UiT The Arctic University of Norway

Faculty of Engineering Science and Technology
Department of Computer Science and Computational Engineering

Spatial Mapping Using HoloLens 2

A Proposal for Improvement and an Analysis of Inner Workings

Casper Andrè Levoll-Steen

Master thesis in Applied Computer Science, Narvik, May 2022

“No plan survives first contact with the enemy.”
–Helmuth von Moltke

“The most common error of a smart engineer is to optimize a thing that should not exist.”
–Elon Musk

Abstract

In this thesis we focus on improving the spatial mapping of Microsoft's augmented reality glasses HoloLens 2. Firstly, an in depth analysis of inner workings and limitations, based on public resources, is conducted. This is then followed by a series of experiments in a small and simple indoor environment, the experiments are designed to extract additional information about the mapping which could not be found through public resources. Some of the experiments have also been conducted with a light detection and ranging (LiDAR) device of the type Velodyne VLP–16. A comparison between the two indicate that HoloLens 2 is able to perform at the same level.

The information from the analysis and experiments provide a strong foundation for improvement of the mapping. Only a simple algorithm have been implemented and tested, but in chapter 6 a series of recommendations and ideas for how to proceed with this project are listed. The implemented algorithm uses plane fitting to "pull" points within a certain distance onto the plane. This helps to improve structures that were originally flat, such as walls and floors.

Acknowledgements

Firstly, I would like to thank Børre Bang and my supervisors Tatiana Kravetc, Aleksander Pedersen, and Tanita Brustad for the tips, feedback and resources they have given me along the way.

Secondly, I would like to thank my family, Arild, Bente, Dorthe, Erle, Faya, and Siri, for all the love and support you have given. Without you my education would have been a lot harder and I would not have been able to accomplish all the things that I have.

I am also thankful for the support from my classmates Magnus Hanssen, Markus Tobiassen, and Tanja Henriksen. Without you I might not have been able to finish my degree on time.

Lastly, I would like to thank two unsung heroes that have helped keeping my spirit up: the coffee machine and the 49” widescreen monitor. My education would have been a lot more tiresome and frustrating without you.

Contents

1 Introduction

1.1 Thesis structure

Chapter 1 gives a small backstory and introduction to the theme of this thesis. Chapter 2 presents all the relevant information about HoloLens 2 that is obtainable through official sources. Chapter 3 presents the author's interpretation of the given task, and limitations made based on the given time-frame and information gathered in chapter 2. Chapter 4 presents an in-depth analysis of the spatial mapping meant to extract additional information which could not be found through official data. Chapter 5 presents and discusses the proposed method for improvement of the mapping. Lastly, chapter 6 summarize the project, and gives recommendations and ideas for how to proceed with this project.

Appendix *A* provides instructions for how to access the source code, original data, and how to set up and run the applications. The original task description is attached in Appendix B.

1.2 Background

The term mixed reality (MR) was introduced in 1994 in a paper by Paul Milgram and Fumio Kishino [62, 38], it is a spectrum with the physical world on one side and the virtual world on the other. Between the two extremes you get a mix of the physical and virtual world in various degrees. Perhaps the more familiar mixes include augmented reality (AR), virtual reality (VR), hand-tracking, spatial audio, and spatial mapping – which is the focus of this thesis.

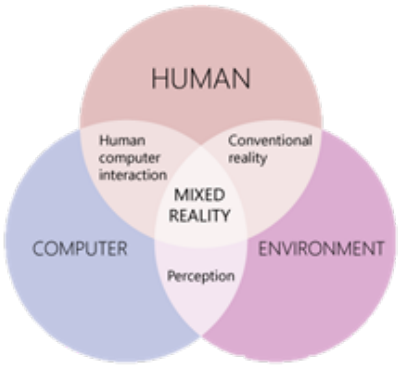
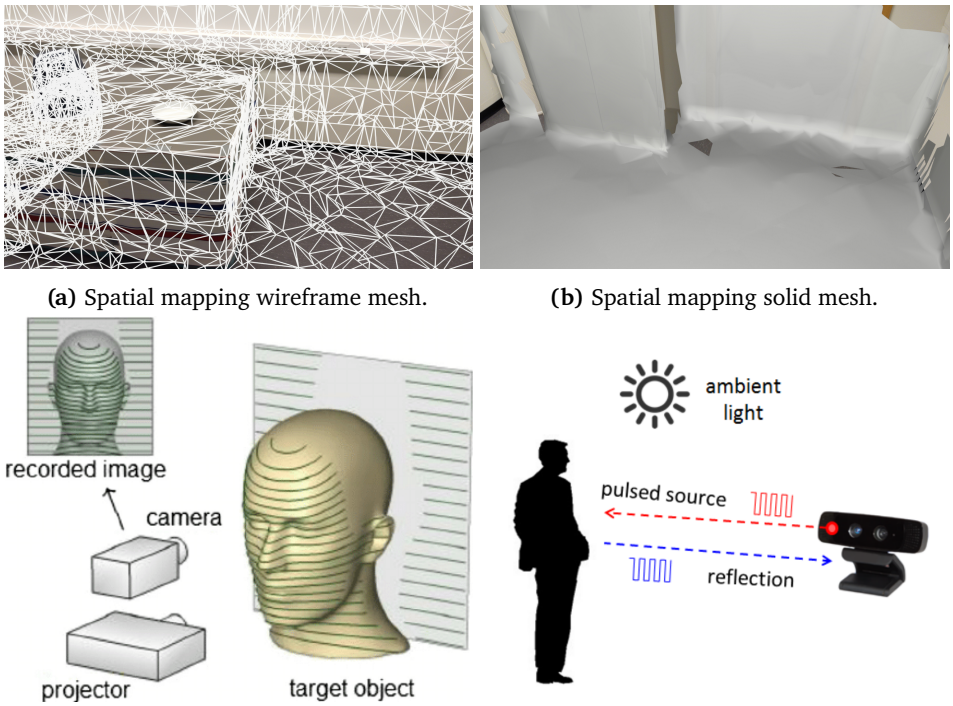


Figure 1.1: Venn-diagram of the interaction between humans, computers and environments [38].

Spatial mapping[1] is a part of the computer vision (CV) field, and its purpose is to provide a detailed representation of real-world surfaces. It is an essential part of making holograms look like they interact with the real world. Early work on this topic, such as Lawrence Roberts in 1963 [67] and Marc Pollefeys in 1998 [22], consisted of extracting 3D-information from 2D images. Today depth sensors in various formats are the most common way to extract 3D-information in MR context. Especially time of flight (TOF) sensors, which "flood" the scene with infrared light and measures the time it takes for the light to bounce back, are widely used.

1. Also known as 3D surface reconstruction



(a) Spatial mapping wireframe mesh.

(b) Spatial mapping solid mesh.

(c) Structured light concept, used on Kinect v1 [16].

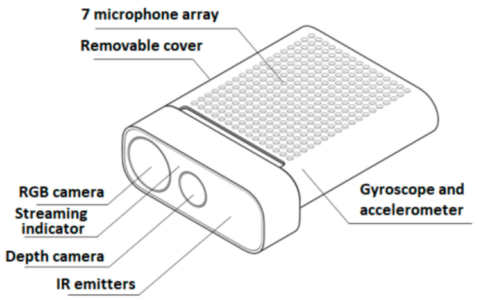
(d) Time of flight concept [25].

Figure 1.2: Examples of depth sensors and how spatial mapping can be visualized in an application.

Microsoft's Kinect camera is a TOF sensor[2] and was first released as a complement to the Xbox 360 gaming console in 2010. The low price and quality of the camera, combined with the fundamental work of i.a. Curless and Levoy in [4], Newcombe et al. in [63], Hoppe in [13] and Klein and Murray in [21], quickly made it popular with scientists, engineers and hobbyists. It helped lay the foundation for MR and spatial mapping as we know it today [84, 65].

2. Not the 1st generation

(a) From left to right: Kinect v1, v2, and v4 (Azure Kinect) [71].



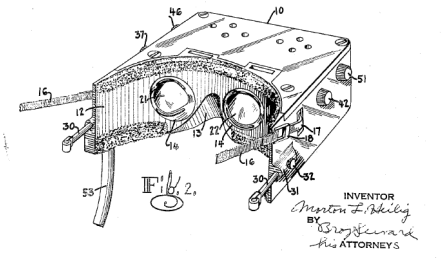
(b) Diagram of components on Azure Kinect [40].

Figure 1.3: Kinect cameras.

VR technology has seen a massive increase in interest and development the last decade, but it is not a new concept, and can in fact be traced back to early 1960s. Around the same time as Lawrence Roberts published his PhD on computer vision; Morton Heilig got patents on a pair of VR-glasses [28] and a VR simulator [27].

Fast forward almost 60 years to 2019 and we arrive at the launch of Microsoft's HoloLens 2[3], which is the 2nd generation of their cutting edge AR glasses and the research object of this thesis. The glasses are fully self-contained and have many sensors and features. For depth sensing and tracking in particular the glasses are equipped with an Azure Kinect camera module, an inertial measurement unit (IMU), and 4 head tracking cameras.

(a) HoloLens 2.



(b) VR glasses from 1960, named "Stereoscopic Television for Individual Use" [28].

Figure 1.4: Illustration of HoloLens 2 and VR glasses from 1960.

3. Hereinafter referred to as "HoloLens" or "the glasses"

2

About HoloLens 2

The aim of this chapter is to use existing resources to create a solid foundation for how the glasses work, with special attention to spatial mapping and rendering. This foundation will be used when a deeper analysis of the mapping is done in chapter 4. The following sources have been used to gather information: Microsoft documentation [39, 43, 44, 45, 46, 40, 49, 50, 51, 47, 42], two talks from Marc Pollefeys [65, 64], a talk from Alex Kipman [18], a talk from Elene Terry [70], notes from Elene Terry's talk [5], an evaluation of Azure Kinect [71], and documentation for research mode [80, 79, 78].

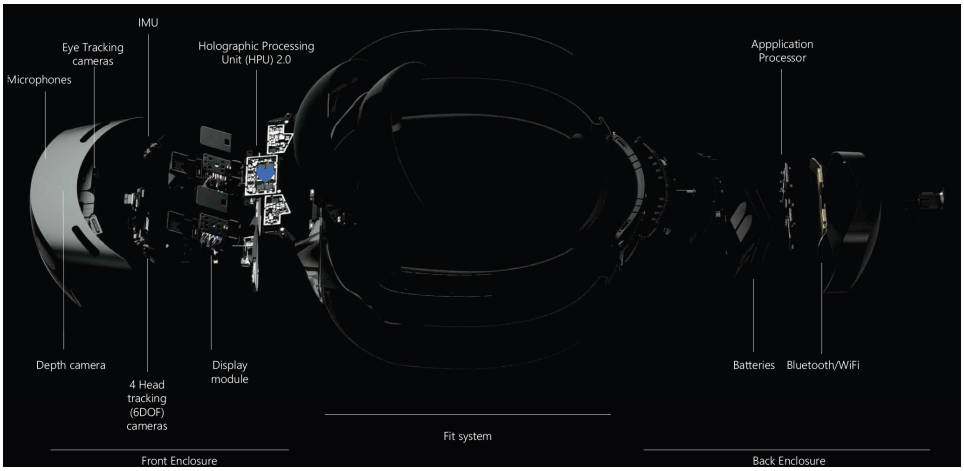


Figure 2.1: Overview of hardware components [70].

2.1 Position tracking

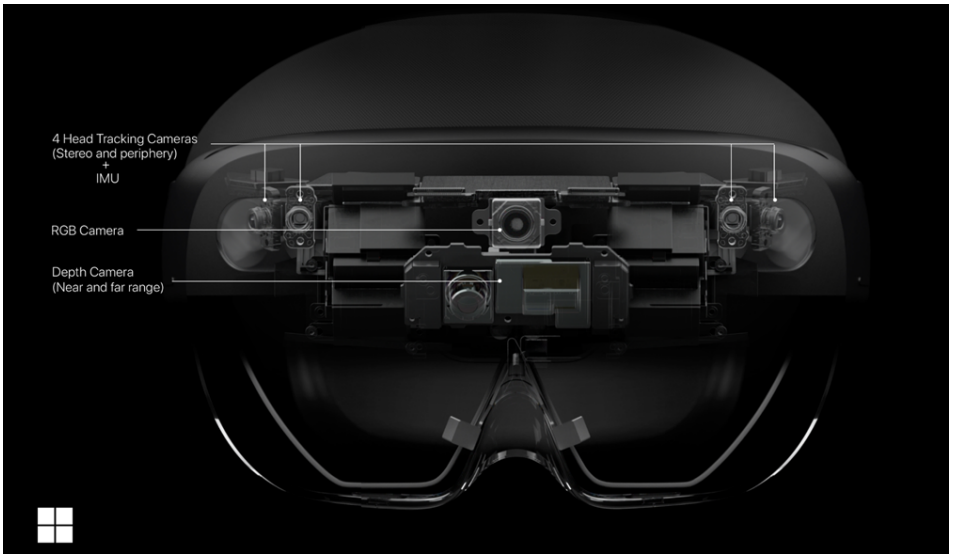


Figure 2.2: Close-up of the sensors for spatial mapping and position tracking [39].

The glasses use 4 gray-scale cameras and an IMU to orient itself. Two of the cameras point forward and the other two to the side, at a high level their functionality can be compared to human stereo and periphery vision. They operate at 30 FPS and 640×480 resolution, and are used to help avoid accumulating drift by predicting motion and detecting landmarks. Since the cameras can recognize landmarks, the glasses also have the ability to remember locations they have visited. The IMU is of the type LSM6DSM [69] and consists of an accelerometer for linear acceleration along the x, y and z axes (and gravity), a gyroscope for rotations, and a magnetometer for absolute orientation estimation. Together they can perform visual-inertial simultaneous localization and mapping (VISLAM) with low drift and latency.

2.2 Depth sensor

Figure 2.3: The Azure Kinect module separated from its housing [18].

As mentioned in section 1.2 the depth sensor uses the TOF concept and consists of a shutter sensor with a resolution of 1 MP and two lasers. The lasers emit photons and the shutter "measures" the returning photons. One laser points downward and is used for articulated hand tracking (AHAT), it operates at a wide angle and 45 FPS up to 1 meter away. The other laser points forward and is used for computing the spatial mapping on the glasses, it operates at a narrow angle and 1–5 fps.

2.3 Displays

The human eye is not able to focus on objects as close as the HoloLens displays. In order to solve this problem, a branch of optics known as waveguides are used, in short the light-waves/photons are guided to the back of the eye. Each display consists of 3 micro electro mechanical system (MEMS) lenses, one for each of the colors red, green and blue, and each lens consists of millions of mirrors/gratings (about 100nm in size). When a photon enters the lens it bounces between these mirrors until it hits total internal reflection (TIR), and shoots to the back of the eye. The displays have a holographic resolution of 2K and a holographic density of >2.5k radiants (light points per radian).

