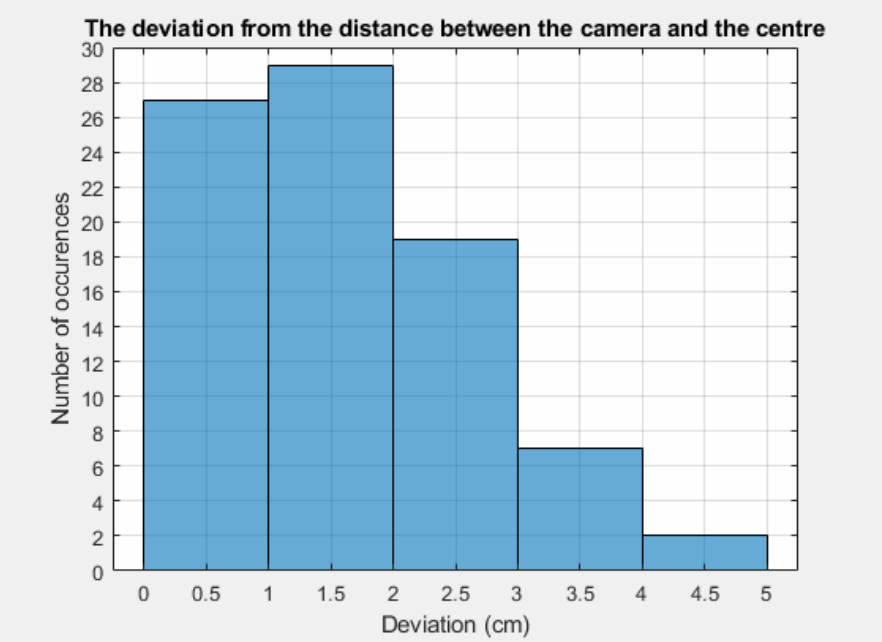


Figure 5.3: Histogram of deviations of HoloLens camera positions from the distance between the center of the rotation table and the HoloLens camera

The deviation mostly from 0 to 3 centimeters, which is a slight inaccuracy caused probably by HoloLens location tracking. These deviations are not big enough to imply that the position returned by HoloLens represents the center of the head. This leads to the deduction that the position acquired during image capture corresponds to the center of the HoloLens camera.

Chapter 6

# Installation

This chapter is about how to install and run the system. The first part is describing running the server and the second part explains the installation of the application on HoloLens.

## 6.1 Server

The server was developed in Python version 3.7.1., so this is the recommended version for running the program. To run successfully the server requires to have Numpy, PIL and flask libraries installed. The server is then started by executing the program using Python in the command line.

## 6.2 Application on HoloLens

There is a version of the build of the application available on the CD enclosed. However, if there is a need to build the application, there are some important settings that can't be forgotten. All those settings can be seen in the image 6.1. First, the platform needs to be switched to "Universal Windows Platform". User needs to select "Universal Windows Platform" and click on "Switch Platform". When the platform is set correctly, it is necessary to set the build type to D3D and to check the box for "Unity C# Projects". With these settings, the build of the application will run on HoloLens.

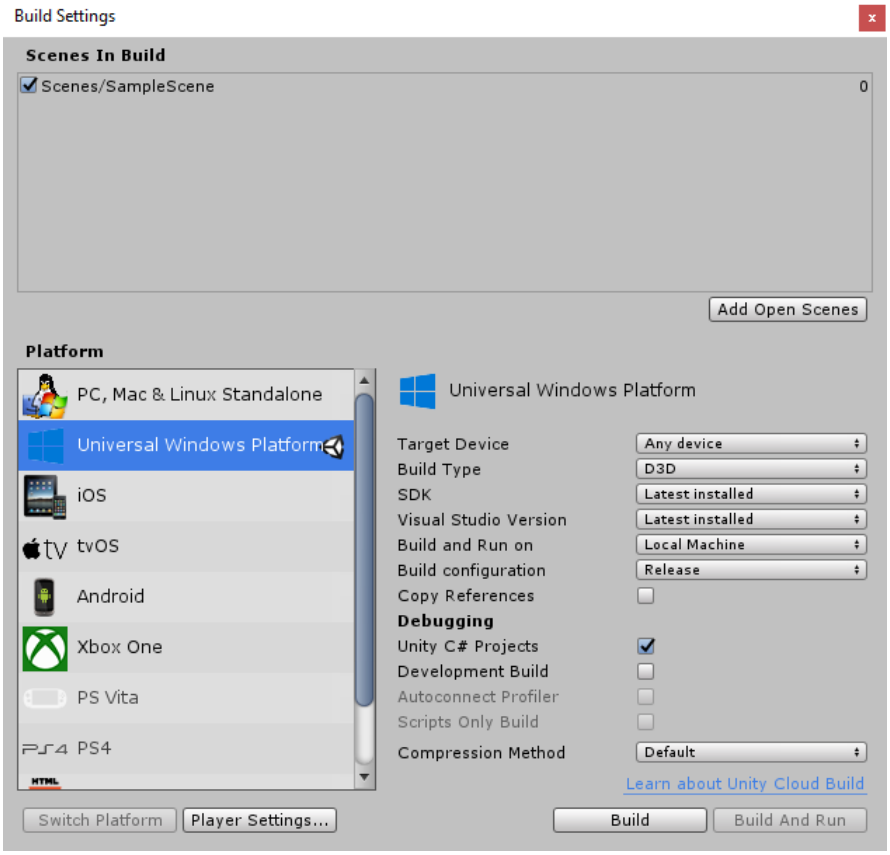


Figure 6.1: Settings of a build of the application

The build of the application generated a Visual Studio project. This project is used to deploy the application to the HoloLens device. When opened in Visual Studio, it needs to be specified where and how the application will run. These settings are demonstrated in the image 6.2. The option of Solution Configurations should be set to Release. The Solution platform is x86. Finally, the last option is on which device the deployment should be. This setting depends on how the HoloLens is connected to the computer. If it is done using a cable, the user needs to select “Device”. In this example, the HoloLens is connected via WiFi, so the “Remote Machine” option is selected. When all these settings are done, the application is deployed by clicking the green triangle button.

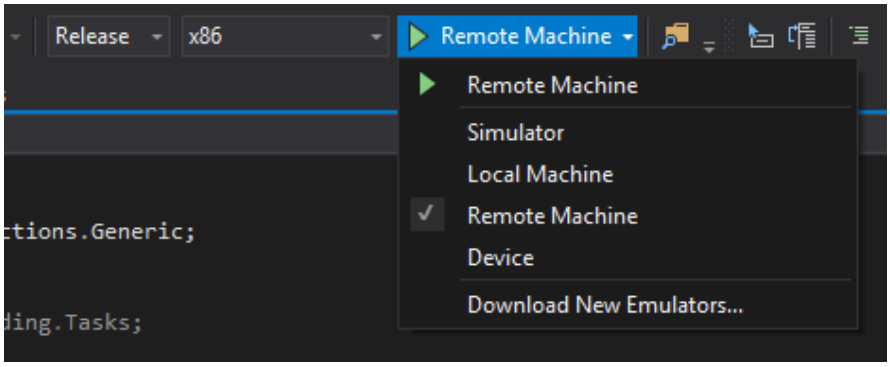


Figure 6.2: Deploying the application to HoloLens

# Conclusion

The goal of this thesis was to propose and implement a system for a fully automatic 3D model reconstruction and model visualization on Microsoft HoloLens. This system consists of a HoloLens application and a sever. The HoloLens application runs the image acquisition and sends it to the server. The server then performs the 3D reconstruction and sends the reconstructed model to the HoloLens application. The application visualizes the reconstruction allowing the user to view the model and manipulate with it.

The result of this thesis is a system for 3D reconstruction of the object, that doesn't require any additional hardware. Users can create a 3D reconstruction, view it and manipulate with it. The system is a great contribution for users that need to recreate some objects from the real world in a virtual world.

There is still some room for future improvement of the system. The possibilities of using an image texture will be examined and implemented.

In conclusion, previously determined goals of this thesis were accomplished.

# Bibliography


1. SCHOENBERGER, Johannes L. *Colmap* [online] [visited on 2019-04-18]. Available from: `https://colmap.github.io/`.
2. SCHÖNBERGER, Johannes Lutz; FRAHM, Jan-Michael. Structure-from-Motion Revisited. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
3. SCHÖNBERGER, Johannes Lutz; ZHENG, Enliang; POLLEFEYS, Marc; FRAHM, Jan-Michael. Pixelwise View Selection for Unstructured Multi-View Stereo. In: *European Conference on Computer Vision (ECCV)*. 2016.
4. REALITYTECHNOLOGIES.COM DIVERSIFIED INTERNET HOLDINGS LLC. *The Ultimate Guide to Understanding Augmented Reality (AR) Technology* [online] [visited on 2019-05-09]. Available from: `https://www.realitytechnologies.com/augmented-reality/`.
5. GOOGLE LLC. *Glass* [online] [visited on 2019-05-09]. Available from: `https://x.company/glass/`.
6. MICROSOFT CORPORATION. *Microsoft HoloLens* [online] [visited on 2019-05-09]. Available from: `https://www.microsoft.com/en-us/hololens`.
7. BBC. *First Hololens kit to cost $3,000* [online] [visited on 2019-05-09]. Available from: `https://www.bbc.com/news/technology-35686616`.
8. MICROSOFT CORPORATION. *HoloLens hardware details* [online] [visited on 2019-05-09]. Available from: `https://docs.microsoft.com/en-us/windows/mixed-reality/hololens-hardware-details`.
9. GRABHAM, Dan. *Microsoft HoloLens 2 features, news and release date: Everything you need to know* [online] [visited on 2019-05-09]. Available from: `https://www.pocket-lint.com/ar-vr/news/microsoft/146803-microsoft-hololens-2-specs-release-date-news-features`.