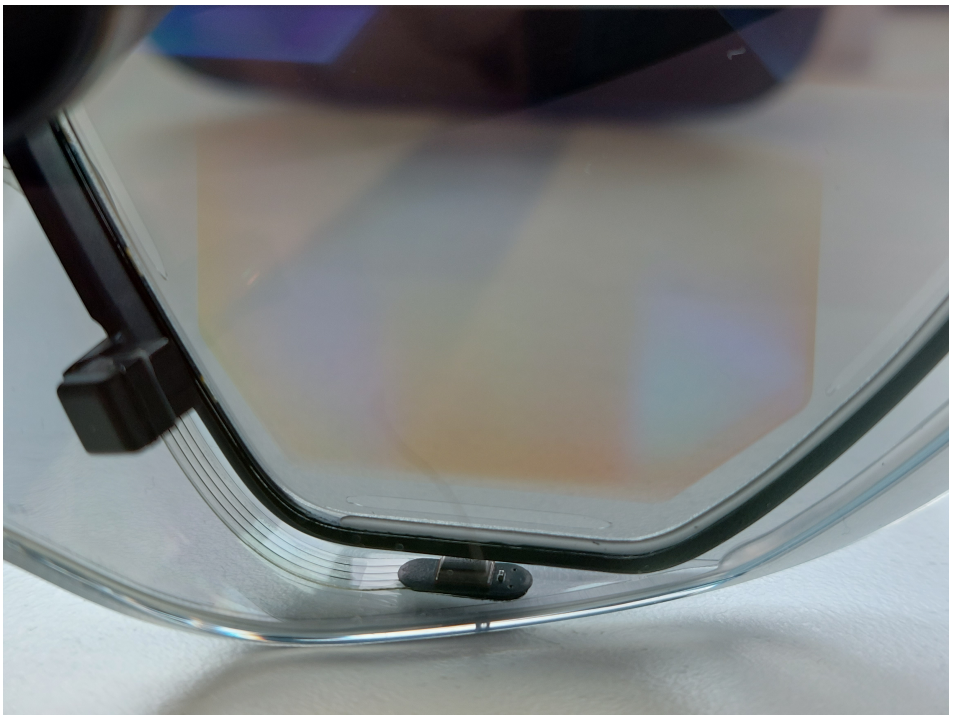


Figure 2.4: A display, the brown/yellow area are the lenses.

3 lasers per eye are used to emit red, green and blue light. In the process of distributing the light, it first hits a fast tracking mirror which oscillates at 12kHz, and then a slow tracking mirror oscillating at 120Hz. The fast mirror is responsible for the horizontal direction and the slow mirror for the vertical. Note that this is a simplified explanation, there are other processes to, such as dis-speckling[1] of the light.

1. Speckles = granular pattern

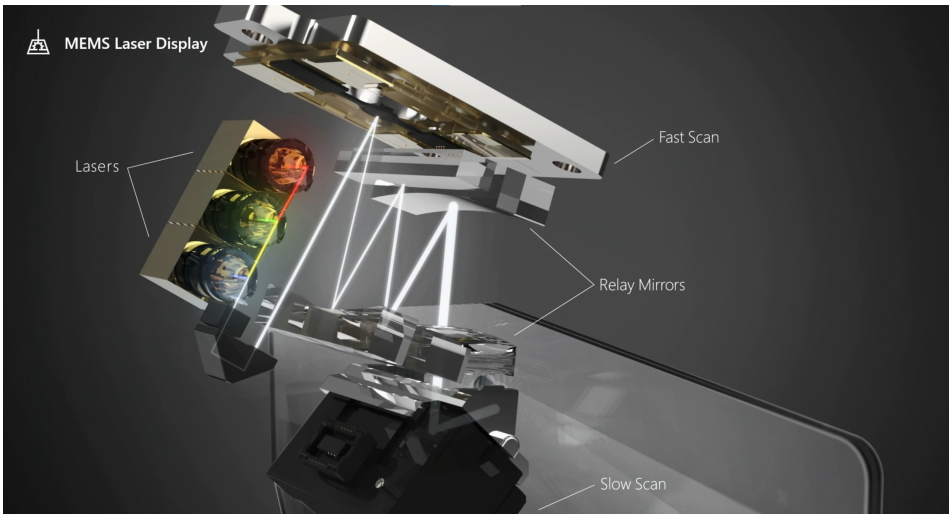


Figure 2.5: Illustration of waveguide process [18].

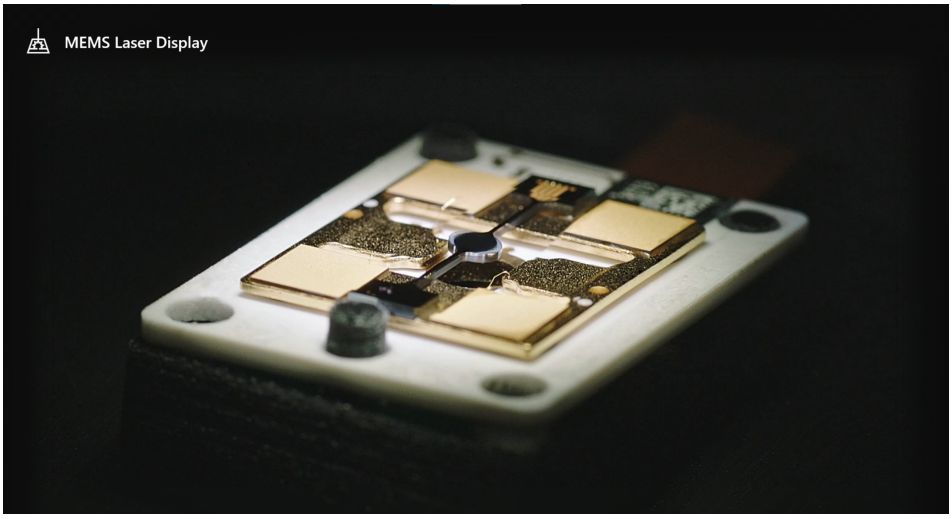


Figure 2.6: View of the fast tracking mirror under a microscope. The mirror spreads the light horizontally [18].

2.4 Processors

There are 2 components on the glasses that are responsible for computation and connectivity: a custom-made holographic processing unit (HPU) located at the front, and a system on a chip (SOC)[2] located at the back. Each of them

2. Also known as application processor

are designed to handle specific workloads, and they communicate via one PCIe 2.0 connection at up to 100 MB/s. The HPU is responsible for receiving and processing all raw sensor data and making final adjustments on holograms when they are returned from the SOC. The SOC is a Qualcomm Snapdragon 850 Compute Platform [66] and is responsible for handling the processed data, applications, connections, storage, memory, and CPU- and GPU-operations. It is important to note that there is a strict separation between front and back; the SOC only gets results and API-calls (e.g. hand- and eye-position).

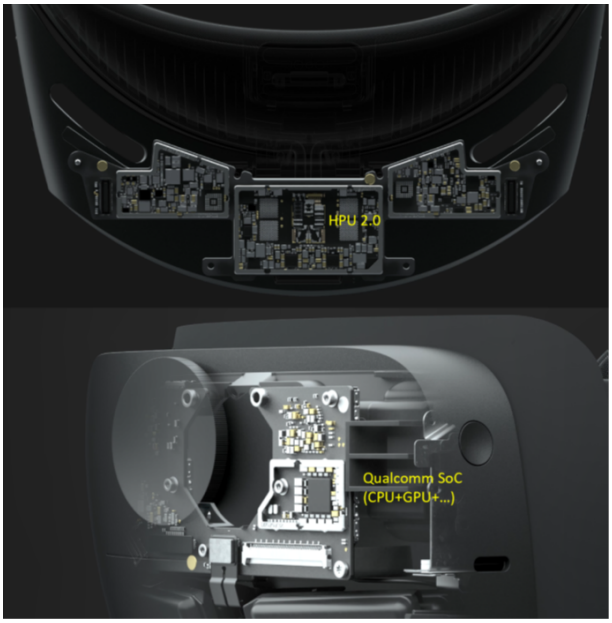


Figure 2.7: Close-up of the HPU and SOC. The two components on either side of the HPU are the light engines [78].

Since the displays are see-through, which means that the real world comes in at 0 ms latency, it is necessary to also have low latency on the holograms. When the user moves around, the inverse motion have to be applied to the holograms to make it look like they belong in the real world. If the hologram-latency goes above 9ms the user will start to notice lagging [18, 65]. Figure 2.8 illustrates the process that makes it possible to stay below the 9ms limit. First the HPU predicts where the glasses will be when it is time to render an image (rendering-time is also taken into account). When the image is done rendering and ready for drawing, a last-second late stage reprojection (LSR) is done. This essentially shifts/warps the image a bit to get the correct viewpoint based on the latest IMU data.

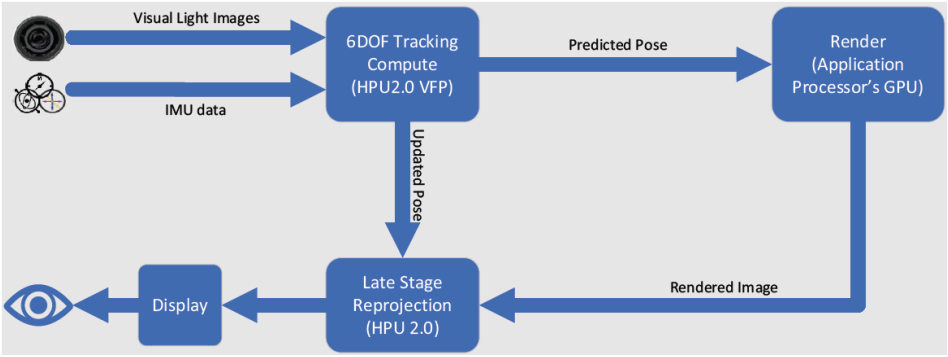


Figure 2.8: An illustration of the pipeline from raw sensor input to display [70].

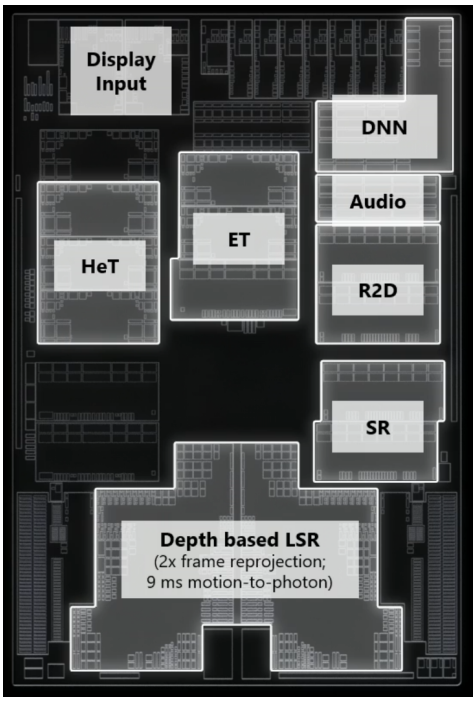
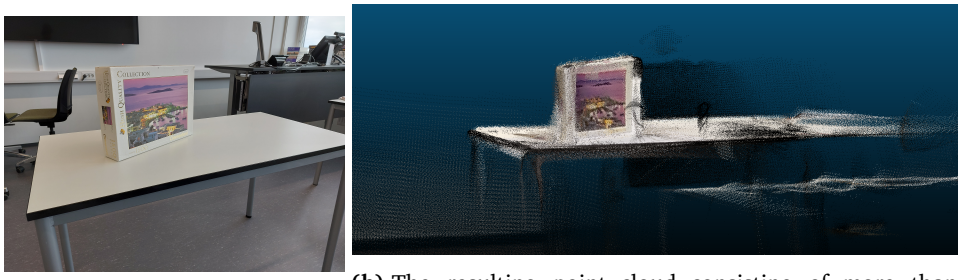


Figure 2.9: Detailed view of the different parts of the HPU. DNN = deep neural network AI core (makes inferences on raw data), R2D = raw data to depth data, SR = surface reconstruction, HeT = head tracking, ET = eye tracking, LSR = late stage reprojection [65].

2.5 Spatial mapping

There are three ways of acquiring the spatial mapping data, but only one of them are suitable for deployed applications if you want to manipulate the data real-time. The first method is to acquire the raw point clouds by putting the glasses in "research mode", which provides access to an extra set of API-calls. However, this mode is not intended for applications in enterprise environments or in other distribution channels such as Microsoft Store [42].



(a) The real model (puzzlebox on a table).

(b) The resulting point cloud consisting of more than 7 000 000 points (the entire cloud is not visible in figure).

Figure 2.10: An illustration of what an extracted point cloud can look like.

Before the second and third method is presented, an explanation of how the glasses store the environment is necessary. As the glasses explore the environment they divide it into multiple surfaces in a way that makes sense for them. Each mesh[3] has its own globally unique identifier (GUID), and lives in its own local coordinate system. API-calls exists for transforming between these systems (and the global system) [61]. All coordinate systems use meters as unit for length.

The second method is through the Windows Device Portal, which is a developer tool that lets you configure and manage the glasses and can also help to debug applications. Figure 2.11 illustrates the "3D View" tool which lets you see the mapping and also save it in .obj format. With this method you can not extract the data in real-time and you can not manipulate it in an application.

3. The terms "surface" and "mesh" will be used interchangeably, but they mean the same

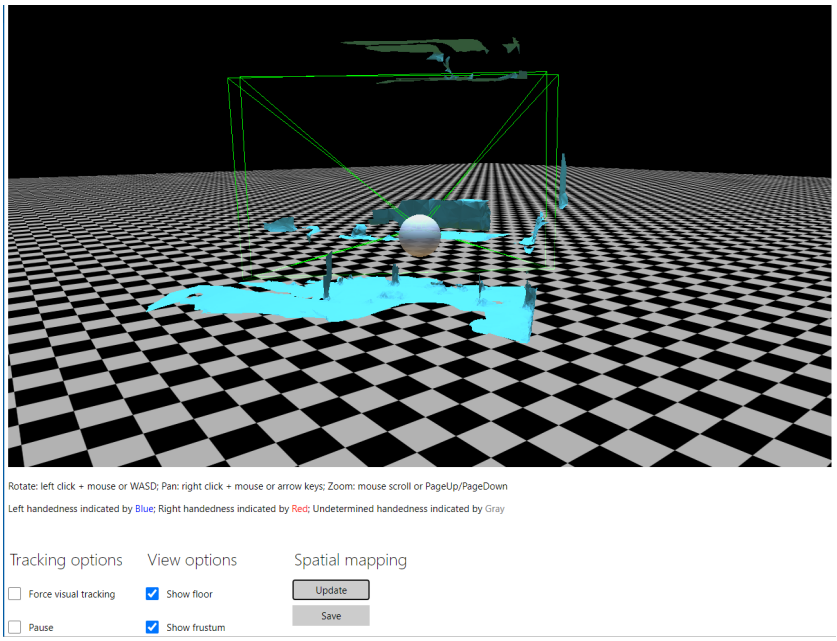


Figure 2.11: 3D View in Device Portal, the blue mesh is the mapping.

The last method is to use the API-calls that are available to applications in normal mode. The basic flow is that the glasses continuously explore and improve its internal understanding of the environment. An application can express interest in this information by subscribing to an update event and specifying a bounding volume (e.g. bounding box, sphere or frustum). When the set of surfaces within that volume has changed, the application is notified and receives information which it can reason about (either add, update or remove a surface). For each mesh it is possible to ask the glasses to compute a triangulation with a maximum resolution (triangles per cubic meter). The application can then draw, manipulate and export this data as it sees fit (but it has no effect on the glasses' internal understanding).

2.6 Scene understanding

Scene understanding is designed to make applications environmentally aware by providing developers with a high-level representation of the environment. It can be thought of as a wrapper of helper-functionality around the spatial mapping. It is important to note that it can be computationally expensive to compute scene understanding (up to multiple seconds).

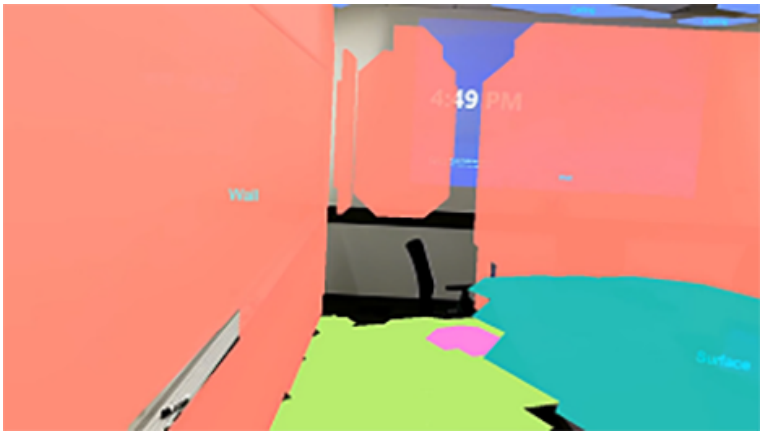
The understanding can be computed in 3 (related) categories:

- A watertight environment by using AI and the spatial mapping to make inferences about the planar structure of the environment as shown in figure 2.12*b*
- Plane regions (not watertight) based on the spatial mapping (called Quads)
- A mesh/triangulation that aligns with the planar surfaces (spatial mapping)

The environment is built using the following constructs:

- SceneComponents – the most basic decomposition, they have their own unique ID and can for instance be a mesh, quad or a bounding box
- SceneObjects – represents real world objects such as background, wall, floor, ceiling, and platform
- SceneMesh – an approximated triangulation of a SceneComponent
- SceneQuad – bounded rectangular surfaces that represents 2D approximations of surfaces in the 3D world

Common use-cases for scene understanding include placement scenarios, occlusion, physics, navigation, and visualization (of the constructs).
To summarize: spatial mapping provides the least amount of latency and highest accuracy whereas scene understanding provides structure and simplicity.



(a) Scene understanding with no inferences.

Figure 2.12: Scene understanding with inferences disabled and enabled [44].

(b) Scene understanding with inferences (creates a watertight mesh).

Figure 2.12: Scene understanding with inferences disabled and enabled [44] (Cont.).

3

Problem description

This section explains the author's interpretation of the given task, decisions made based on the acquired information about the glasses, and limitations made in order to meet the deadline. The original description can be seen in appendix *B*.

The task consists of two parts, and they are interpreted as such: (1) analyze limitations, weaknesses, and inner workings of the glasses with focus on spatial mapping. This will lay the foundation for (2) developing an algorithm that can improve the mapping in real time. There are mentioned some tools and algorithms in the description, these are considered as suggestions for where it can be a good idea to start.

Although it is possible to extract the raw point clouds generated by the sensors, the focus will be on the mesh computed internally by the glasses. There are two main reasons for this: firstly, as mentioned in figure 2.10*b*, the raw point clouds can easily consist of millions of points. This amount of points is not suitable to process on the glasses, and it would take too long to develop a solution that could handle it (e.g. remote computation). Secondly, the raw data is only available when the glasses are in "research mode", which would drastically narrow the usability of the developed algorithm.

Another limitation made in order to meet the deadline is to work in a small, simple and static environment. This implies that problems associated with creating world-scale experiences [43, 14] such as drift of holograms and multiple coordinate systems are not taken into consideration. The goal is to find and improve the major weak-points in simple environments first, and then gradually transition to larger and more complex environments.

4

Analysis of the spatial mapping

This chapter describes the methods, tools and results of the conducted experiments. The purpose is to extract information about the spatial mapping which was not obtainable through documentation or related work.

4.1 Environment

Microsoft's guidelines for environment construction [48] is used as foundation for designing the testing area. Figure 4.1 shows the entrance of the room from the inside. Figure 4.2 shows the view from the entrance and inwards, this perspective is assumed when referring to the walls as left or right. Figure 4.3 shows the view from the left wall, on the table is a blue flower pot flipped upside down and a ceramic bowl, this perspective is assumed when referring to objects relative to the table. Notice the pipes and the "double" corner at the bottom left, this is the only "clean" corner in the room. Figure 4.4 shows the room seen from the right wall, notice the vertical list ornaments on the wall. Lastly figure 4.5 shows a baking bowl that was added in retrospect of the initial experiments.

Figure 4.1: The entrance seen from inside the room.

Figure 4.2: View inwards from the entrance.

Figure 4.3: The room seen from the left wall.

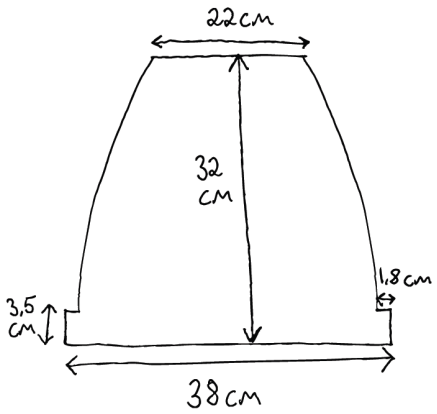
Figure 4.4: The room seen from the right wall.

(a) Sideview. **(b)** Topview.

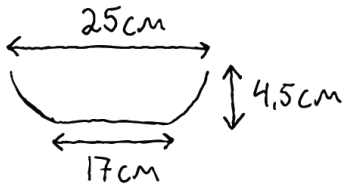
Figure 4.5: A baking bowl.

The room and objects has the following noteworthy characteristics:

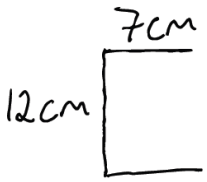
- Room
  - Dimensions (WHL): 3.08m × 3.11m × 6.45m
  - Fluorescent lights
    - Frequency: 50Hz
    - Lux-values: ~ 600 directly under light, ~ 200 on bright area of wall and table, ~ 70 on floor and darker areas (shadow)
- Flower pot
  - Volume: 0.0261m^3
  - Dimensions:



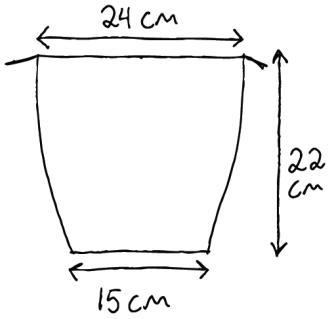
- Bowl
  - Dimensions:



- Electric channel above table
  - Gap between table and lower edge of channel: 12.5cm
  - Gap between floor and lower edge of channel: 84.5cm
  - Dimensions:



- Baking bowl
  - Dimensions:



All dimensions have been measured manually with a tape measure.

4.2 Experiments

Some experiments are based on related work about HoloLens 1, some on personal observations and ideas, and some have been added in retrospect due to

observations from the initial tests.
It is important to note that the experiments are not designed to do an *exact* analysis of accuracy, but rather get indications of accuracy in a realistic usage setting, and to get a deeper understanding of how the mapping is constructed. For an exact analysis the data sets need to be larger, the environment more dynamic, and factors like warm up time [71, 14] needs to be taken into account.

A light detection and ranging (LiDAR)-scanner of type Velodyne VLP-16 Puck[1] [83] will be used as benchmark against the original and improved spatial mapping.

Performance (HoloLens)

Description: measure the resource-usages of the glasses in 3 scenarios:

- When no app is running, i.e. the glasses are on, but idle
- When the app for spatial mapping is idle, i.e. not rendering or updating meshes
- When the app for spatial mapping is running with different resolutions, starting at 1 000 triangles per cubic meter

Purpose: see how much processing power is left for improving the meshes, and find a reasonable trade-off between performance and mapping resolution. The final resolution will be used to map the entire test environment, and this mapping will be used in the subsequent experiments that require mapping data. The scanning will try to comply with the guidelines from Microsoft [41].
Execution: each scenario will be run for 15 minutes, and data will be collected through the Device Portal. The glasses' internal cache for spatial mapping will be reset between each run. A bounding box of 5m×3m×3m is used as volume for updating meshes.

IMU (HoloLens)

Description: record IMU-data (accelerometer, gyroscope, magnetometer) over a period of 30 minutes.
Purpose: measure drift of values.
Execution: the glasses will hang still on a tripod during the recording.

1. Hereinafter referred to as "the LiDAR"

Flat surfaces (HoloLens and LiDAR)

Description: get vertex data from the floor and walls.
Purpose: measure global accuracy and local precision. Compare the results with the evaluation of HoloLens 1 in [17], the evaluation of Azure Kinect in [71], and with the LiDAR.
Execution: use plane fitting to compute the root mean squared error (RMSE) value, measure angles between the plane normals, and measure distance between the planes.

Sharp transitions (HoloLens)

Description: get vertex data from a small and large 90° transition.
Purpose: measure local precision/"tightness" of sharp transitions.
Execution: measure per-vertex distance from the bounding box of the transitions.

Convex/concave surfaces (HoloLens)

Description: get vertex data from convex and concave surfaces.
Purpose: see how well the glasses deal with convex and concave objects. Yang Liu et al. in [26] found that HoloLens 1 struggles most with concave surfaces/angles.
Execution: get vertex data from the flipped flower pot (convex), and the deep and shallow bowl (concave). Use volume calculation and visual comparison against reference models to determine accuracy.

Triangle coloring (HoloLens)

Description: color the triangles of each surface.
Purpose: get a better understanding of how surfaces are built and look for any patterns.
Execution: each surface will fade from black to white (the first triangle is black, and the last is white).

4.3 Choice of technology

An extensive study of tools and libraries has been conducted for this thesis. The goal is to find a setup that best suits the task, makes the best use of the

author's experience, and that makes the thesis as survey-able and transparent as possible.

4.3.1 Author's experience

Courses

- DTE-3610 Finite element methods, programming [76]
- DTE-3609 Virtual reality/graphics/animation - theory [75]
- DTE-3607 Advanced game and simulator programming [74]
- DTE-3605 Virtual reality, graphics and animation - project [73]
- DTE-3604 Applied geometry and special effects [72]

Tools

- Microsoft Visual Studio
- JetBrains products and solutions
- Blender

Programming languages and libraries

- C++ (11–20)
- Python
- OpenGL, glm, GLFW
- WebGL, Three.js
- Gmlib (in-house C++ library at UiT Narvik for geometric modelling)
- Boost C++ libraries
- Blaze C++ math library

4.3.2 Review of tools and libraries

OpenXR

OpenXR [52] is an API-standard created by the Khronos group. The purpose of the API is to simplify software development for AR and VR by providing cross-platform access to device runtimes and making applications able to run on any system that exposes the OpenXR APIs. OpenXR also has an extension mechanism which allows vendors to expose additional functionality beyond the core features (e.g. scene understanding, spatial mapping and holographic remoting for HoloLens 2).

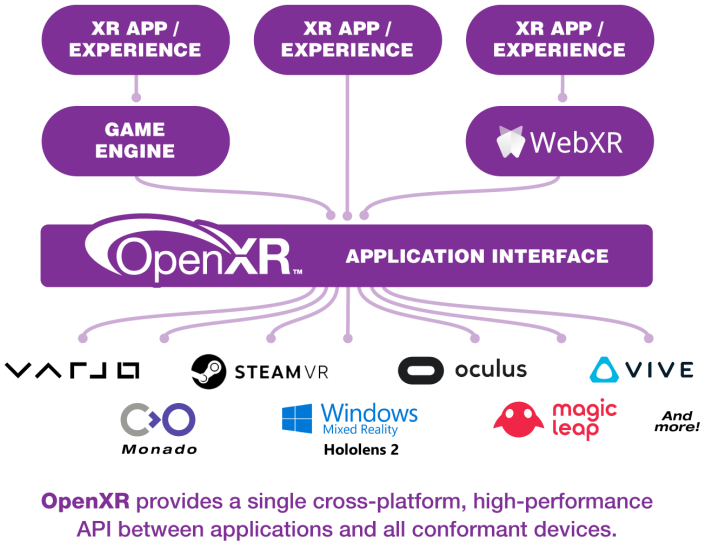


Figure 4.6: Illustration of how OpenXR works [52].

Mixed reality toolkit

Mixed reality toolkit (MRTK) [57, 31] is a project driven by Microsoft and its purpose is to accelerate cross-platform MR app development in the Unity game engine. Some of the included features are scene understanding, spatial mapping, UI controls and example scenes. There is also developed an MRTK-version that works in Unreal Engine, however this version is at a far earlier stage than the one for Unity.

Figure 4.7: An example scene from MRTK [57].

Windows universal samples

Universal windows platform (UWP) was introduced in Windows 10, and its purpose is to help develop universal applications that can run on all platforms created by Microsoft (e.g. Windows 10/11, Xbox and HoloLens). It provides UI and asynchronous features that are ideal for internet-connected devices by using the native API provided by the operating system. The API is implemented in C++ and is supported in C#, Visual Basic, C++ and JavaScript [58].

"Universal Windows Platform (UWP) app samples" [32] is a repository maintained by Microsoft. As the name suggests, it contains a wide array of sample apps that demonstrate the API usage patterns in the Windows Software Development Kit for Windows 10. One app of particular interest is the "Holographic spatial mapping sample" which demonstrates how to extract and visualize the spatial mapping data using C++, UWP and DirectX. The visualizations demonstrated in figures 1.2*a* and 1.2*b* are created with this app.

HoloLens2ForCV

HoloLens2ForCV [77] is a repository created by Microsoft. It provides access to the research mode API-calls and UWP sample apps that handles the raw data streams (depth camera, gray-scale cameras and IMU). Its purpose is to make it easier to use the glasses as a Computer Vision and Robotics research device. The sample app "StreamRecorder" was used to extract the point cloud seen in figure 2.10*b*.

Holographic remoting

Holographic Remoting [55] is the process of streaming holographic content from and to the glasses in real time. There are two main uses of this: (1) previewing or debugging the application during development by running the app locally in the Unity or Unreal game engines, and then stream the experience to the glasses. (2) using the resources of a PC to power the app instead of using the glasses' onboard resources. This lets the user experience the app on the glasses, while it actually runs on the PC. The first method is called a "Holographic Remoting *Player* app", and the second method is called a "Holographic Remoting *Remote* app". For the latter method it is recommended to use the OpenXR API with either Unreal Engine, Unity or an UWP application. A repository containing example applications can be found here [56].