10. MICROSOFT CORPORATION. *HoloLens2-Overview, Features and Specs* [online] [visited on 2019-05-09]. Available from: https://www.microsoft.com/en-us/hololens/hardware.

11. WALFORD, Alan. *Photogrammetry* [online] [visited on 2019-05-09]. Available from: http://www.photogrammetry.com/.

12. MICROSOFT CORPORATION. *Microsoft App Store* [online] [visited on 2019-04-16]. Available from: https://www.microsoft.com/en-us/store/collections/hlgettingstarted/hololens.

13. IMMERSIV.IO. *Bridge* [online] [visited on 2019-04-16]. Available from: https://www.microsoft.com/en-us/p/bridge-immersivio/9pb6lvrltxpc?cid=msft_web_collection&activetab=pivot:overviewtab#.

14. ARVIZIO INC. *MR Studio Immerse* [online] [visited on 2019-04-16]. Available from: https://www.microsoft.com/en-us/p/mr-studio-immerse-20/9ngr4c2zk1lp?cid=msft_web_collection&activetab=pivot%5C%3Aoverviewtab.

15. 3DBEAR OY. *3DBear Holo* [online] [visited on 2019-04-16]. Available from: https://www.microsoft.com/en-us/p/3dbear-holo/9p2109dw9nnx?cid=msft_web_collection&activetab=pivot%5C%3Aoverviewtab#.

16. CASE WESTERN RESERVE UNIVERSITY. *3dDraw* [online] [visited on 2019-04-16]. Available from: https://www.microsoft.com/en-us/p/3ddraw-for-hololens/9nblggh51dwh?activetab=pivot:overviewtab#.

17. GARON, Mathieu; BOULET, Pierre-Olivier; DOIRONZ, Jean-Philippe; BEAULIEU, Luc; LALONDE, Jean-Franfffdfffdois. Real-Time High Resolution 3D Data on the HoloLens. In: 2016. Available from DOI: 10.1109/ISMAR-Adjunct.2016.0073.

18. JOACHIMCZAK, Michal; LIU, Juan; ANDO, Hiroshi. Real-time Mixed-reality Telepresence via 3D Reconstruction with HoloLens and Commodity Depth Sensors. In: *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. Glasgow, UK: ACM, 2017. ICMI '17. ISBN 978-1-4503-5543-8. Available from DOI: 10.1145/3136755.3143031.

19. ORTS, Sergio et al. Holoportation: Virtual 3D Teleportation in Real-time. In: 2016. Available from DOI: 10.1145/2984511.2984517.

20. DONG, Samuel; HÖLLERER, Tobias. Real-Time Re-Textured Geometry Modeling Using Microsoft HoloLens. *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2018.

21. LIEVENDAG, Nick. *Autodesk 123D Catch (Discontinued)* [online] [visited on 2019-04-19]. Available from: https://3dscanexpert.com/autodesk-photogrammetry-review-123d-catch/.

22. PHAN, Annie. *ReCap Pro app for iPad Profffdfffd* [online] [visited on 2019-04-19]. Available from: https://knowledge.autodesk.com/search-result/caas/simplecontent/content/recap-pro-app-for-ipad-pro-C2-AE.html.

23. SCANDY CO. *Scandy Pro* [online] [visited on 2019-04-19]. Available from: https://www.scandy.co/products/scandy-pro.

24. SCANDY CO. *Using Scandy Pro scans in AR Lenses or Filters* [online] [visited on 2019-04-19]. Available from: https://www.youtube.com/watch?v=mcDGJodHJKs.

25. SONY MOBILE COMMUNICATIONS INC. *Xperia XZ1 3D Image Creation* [online] [visited on 2019-04-19]. Available from: https://www.sonymobile.com/us/products/phones/xperia-xz1/3d-creator/.

26. MICROSOFT CORPORATION. *Development launchpad* [online] [visited on 2019-04-19]. Available from: https://docs.microsoft.com/en-us/windows/mixed-reality/development.

27. GOINS, Dwight. *Unity vs DirectX performance* [online] [visited on 2019-05-09]. Available from: https://forums.hololens.com/discussion/2552/unity-vs-directx-performance.