Microsoft visual studio

Visual Studio [35] is an integrated development environment (IDE) created by Microsoft, it features all the tools needed to edit, debug, build and publish an application. It is available for both Windows and Mac and comes in 3 editions: Community (free version with the most important features), Professional (not free, almost fully featured and designed for individuals or small teams), and Enterprise (not free and fully featured for teams of any size). By using the Visual Studio Installer tool you can customize the workloads/features that you want the IDE to provide, e.g. desktop development with C++, UWP development, or game development with C++/Unreal Engine or C#/Unity. The latest version of Visual Studio is the 2022 release which, among other things, features significant performance improvements (by making it a 64-bit application) and better developer tools for C++. Another thing to note is that Visual Studio has been around for many years and is well integrated into the development industry. This means that it comes with a large supportive community, a lot of documentation, and many tutorials [37, 6].

Unreal Engine

Unreal Engine (UE) [10] is a game engine developed by Epic Games and is the tool suggested in the original task description. Main features include:

- Free to use for learning, and developing internal projects. All features are available from the beginning.
- Many free asset samples are available
- Blueprints – a UE scripting tool for assets, visuals, behaviors and ease of use
- C++ in e.g. Visual Studio – for runtime performance, fundamental code, engine functionality, and external libraries
- Debugging, streaming, and launch/deployment of applications
- A decent community for HoloLens
- Good tutorials such as app development [7], spatial mapping [9] and scene understanding [53]
- Good plugins such as Mixed Reality Toolkit [31], OpenXR [30] and [29] for research mode.

Unity

Unity [81] is a game engine developed by Unity Technologies, it has the following main features:

- Free to use for learning, and developing internal projects. All features are not included from the beginning.
- Many free asset samples are available
- Scripting with the C# language in e.g. Visual Studio
- Debugging, streaming, and launch/deployment of applications
- A very good community for HoloLens
- Good tutorials such as app development [33], spatial mapping [60] and scene understanding [45]
- Good plugins such as Mixed Reality Toolkit [34], OpenXR [59] and [12] for research mode.
- Has a tendency of getting updates and new features a while before Unreal Engine

Rider

Rider [15] is an IDE developed by JetBrains and can, among other things, be used for writing C# for Unity and C++ and Blueprints for Unreal Engine. The support for Unity was introduced in 2017 and has by now been deeply integrated with the engine. At time of writing the support for Unreal Engine is still in beta-version, which means that it could be prone to bugs and has not had the time to build a large community. Note that there is no free version of Rider available and that it, unlike Visual Studio, provides the same features on Windows, Mac and Linux.

Blender

Blender [2] is an open source and free cross-platform 3D creation suite, it supports the entire 3D pipeline such as modeling, animation, simulation and rendering. It is also possible to customize Blender and create specialized tools by using the Python programming language combined with the Blender API.

CloudCompare

CloudCompare [3] is a free and open source program designed to do scientific analysis of and comparisons between 3D point clouds and triangular meshes. Some of its features include plane fitting and computation of curvature, density and volume.

LidarView

ParaView is a free and open source application produced by Kitware, it provides functionality for loading, showing and interacting with data. LidarView [20] is a wrapper-application around ParaView, and provides tools specifically designed for LiDAR data, both real-time and post-processing. Functionality of particular interest include recording and playback of scans, simultaneous localization and mapping (SLAM)-algorithm for mapping without GPS/IMU-data, and exportation to "LAS" file format for further processing in other applications.

LidarView is actually a brand-independent version of VeloView, which Kitware is developing with Velodyne. Both provide the same functionality, but the process for setting up VeloView can be a bit lengthy, and LidarView provides everything out-of-the-box [20].

4.3.3 Chosen setup

The original setup for extracting and manipulating the mapping consisted of using Unreal Engine, OpenXR, MRTK and Visual Studio combined with a tutorial on spatial mapping [9]. This seemed like an ideal solution since the author has experience with some of the tools, and it is relatively easy to expand with user interface and visual effects while maintaining a high level of control and flexibility with C++. However, as far as the author can tell, it is not possible to extract or manipulate the mesh data, only visualize it. The "MRMesh" class [8, 54], which doesn't seem to have any option for extraction or manipulation, was the closest it was possible to get to the data.

Since Unreal Engine is not an option, the chosen setup became Visual Studio combined with the "Holographic spatial mapping sample app" from the UWP repository. The main reasons for this are that minimum effort has to go into learning a new tool or language and that the app is designed to solely extract and visualize the spatial mapping, which is the most critical part of this project. Expanding the project with e.g. scene understanding, remote rendering and UI presents an opportunity for future work (see chapter 6).

CloudCompare, Blender, and LidarView will be used for analysis and post-processing. The main reasons for choosing this combination is that they are all free of charge, open source, and complement each other and the other tools nicely. For instance a LiDAR-recording can be loaded and processed in LidarView, exported to a "LAS"-format for processing in CloudCompare, which can export it to a "ply" format for processing in Blender.

4.4 Results

4.4.1 HoloLens 2

Performance

Table 4.1 shows the collected data:

Table 4.1: Resource usage in various scenarios.

Scenario	FPS	CPU	GPU	RAM
No app	60	30% ± 10%	22.5% ± 2.5%	~ 1.9/4.0GB
App idle	60	~ 50%	~ 25%	~ 1.9/4.0GB
1 000 t/m^3	60 – 30	65% ± 5%	22.5% ± 2.5%	~ 1.9/4.0GB
1 500 t/m^3	60 – 30	65% ± 5%	20% ± 2.5%	~ 2.0/4.0GB
2 000 t/m^3	60 – 30	65% ± 5%	20% ± 2.5%	~ 2.1/4.0GB
2 500 t/m^3	60 – 30	65% ± 5%	20% ± 2.5%	~ 2.0/4.0GB
3 500 t/m^3	60 – 40	70% ± 5%	25% ± 2.5%	~ 2.0/4.0GB
4 500 t/m^3	60 – 30	65% ± 5%	20% ± 2.5%	~ 1.8/4.0GB
8 000 t/m^3	60 – 25	65% ± 5%	17.5% ± 2.5%	~ 1.9/4.0GB

The major takeaways from this experiment are that the CPU is the limiting factor, and that once the CPU reaches about 70% utilization the process throttles and FPS decreases. In the scenarios where the mapping application is running, the FPS starts at 60 and after 5-8 minutes it has dropped and settled at a lower value, usually around 30 FPS. If the app is idle then around 50% of CPU is available to use for improving the mesh.

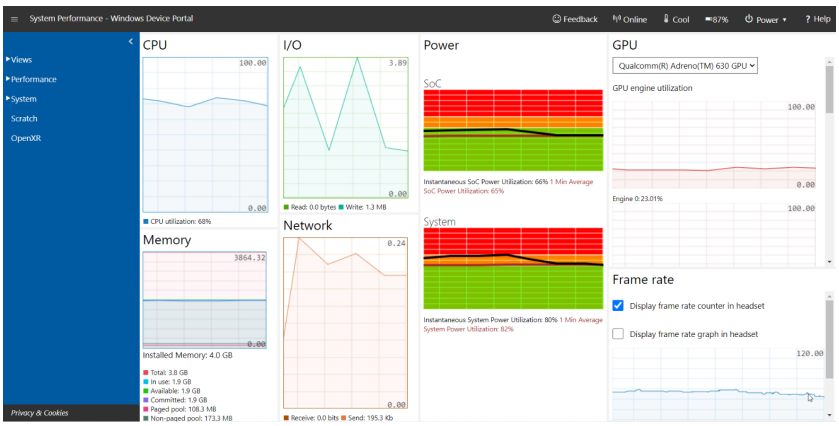


Figure 4.8: A snapshot of the overview in the Device Portal.

8 000 triangles per cubic meter is set as a reasonable trade-off between resolution and performance, 25–30 FPS is okay as long as no fast or sudden movements are executed. This of course is a personal opinion, people respond differently to VR experiences as described by Barrett in [1].

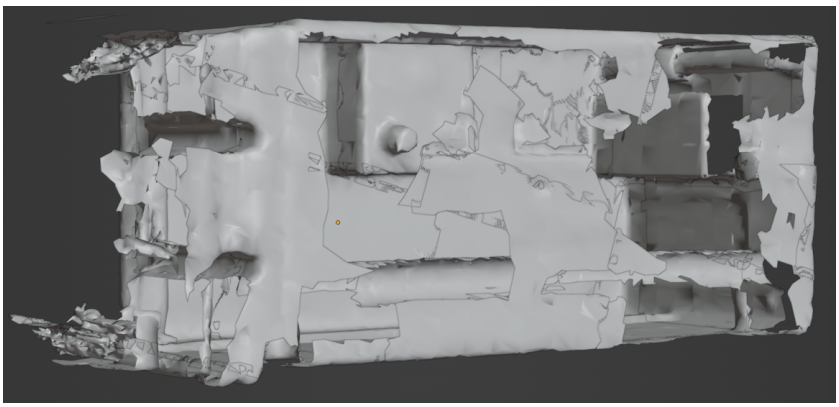


Figure 4.9: Original scan of the room, the roof and the mesh outside the windows will not be used in the experiments.

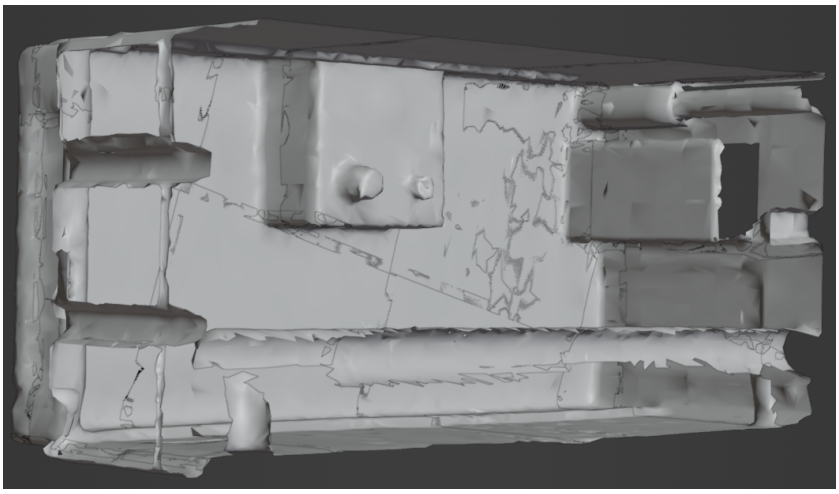


Figure 4.10: The model after removing the roof and the mapping outside the window.

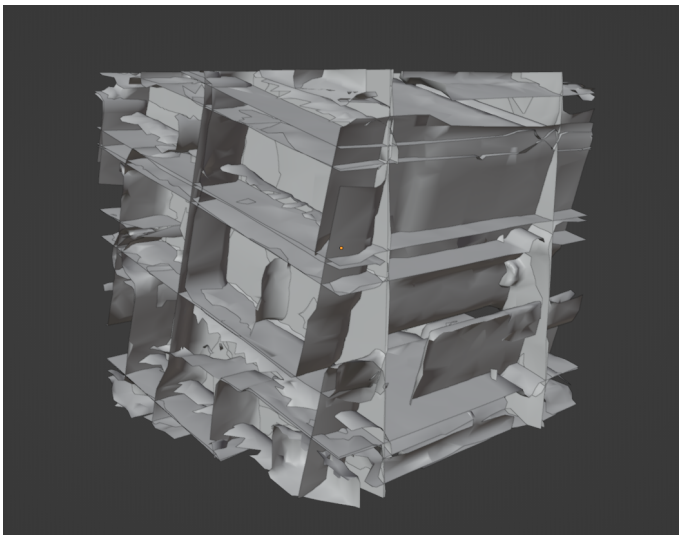


Figure 4.11: All meshes in their own local coordinate system.

IMU

The values in figure 4.12 and 4.14 has been normalized to a value between −1 and 1 in order to make the x, y, and z values fit into the same graph. See appendix *A* for how to access the original data.

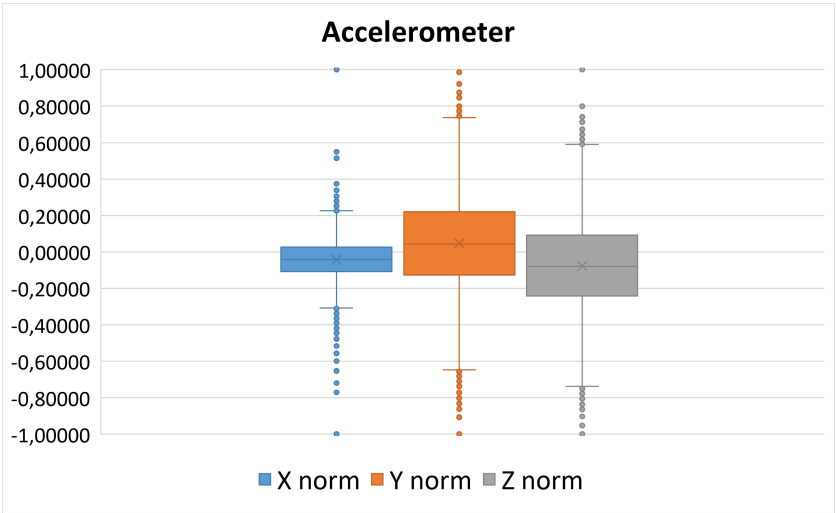


Figure 4.12: Distribution of normalized values for accelerometer.

Table 4.2: Additional data for accelerometer (not normalized).

	X	Y	Z
Average/mean	9.80842	−0.76339	−0.20520
RMSE/Std. Dev.	0.00600	0.00899	0.00836
Max value	9.86962	−0.73000	−0.16858
Min value	9.75222	−0.80004	−0.23655

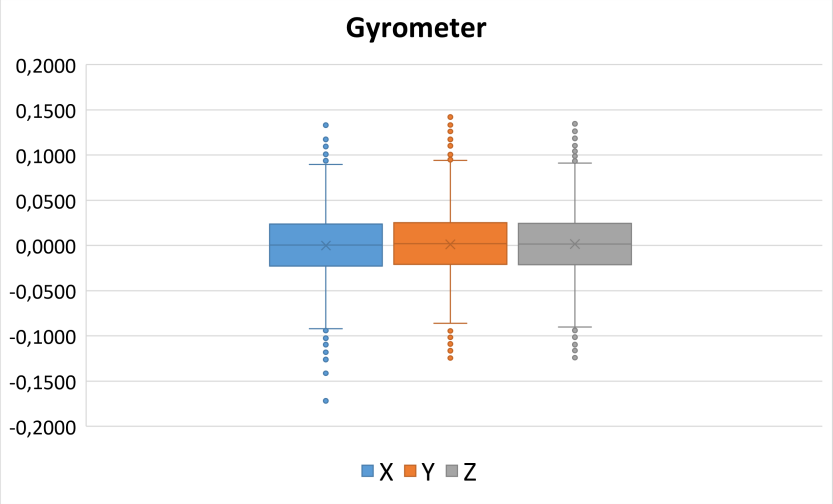


Figure 4.13: Distribution of values for gyrometer.

Table 4.3: Additional data for gyrometer.

	X	Y	Z
Average/mean	−0.00011	0.00129	0.00162
RMSE/Std. Dev.	0.03384	0.03354	0.03268
Max value	0.13319	0.14212	0.13457
Min value	−0.17199	−0.12452	−0.12411

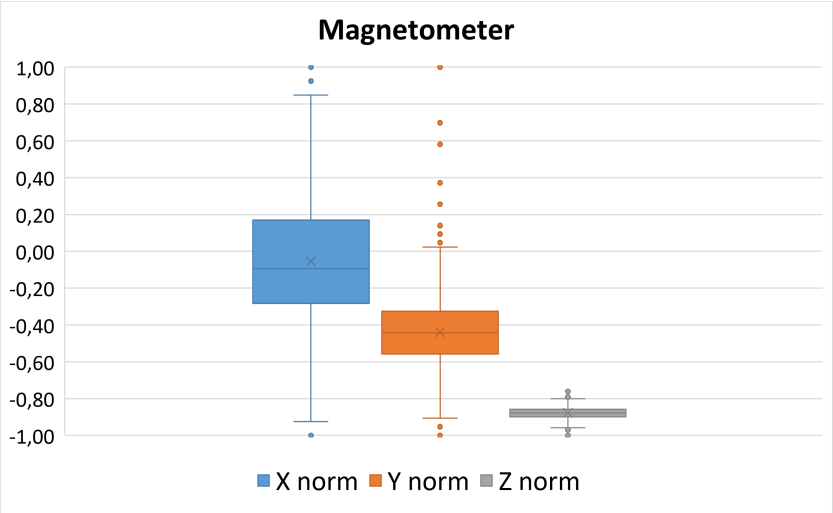


Figure 4.14: Distribution of normalized values for magnetometer.

Table 4.4: Additional data for magnetometer (not normalized).

	X	Y	Z
Average/mean	408.15000	434.40000	−56.70000
RMSE/Std. Dev.	1.19895	1.11052	0.88789
Max value	410.85000	438.45000	−51.90000
Min value	402.90000	425.55000	−58.95000

Figure 4.15: Illustration of the setup, the tripod is of type “Manfrotto Befree GT”.

Flat surfaces

Table 4.5 shows the results from plane-fitting done in CloudCompare, and the data from the evaluation of HoloLens 1 in [17]. The list-ornaments may have slightly corrupted the RMSE value of the left wall.

Table 4.5: Plane-fitting data of flat surfaces.

	Right wall	**Left wall**	**Floor**	**HoloLens 1**
RMSE/Std. Dev.	0.00571	0.00643	0.00959	0.02500
Normal	[−0.22076, 0.00201, 0.97533]	[−0.22678, 0.00450, 0.97394]	[0.00004, 0.99999, 0.00297]	
Center	(2.069, 0.607, −1.447)	(1.271, 0.375, 1.540)	(1.706, −1.510, 0.053)	
No. vertices	820	892	1 718	4 002 (10 planes)

The data in table 4.5 indicate that there is an improvement in accuracy going from HoloLens v1 to v2, and that the standard deviation of $\leq$ 17mm stated in Microsoft documentaion [40], and further backed up by Tölgyessy et al. in [71], is true.

The shortest distance between the right wall center and the left plane is 3.089m, which is 9mm longer than the measured 3.08m. The angle between the wall normals is 0.38°, this gives a maximum distance deviation of $\sim\pm$6mm per meter. The angle between the right wall normal and the floor is 89.72°, which is 0.28° off from the ideal 90°.

Figures 4.16 and 4.17 illustrate how the distances between the mesh and the fitted plane is distributed for the floor and right wall. As can be seen, the meshes have a tendency to "dip" along the edge, i.e. where it is about to transition into another object (e.g. floor to wall). Based on results in [71], the higher RMSE value of the floor is most likely due to the lower reflectivity, and not the angle of scanning.

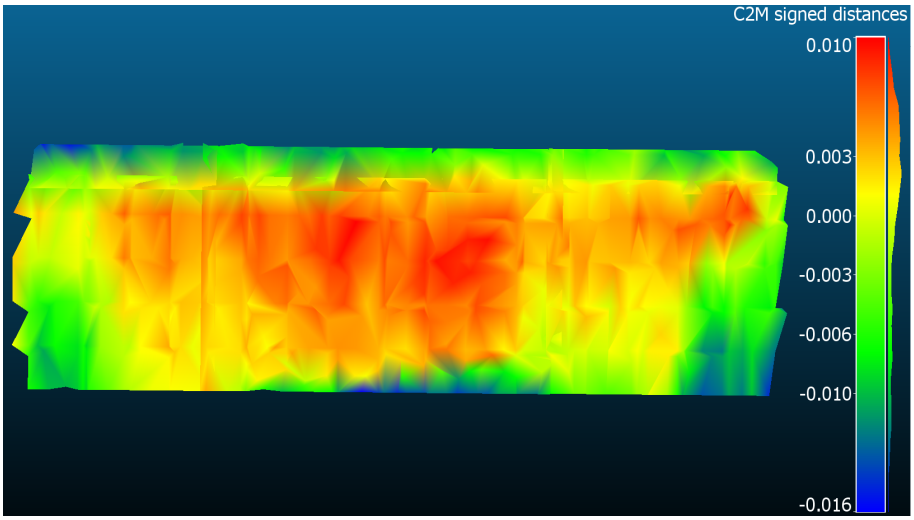


Figure 4.16: Distances between the vertices of the right wall and the fitted plane.

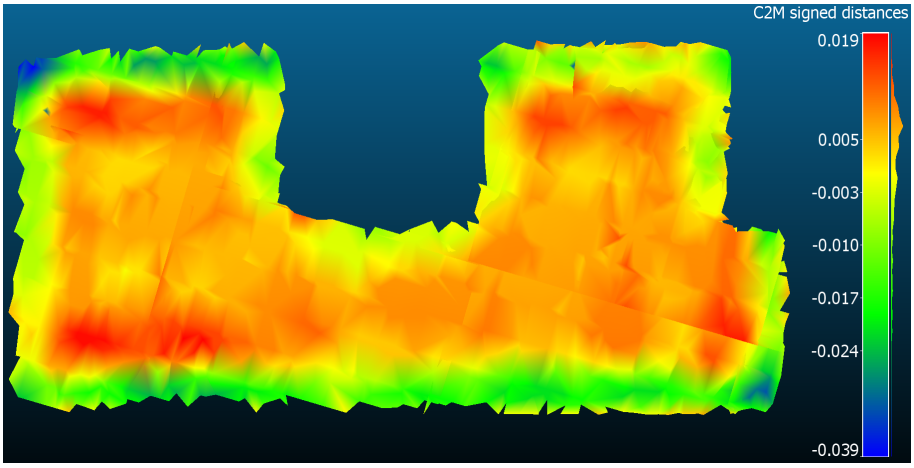


Figure 4.17: Distances between the floor-vertices and the fitted plane.

Sharp transitions

Figures 4.18 and 4.19 indicate that a distance/error of up to 4.5cm can be expected when dealing with large sharp transitions.

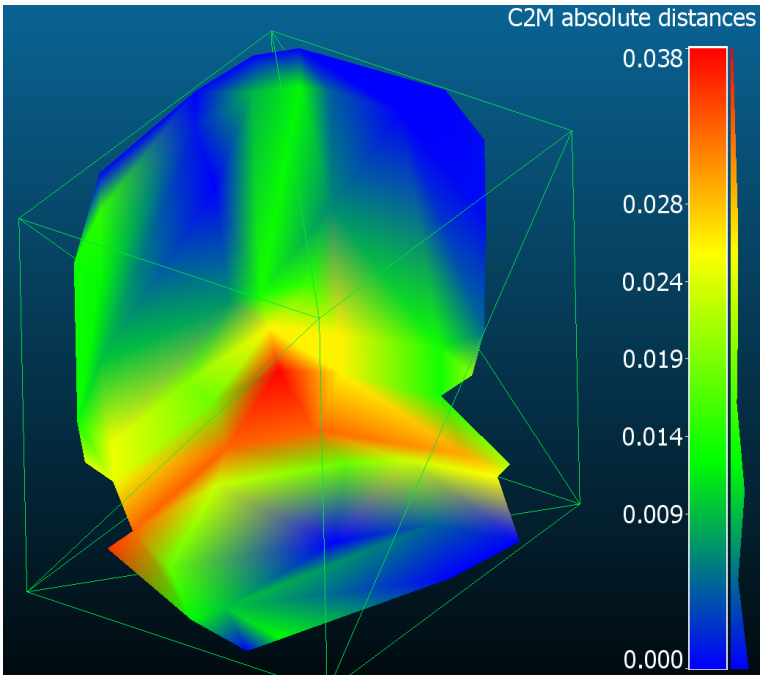


Figure 4.18: Distances from the corner to the bounding box. This is the corner to the left of the table.

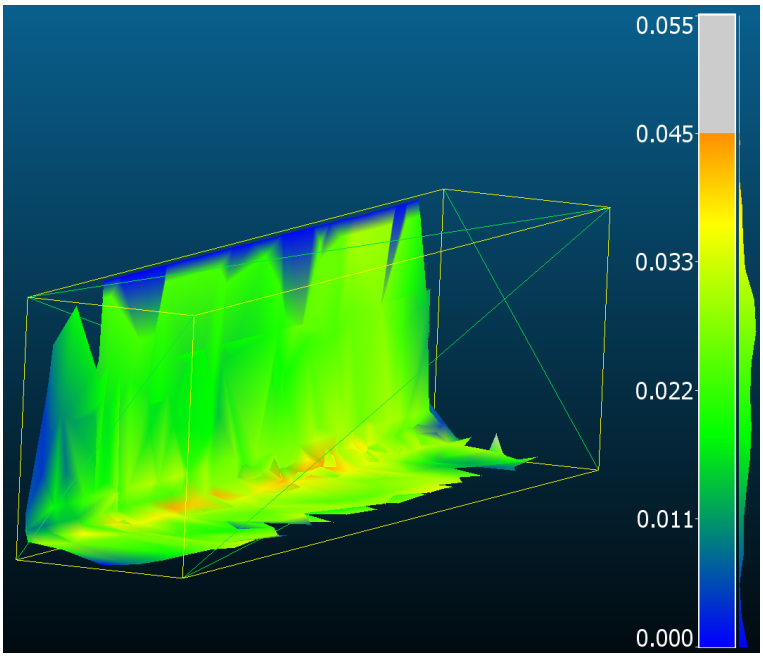


Figure 4.19: Distances from the wall/floor-transition to the bounding box. This is the transition to the right side of the table.

Figure 4.20 shows a distance of at most 8.2cm from the fitted plane to the electric channel, which is 1.2cm more than the measured 7cm. Figure 4.21 shows that the glasses struggle to get a square shape, it starts with a ~45° slope out from the wall, followed by a somewhat flat area, and then a ~45° slope back into the wall.

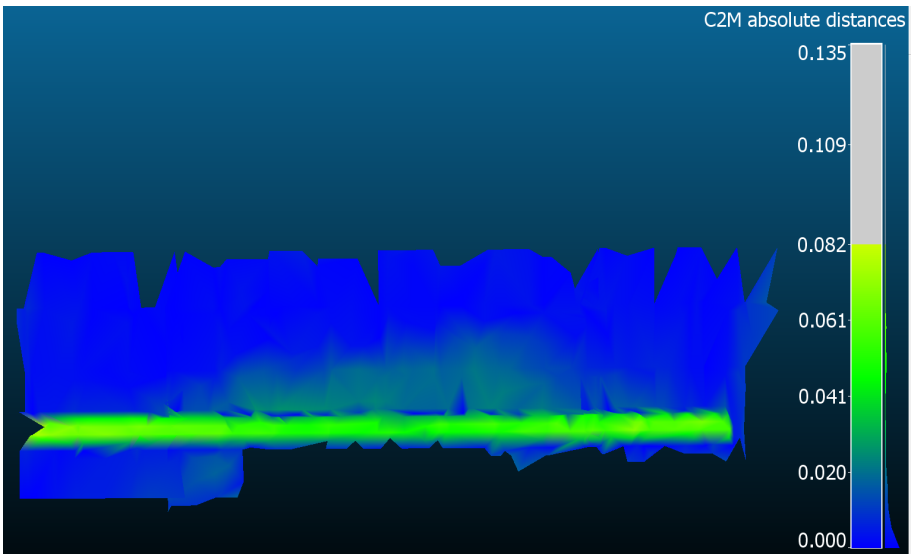


Figure 4.20: Distance between the electric channel and the fitted plane of the right wall.

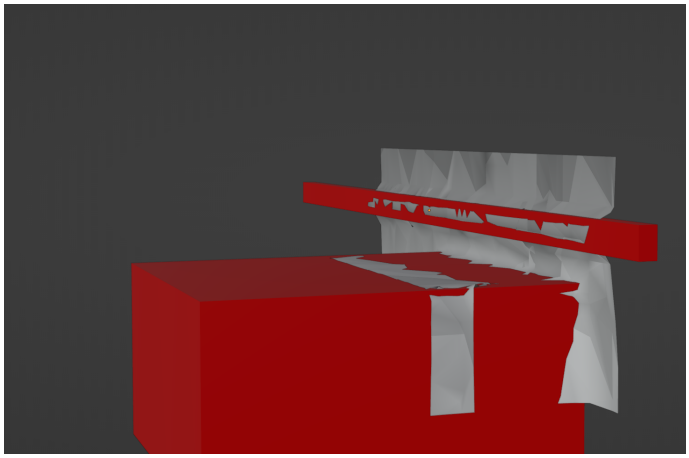


Figure 4.21: How the table and electric channel compares to the reference models in red (only the first half of the table is included). Notice that the "dip" mentioned earlier is also present in the transition between wall and table.

Convex/concave surfaces

Figures 4.22 and 4.23 show that the glasses are able to do a decent scanning of the flower pot, but has a really hard time detecting the bowl. The real volume of the flower pot is calculated to be approximately $0.0261m^3$ and the volume of the scanned one is approximated to $0.0202m^3$, i.e. it is a little undersized. Since the height of the bowl and the upper distance-limit found in the sharp transition experiments are quite similar, the baking bowl was added in retrospect to see if the glasses could handle it better. As figure 4.22 shows, the baking bowl is much more visible but still only about half the height of the real one, and the glasses really struggle with the thickness of the walls (they have almost created a lid on top).

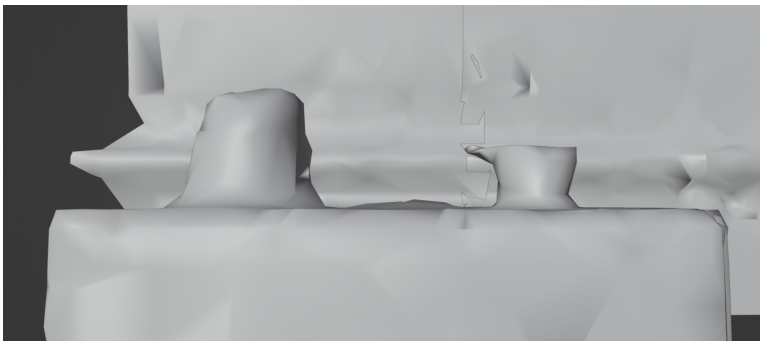


Figure 4.22: The flower pot, baking bowl, and bowl (the small dark gray bump between the pot and baking bowl).