28. MICROSOFT CORPORATION. *Unity development overview* [online] [visited on 2019-04-19]. Available from: https://docs.microsoft.com/en-us/windows/mixed-reality/unity-development-overview.

29. MICROSOFT CORPORATION. *Mixed Reality Toolkit* [online] [visited on 2019-04-17]. Available from: https://github.com/Microsoft/MixedRealityToolkit-Unity.

30. DJANGO SOFTWARE FOUNDATION. *Django* [online] [visited on 2019-05-09]. Available from: https://www.djangoproject.com/.

31. PALLETS TEAM. *Flask* [online] [visited on 2019-04-17]. Available from: http://flask.pocoo.org/.

32. AGENDALESS CONSULTING. *Pyramid* [online] [visited on 2019-05-09]. Available from: https://trypyramid.com/.

33. HRONČOK, Miro; VIKTORIN, Petr. *MI-PYT - Flask* [online] [visited on 2019-05-09]. Available from: https://naucse.python.cz/course/mi-pyt/intro/flask/.

34. AUTODESK INC. *ReCap* [online] [visited on 2019-04-19]. Available from: https://www.autodesk.com/products/recap/overview.

35. AUTODESK INC. *About the DeCap Command-Line Tool* [online] [visited on 2019-04-19]. Available from: https://knowledge.autodesk.com/support/recap/learn-explore/caas/CloudHelp/cloudhelp/2017/ENU/Reality-Capture/files/GUID-A9D25D0F-7EBF-4BFB-B94F-66A10EB73D9B-htm.html.

36. CAPTURING REALITY S.R.O. *CapturingReality* [online] [visited on 2019-04-19]. Available from: https://www.capturingreality.com/Home.

37. ALICEVISION. *Meshroom* [online] [visited on 2019-05-01]. Available from: https://github.com/alicevision/meshroom.

38. ALICEVISION. *Meshroom Documentation* [online] [visited on 2019-05-01]. Available from: https://github.com/alicevision/meshroom/wiki.

39. AUTODESK INC. *About the DeCap Command-Line Tool* [online] [visited on 2019-04-19]. Available from: https://peterfalkingham.com/2018/08/11/photogrammery-testing-14-alicevision-meshroom/.

40. AUTODESK INC. *About the DeCap Command-Line Tool* [online] [visited on 2019-04-19]. Available from: https://www.gamefromscratch.com/post/2018/10/18/Creating-3D-Models-From-Photos-Using-Meshroom.aspx.

41. AUTODESK INC. *About the DeCap Command-Line Tool* [online] [visited on 2019-04-19]. Available from: https://www.blendernation.com/2018/08/26/how-to-photoscan-easy-and-free-meshroom-and-blender/.

42. BLENDER FOUNDATION. *Blender* [online] [visited on 2019-04-18]. Available from: https://www.blender.org/.

43. UNITY TECHNOLOGIES. *Unity* [online] [visited on 2019-04-17]. Available from: https://unity.com/.

44. UNITY TECHNOLOGIES. *PhotoCapture* [online] [visited on 2019-04-18]. Available from: https://docs.unity3d.com/ScriptReference/XR.WSA.WebCam.PhotoCapture.html.

45. UNITY TECHNOLOGIES. *WebRequest* [online] [visited on 2019-04-18]. Available from: https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest.html.

46. ERIC5H5. *Vertex limit* [online] [visited on 2019-04-17]. Available from: https://answers.unity.com/questions/255405/vertex-limit.html.

47. ROSS, Amy. Procrustes Analysis. 2005. Available from DOI: 10.1.1.119.2686.

Appendix A

# Acronyms

**API** Application programming interface

**AR** Augmented reality

**MVS** Multi-View Stereo

**RGB** Red green blue

**RGBA** Red green blue alpha

**RGBD** Red green blue depth

**SfM** Structure-from-Motion

**SDK** Software development kit

**UI** User interface

**URL** Uniform Resource Locator

**UX** User experience

**VR** Virtual reality

Appendix B

# Contents of enclosed CD

- `CD1` ..... The first CD
  - `readme.txt` ..... the file with CD contents description
  - `hololens_app` ..... the directory of HoloLens application
    - `hololens_project.zip` ..... the compressed Unity project of the application
  - `thesis` ..... the directory of the thesis
    - `2019-Zderadickova-BSc.zip` ..... LaTeX source codes of the thesis
    - `2019-Zderadickova-BSc.pdf` ..... the thesis text in PDF format
  - `webserver` ..... the directory with the server
    - `webserver.zip` ..... the compressed server
  - `wbdcm` ..... implementation sources
- `CD2` ..... The second CD
  - `readme.txt` ..... the file with CD contents description
  - `testing.zip` ..... the compressed videos from user testing