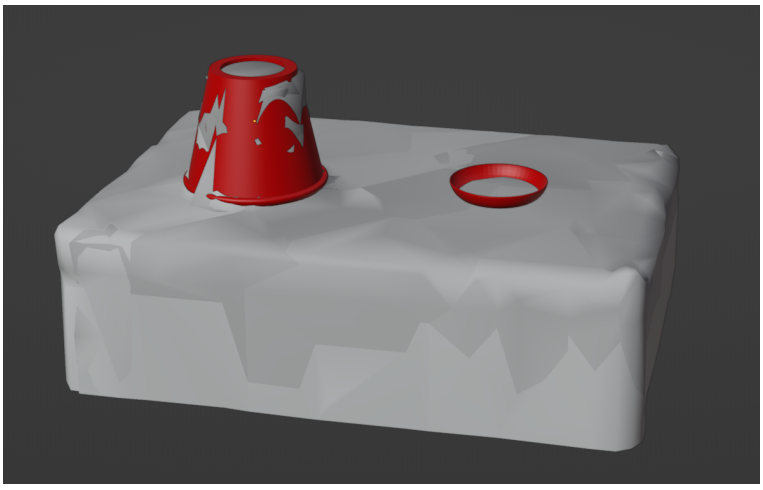


Figure 4.23: How the flower pot and bowl compare to the reference models (before the baking bowl was added).

Triangle coloring

The results from this experiment shows that the glasses scan the room/surfaces in stripes with a top-to-bottom approach. It is the Y-position (height) of the vertices that matters most when it comes to indexing, X- and Z-position comes second.

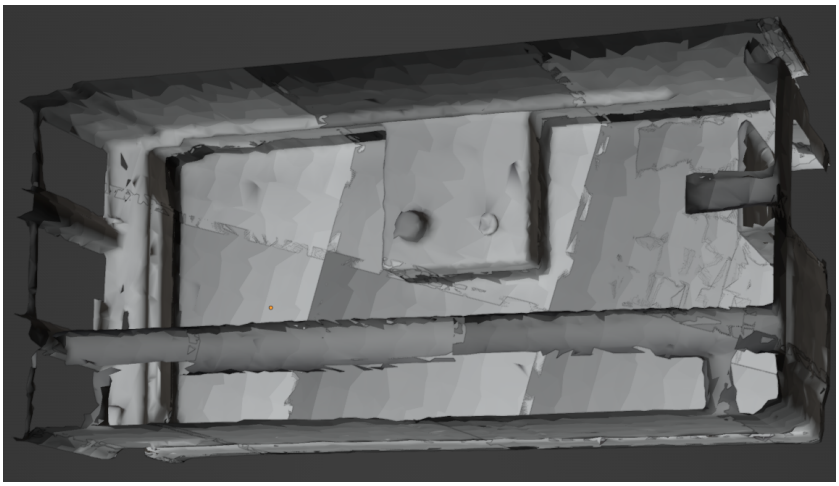


Figure 4.24: Overview.

Figures 4.25–4.27 demonstrates that the Y-position triumphs X and Z.

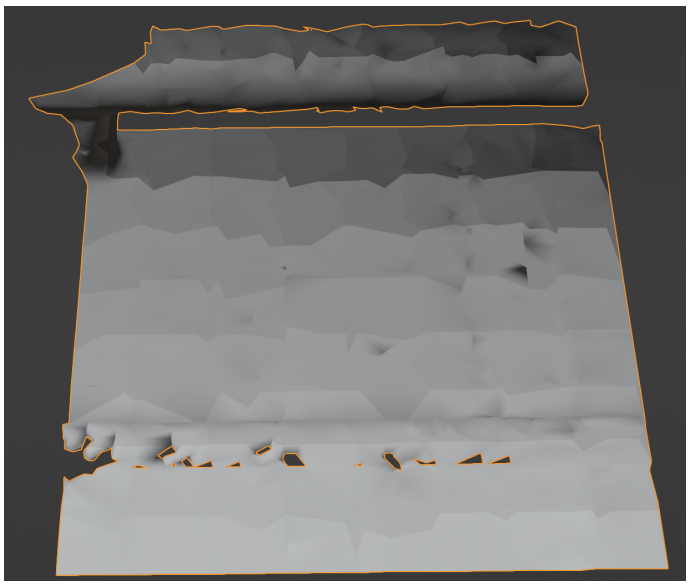


Figure 4.25: The left wall and ventilation-pipe, the top part of the wall and the ventilation are indexed first.

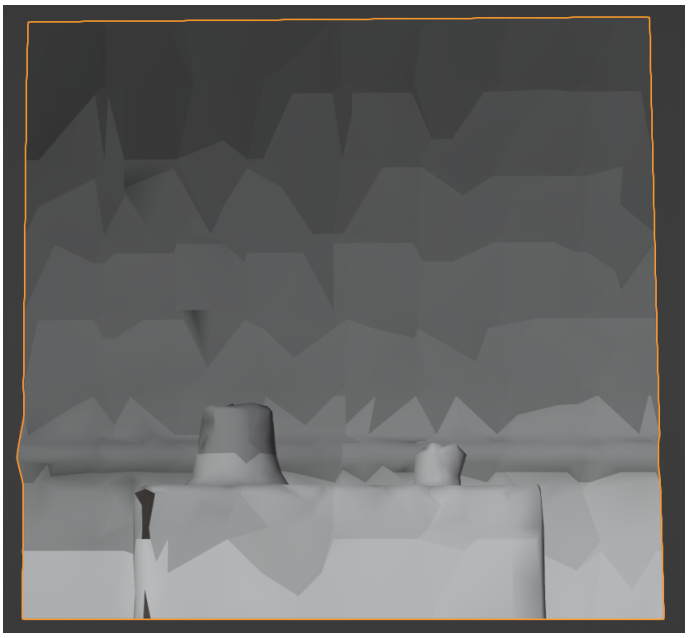


Figure 4.26: The table and right wall seen from the front.

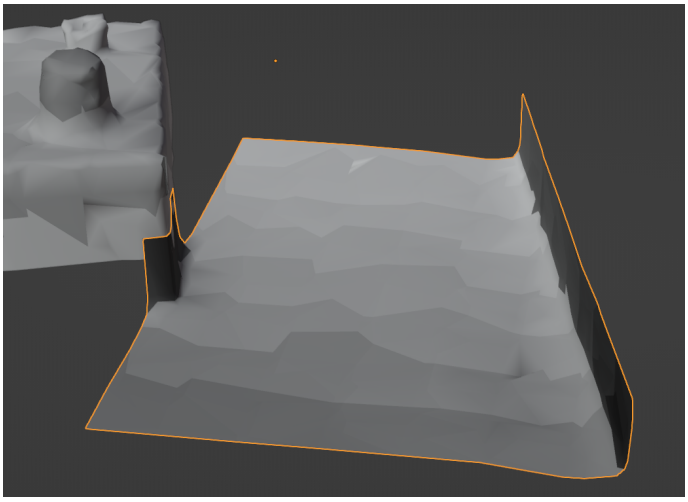


Figure 4.27: Coloring of the floor and part of wall and table.

The next two sections presents useful observations that were not planned to look for in the original experiments.

Surface structure

The general approach is to divide the room into square surfaces, i.e. if you collapse them to a 2D structure they generally become a square plane, like the one in figure 4.30. The dimension of such a plane is quite accurately 2.52×2.62m, and planes overlap each other by $\sim$ 10cm.

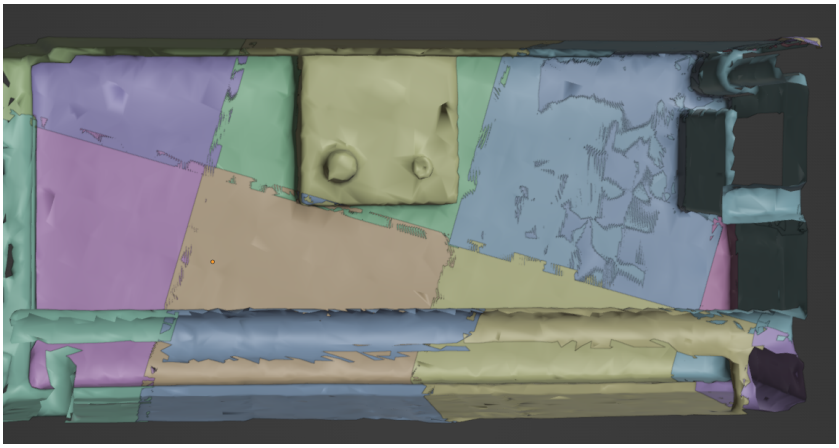


Figure 4.28: Surface structure of entire room.

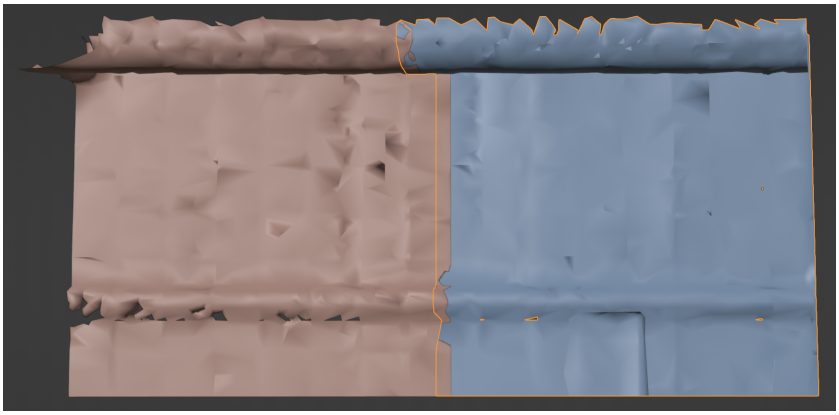


Figure 4.29: Surfaces overlap by $\sim$ 10cm.

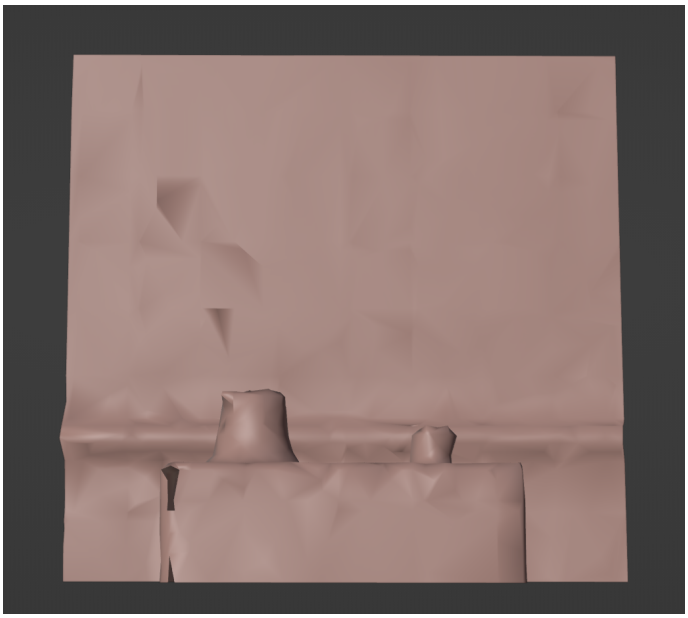


Figure 4.30: Wall and table. The surface becomes a square plane if it is collapsed.

A surface doesn't necessarily have to be one connected mesh, as illustrated in figure 4.31. Here the mesh consists of wall, ventilation pipe, and a small disconnected piece of the table.

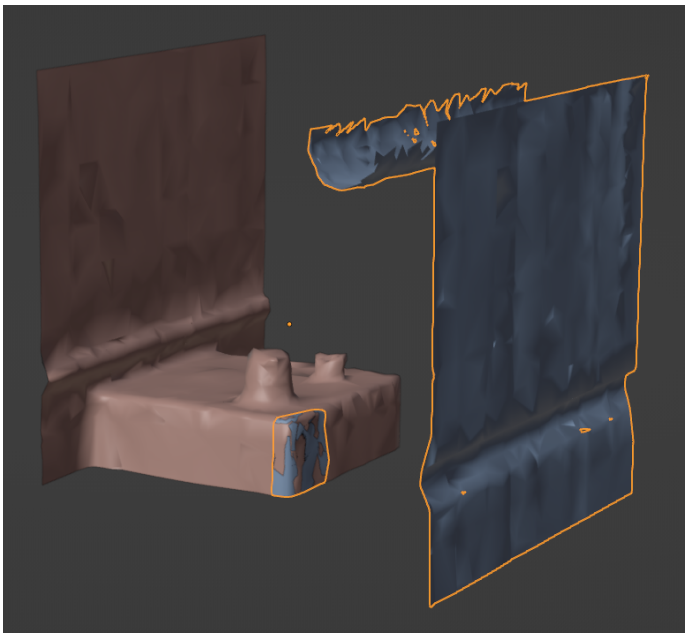


Figure 4.31: Surfaces can be disconnected.

The general substructure of a surface are triangulated quads, where the width and length tends to be a multiple of 32cm. Figure 4.32 shows that one of the square qauds is about 0.32×0.36m, one of the rectangular quads is 0.67×0.32m and the width and length of the entire surface is about 2.62 × 2.52m.

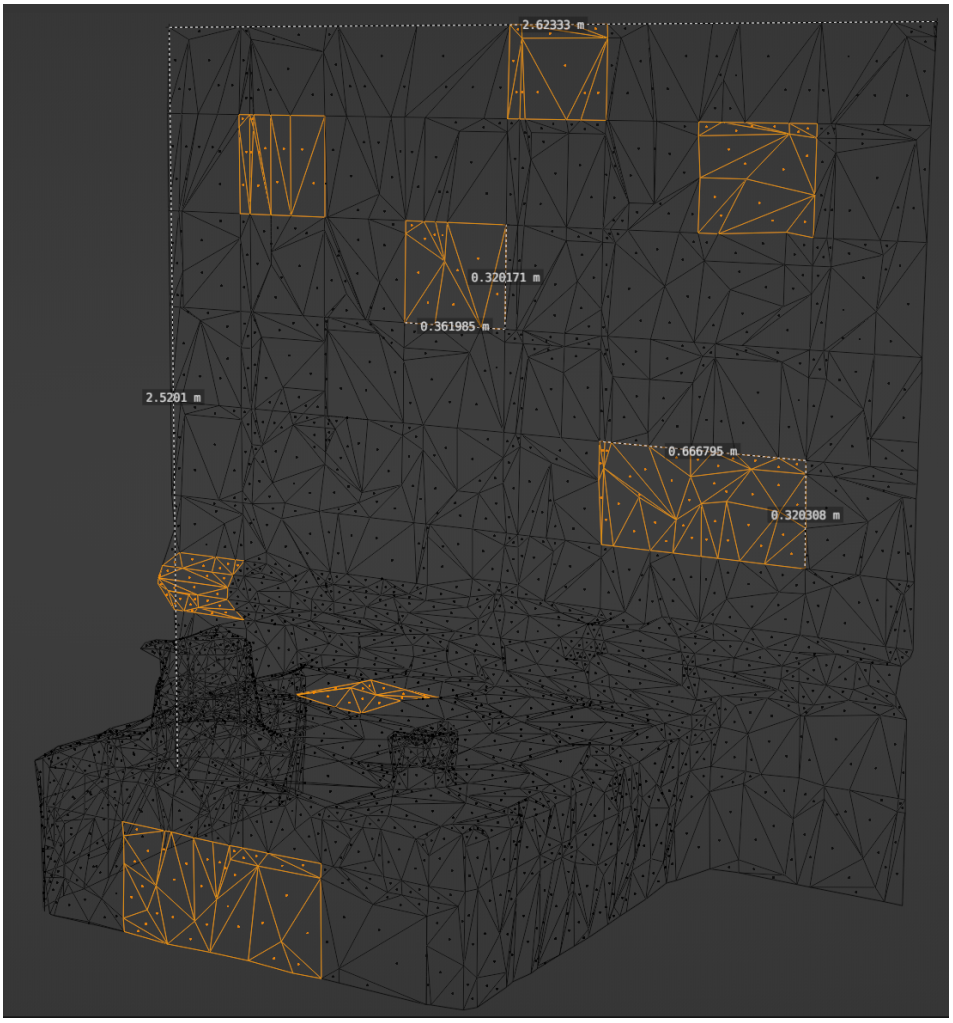


Figure 4.32: Substructure of surfaces and dimensions.

Scanning techniques

The general scanning technique proposed by Microsoft [41] is to move slow and smoothly around the environment. In the case of the flower pot, a faster and less smooth approach yielded better results. If the movement were too slow, the glasses tried to “connect” the top of the pot to the background, which created jagged areas on the flower pot.

4.4.2 LiDAR

Setup and execution

The LiDAR has been rigidly mounted to the same tripod used in figure 4.15 during the whole experiment. The standard configuration of the LiDAR has been used, see figure 4.33 and 4.36. No GPS or IMU has been used in this experiment.

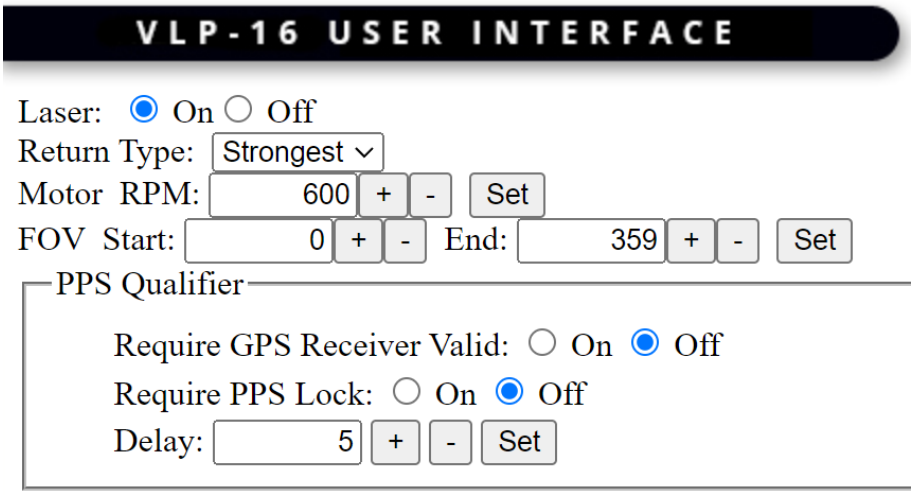


Figure 4.33: LiDAR configuration.

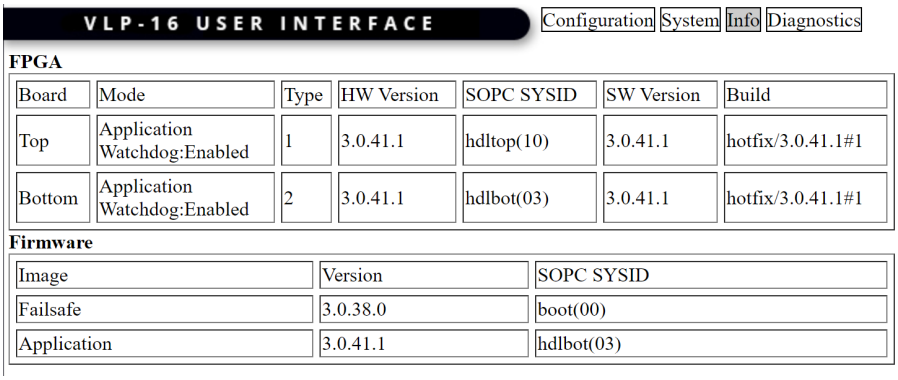


Figure 4.34: Information about field-programmable gate array (FPGA) and firmware.

Since the collected frames[2] are relative to the LiDAR, the SLAM algorithm

2. A frame is the aggregate of all data points acquired during an entire scan (e.g. one full 360° sweep).

provided by LidarView has been used to fit the frames into one global coordinate system. Kitwares tutorial [19] has been followed for installation, SLAM, and exportation. Specifically the pre-built binaries have been used to install the application, the SLAM parameters for indoor scenes have been used for tuning the algorithm, and the procedure under "Directly aggregate all points in a LAS file" was used for exportation.

2 scans and 3 point clouds have been used in this experiment. Scan one is done by walking back and forth in the room to scan as much as possible (figure 4.35), and scan two is done by letting the LiDAR stand still on the table and scan a few frames (figure 4.36). 2 of the point clouds are generated by using SLAM on all frames in each scan, and the third point cloud is generated by exporting only 1 frame from the second scan without using SLAM. The reason for this is to test the accuracy in different scenarios, and to get data unaffected by the SLAM algorithm.

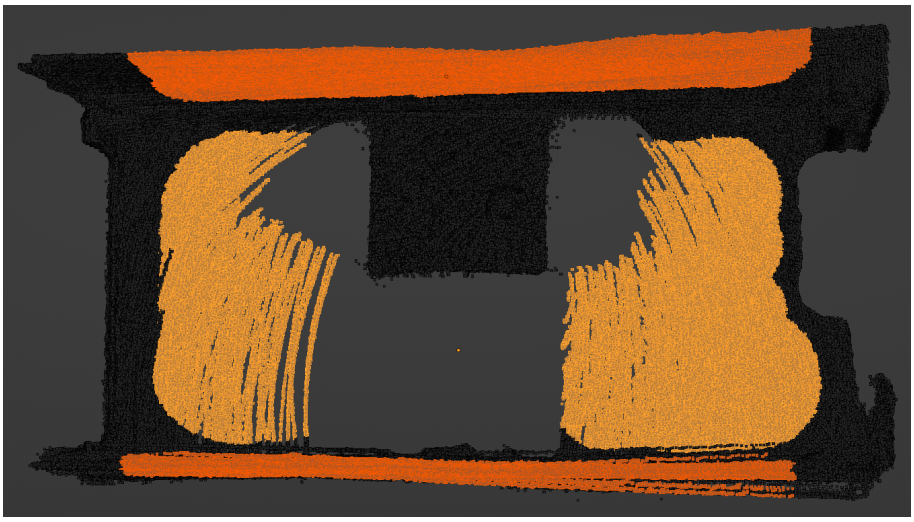


Figure 4.35: The resulting point cloud from scan one (unnecessary points have been removed, e.g. door, locker, and points outside the window). The orange parts represent the parts used in the experiments. The entire point cloud consists of 15 479 458 points and 935 frames.

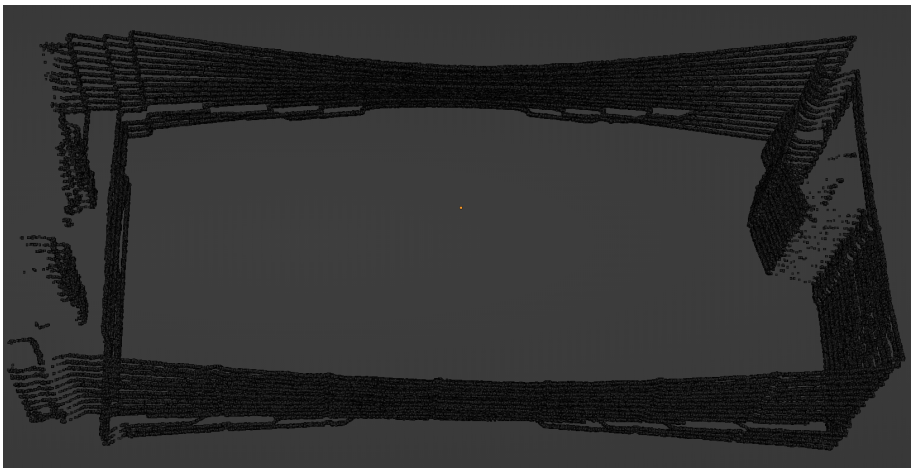


Figure 4.36: The resulting point cloud from scan two, only the walls are used from this scan. The cloud consists of 813 414 points and 29 frames.

Flat surfaces

Tables 4.6–4.7 and figures 4.37–4.39 shows the results from the plane-fitting done in CloudCompare. As can be seen they indicate that the LiDAR has a slightly worse performance overall compared to HoloLens 2. The official accuracy for the LiDAR is ±3cm and a range up to 100m [83], which seems to be correct.

Table 4.6: Plane-fitting data of flat surfaces.

		Right wall	Left wall	Floor
Scan 1, all frames, w SLAM	**RMSE/Std. Dev.**	0.01070	0.01242	0.02969
	Normal	[0.94163, −0.04742, 0.33329]	[0.93992, −0.03319, 0.33978]	[−0.00900, 0.98635, 0.1644]
	Center	(−2.010, 0.523, −0.755)	(0.873, 0.428, 0.368)	(−0.571, −1.578, −0.426)
	No. vertices	4 065 472	4 813 978	101 702
Scan 2, all frames, w SLAM	**RMSE/Std. Dev.**	0.00815	0.00991	
	Normal	[0.99295, 0.00348, −0.11850]	[0.99270, −0.00600, −0.12044]	
	Center	(1.251, 0.292, −0.067)	(−1.836, 0.376, 0.224)	
	No. vertices	246 415	176 789	
Scan 2, one frame, w/o SLAM	**RMSE/Std. Dev.**	0.00780	0.00960	
	Normal	[0.99305, 0.00419, −0.11760]	[0.99276, −0.00472, −0.12004]	
	Center	(1.266, 0.260, 0.087)	(−1.83, 0.369, 0.305)	
	No. vertices	8 534	6 178	

Table 4.7: Angles and distances between the fitted planes.

		Angle normals	Distance planes	Distance deviation/m
Scan 1, all frames, w SLAM	**Floor / Right wall**	90.03° (0.03°)	Not relevant	Not relevant
	Right wall / Left wall	0.90°	3.094m	0.0160 m
Scan 2, all frames, w SLAM	**Right wall / Left wall**	0.55°	3.100m	0.0097 m
Scan 2, one frame, w/o SLAM	**Right wall / Left wall**	0.53°	3.100m	0.0093 m

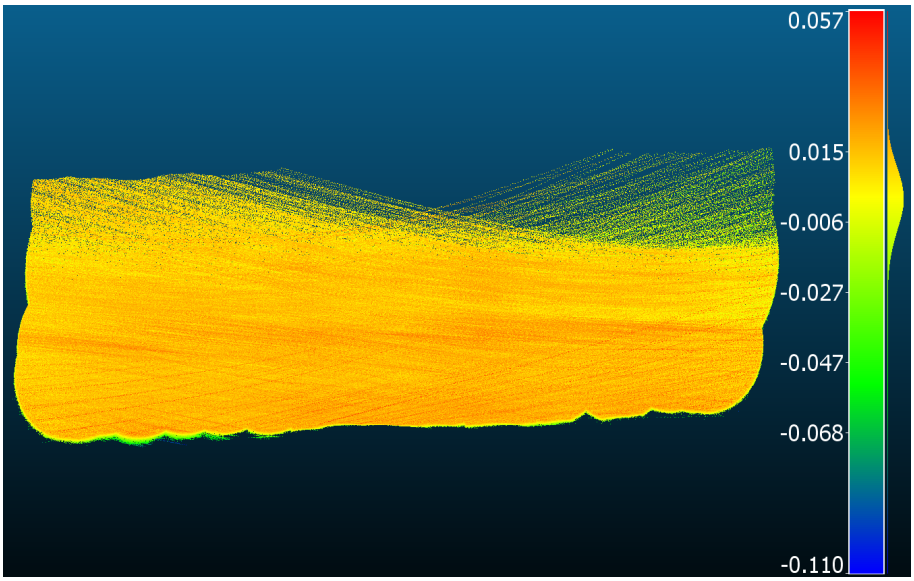


Figure 4.37: Distances from the plane fitting of the right wall in scan one.

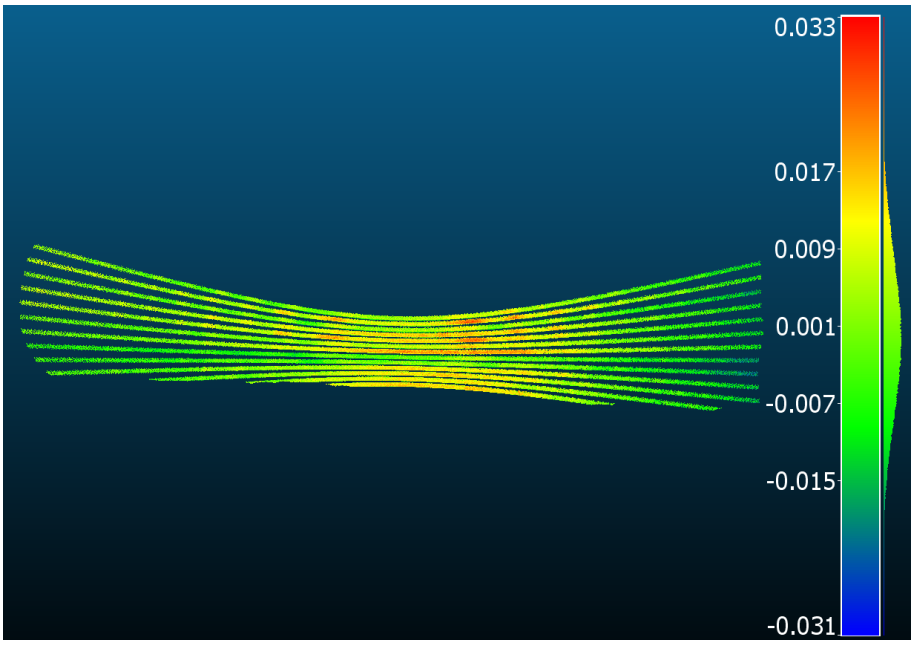


Figure 4.38: Distances from the plane fitting of the right wall in scan two (all frames).

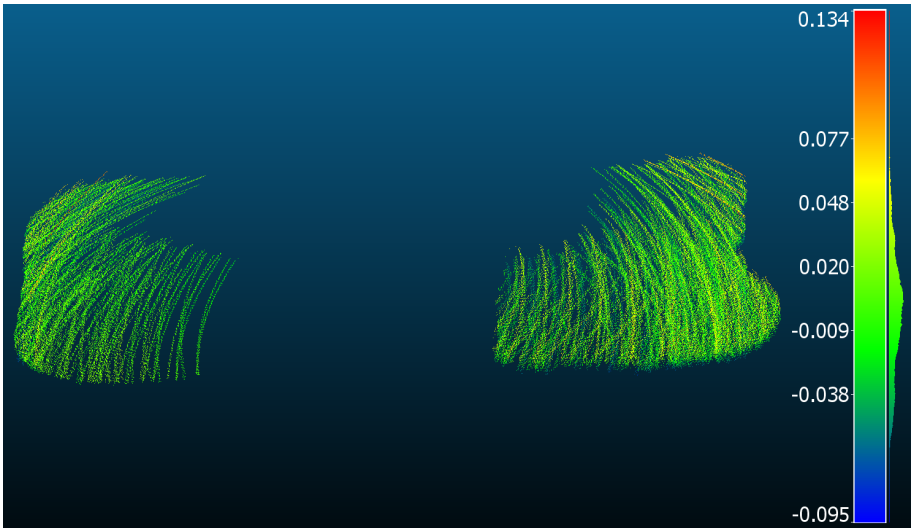


Figure 4.39: Distances from the plane fitting of the floor in scan one.

5

Improvement of mesh

This chapter presents the results and proposed method for improving the mesh. The time frame of the thesis and the available resources onboard the glasses have had major influence.

5.1 Method

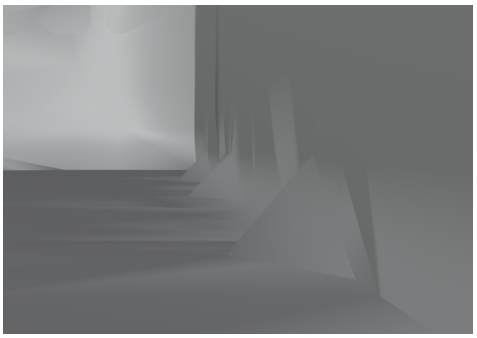
Based on the information from chapter 2 and 4 it is clearly most practical to treat the meshes as a whole in a global coordinate system instead of improving them one by one locally. It is also most practical to do improvements in batches rather than continuously, this way you can do a complete scan, turn off updating and rendering to get the maximum available computing power, and then run an improvement on the whole environment. However, the divided structure of the surfaces should be kept, since if you merge them you would lose the ability to check against updated meshes coming from the glasses' internal memory. In addition, you would have to render the whole environment instead of just the meshes that are within field of view.

Since the generated planes in section 4.4.1 turned out to have a reasonably good global accuracy, the proposed method consists of using these planes to "pull" points that are within a certain distance onto the plane. This will produce completely flat surfaces with sharp 90° transitions, it also means that features on or close to the wall may end up being pulled in as well (e.g. electric channels, paintings, and lists).

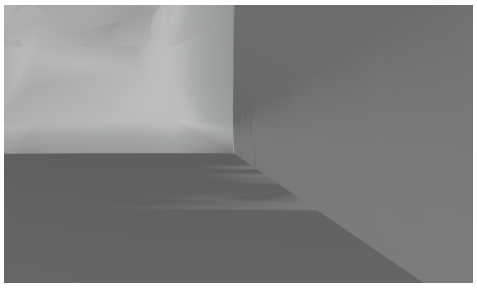
Section 5.2 shows results from a preliminary post-processing done with a Python-script on the same meshes illustrated in section 4.4.1. The algorithm has yet to be implemented and tested on the glasses themselves.

5.2 Results and discussion

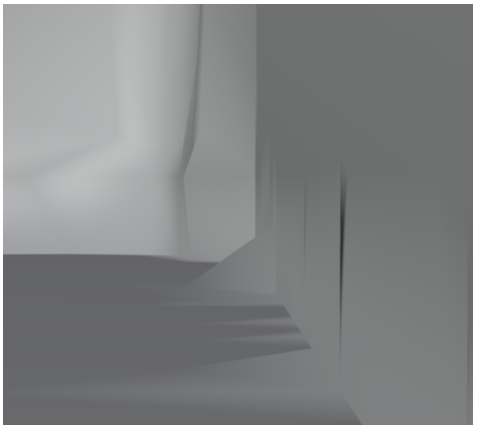
The results in figure 5.1 shows that a threshold of 4–4.5cm from the plane seems to work well (as indicated in chapter 4).



(a) Right side of table with 3.5cm as threshold.



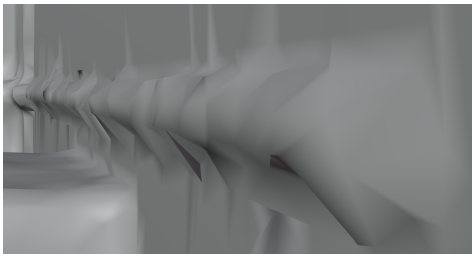
(b) Right side of table with 4cm as threshold.



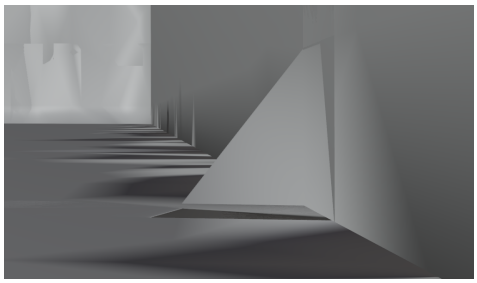
(c) Left side of table with 4cm as threshold. The two pipes is believed to be the cause for the one protruding vertex.

(d) Some parts of the vertical ventilation pipe have merged into the wall (4cm threshold).

Figure 5.1: Results at varying distance thresholds.



(e) Some parts of the electric channel is flattened at 4cm threshold.



(f) Slightly more than 6cm of threshold is required to remove the curvature on the left wall, the need for higher threshold might be because of the list ornaments.

Figure 5.1: Results at varying distance thresholds (Cont.).

The proposed method is simple and very naive, but is well suited for the available computational power on the glasses, and it works fine as long as the real-world surface was flat and without features too small. The results presented in chapter 4 indicates that, in the given circumstances, the glasses are equally, if not slightly better, than the LiDAR. Mind you that the glasses and LiDAR are designed to perform in different scenarios, e.g. the glasses would never be able to scan outdoors up to 100m with ±3cm accuracy. Additionally, if an IMU was used in combination with the LiDAR, and the SLAM algorithm was fine-tuned even more then the results would probably have been better.

6

Conclusion and future work

Although with a slightly different approach than the one suggested in the original task description, this project has successfully addressed both of the research questions, and created a very strong foundation for further development. Chapter 2 presented all the relevant information about HoloLens 2 that could be obtained through official sources. Chapter 4 presented a thorough analysis of the spatial mapping in an effort to extract as much information as possible, the accuracy of the spatial mapping was also compared against a LiDAR. Chapter 5 presented a method for improving the mapping based on the information from chapter 2 and the results from chapter 4.

As a concluding remark, some recommendations and ideas for how to proceed with this project is presented below.

The first recommendation is to convert the current application into a "Holographic Remoting Remote app" so that the full power of a computer can be utilized. This will provide a great platform for implementing more advanced algorithms.

The second recommendation is to implement the proposed method on the glasses. Since the planes/meshes used to generate the results in section 5.2 was manually cut out in Blender, a way to extract the same information in the application needs to be figured out. This could be done by using Scene Understanding, manually checking relations between positions and normals, or with the help of the user (e.g. define/draw bounding boxes) – or maybe a combination of the three. When this is done it should be tested with even higher resolutions than the 8 000 triangles/m^3 used in this project to see if it is of any use.

When it comes to further processing of the mapping, three algorithms/ap-

proaches was found particularly interesting. The first one is by Shen et al. [68] where they present a method of reconstructing sharp features from blended or chamfered features by using normal filtering. It seems like the presented results can be useful for this project, and may be something to consider doing before running the proposed plane fitting algorithm. The second algorithm is Lloyd's algorithm [85], which helps to make the meshes more uniform by evenly distributing the vertices. Whether to use this on all surfaces, or to run it first, between other algorithms, or last, is something that should be tested. The third algorithm/suggestion is to convert the mapping into a blending spline surface since such surfaces are good for difficult shaping and dynamic change of shapes [23]. This could for instance enable the user to interact with the mapping and shape it so it becomes more accurate. Resources/libraries such as [24, 23, 11] is something to consider for implementing this.

Other ideas that should be considered is to switch to Unity so it is easier to create UI and other effects, and maybe create a version that uses the point cloud from "research mode" instead of the mapping provided by the glasses.

Bibliography

[1] Barrett, J. Side Effects of Virtual Environments: A Review of the Literature.

[2] Blender Technologies. Blender. https://www.blender.org/, Visited: 2022-04-28.

[3] CloudCompare. CloudCompare. https://www.danielgm.net/cc/, Visited 2022-04-27.

[4] Curless, B., and Levoy, M. A Volumetric Method for Building Complex Models from Range Images. *SIGGRAPH 3* (09 1996).

[5] Cutress, I., and Terry, E. Hot Chips 31 Live Blogs: Microsoft Hololens 2.0 Silicon. https://www.anandtech.com/show/14775/hot-chips-31-live-blogs-microsoft-hololens-20-silicon, 2019, Visited: 2022-04-20.

[6] Enlyft. Companies using microsoft visual studio. https://enlyft.com/tech/products/microsoft-visual-studio, Visited: 2022-05-15.

[7] Epic Games. Create A HoloLens 2 App With Unreal Engine And Mixed Reality UX Tool | Inside Unreal. https://dev.epicgames.com/community/learning/livestreams/4vR/create-a-hololens-2-app-with-unreal-engine-and-mixed-reality-ux-tools-inside-unreal, Visited: 2022-04-25.

[8] Epic Games. MRMesh. https://docs.unrealengine.com/4.27/en-US/API/Runtime/MRMesh/, Visited: 2022-04-25.

[9] Epic Games. Spatial Mapping. https://dev.epicgames.com/community/learning/courses/qyR/mapping-the-real-world/LBo/spatial-mapping, Visited: 2022-04-25.

[10] Epic Games. Unreal Engine. https://www.unrealengine.com/en-US/, Visited: 2022-04-25.

[11] Floater, M. Meshless Parameterization and B-Spline Surface Approximation.

[12] Gu, W. HoloLens2-ResearchMode-Unity. https://github.com/petergu684/HoloLens2-ResearchMode-Unity, Visited: 2022-04-28.

[13] Hoppe, H., Derose, T., Duchamp, T., Mcdonald, J., and Stuet-zle, W. Surface reconstruction from unorganized point clouds.

[14] Hübner, Patrick et. al. Evaluation of HoloLens Tracking and Depth Sensing for Indoor Mapping Applications. *Sensors 20*, 4 (2020).

[15] Jetbrains. Rider. https://www.jetbrains.com/rider/, Visited: 2022-04-25.

[16] Kah, P., Shrestha, M., Hiltunen, E., and Martikainen, J. Robotic arc welding sensors and programming in industrial applications. *International Journal of Mechanical and Materials Engineering 10* (12 2015).

[17] Khoshelham, K., Tran, H., and Acharya, D. INDOOR MAPPING EYEWEAR: GEOMETRIC EVALUATION OF SPATIAL MAPPING CAPABILITY OF HOLOLENS. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-2/W13* (2019), 805–810.

[18] Kipman, A. HoloLens 2: Unpacked. https://video.ethz.ch/speakers/global-lecture/2019/61d6198e-bf0d-4970-962a-a56e70482fce.html, 2019, Visited: 2022-02-22.

[19] Kitware. How to SLAM with LidarView? https://gitlab.kitware.com/keu-computervision/slam/-/blob/master/paraview_wrapping/Plugin/doc/How_to_SLAM_with_LidarView.md, Visited: 2022-05-10.

[20] Kitware. LidarView: The ParaView Lidar App. https://www.paraview.org/lidarview/, Visited: 2022-05-10.

[21] Klein, G., and Murray, D. Parallel Tracking and Mapping for Small AR Workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality* (2007), pp. 225–234.

[22] Koch, R., Pollefeys, M., and Van Gool, L. Multi Viewpoint Stereo from Uncalibrated Video Sequences. pp. 55–71.

[23] Lakså, A. Blending techniques in Curve and Surface constructions. UiT The Arctic University of Norway, May 2021.

[24] Lakså, A. gmlib. https://source.coderefinery.org/gmlib/gmlib1, Visited: 2022-05-15.

[25] Li, L. Time-of-Flight Camera - An Introduction, 2014.

[26] Liu, Yang et. al. Technical Evaluation of HoloLens for Multimedia: A First Look. *IEEE Multimedia PP* (10 2018), 1–1.

[27] M. L. Heilig. Sensorama Simulator. U. S. Patent 3,050,870, Issued August 28. 1962.

[28] M. L. Heilig. Stereoscopic Television for Individual Use. U. S. Patent 2,955,156, Issued October 4. 1960.

[29] Microsoft. Microsoft Hololens 2 Research Mode for Unreal Engine. https://github.com/microsoft/HoloLens-ResearchMode-Unreal, Visited: 2022-04-25.

[30] Microsoft. Microsoft-OpenXr-Unreal. https://github.com/microsoft/Microsoft-OpenXR-Unreal, Visited: 2022-04-25.

[31] Microsoft. MixedRealityToolkit-Unreal. https://github.com/microsoft/MixedRealityToolkit-Unreal/blob/master/FAQ.md#how-does-mrtk-unreal-relate-to-mrtk-unity, Visited: 2022-04-25.

[32] Microsoft. Universal Windows Platform (UWP) app samples. https://github.com/microsoft/Windows-universal-samples, Visited: 2022-04-27.

[33] Microsoft. HoloLens 2 fundamentals: develop mixed reality applications. https://docs.microsoft.com/en-us/learn/paths/beginner-hololens-2-tutorials/, Visited: 2022-04-28.

[34] Microsoft. Mixed Reality Toolkit Unity. https://github.com/microsoft/MixedRealityToolkit-Unity/releases, Visited: 2022-04-28.

[35] Microsoft. Visual Studio. https://visualstudio.microsoft.com/, Visited: 2022-04-28.

[36] Microsoft. Using Visual Studio to deploy and debug. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-visual-studio?tabs=hl2, Visited: 2022-05-15.

[37] Microsoft. Visual Studio community support. https://docs.microsoft.com/en-us/answers/products/vs, Visited: 2022-05-15.

[38] Microsoft Documentation. *What is mixed reality?*, 2021, Visited: 2022-02-18. https://docs.microsoft.com/en-us/windows/mixed-reality/discover/mixed-reality.

[39] Microsoft Documentation. *About HoloLens 2*, 2021, Visited: 2022-04-20. https://docs.microsoft.com/en-us/hololens/hololens2-hardware#sensors.

[40] Microsoft Documentation. *Azure Kinect DK hardware specifications*, 2021, Visited: 2022-04-20. https://docs.microsoft.com/en-us/azure/kinect-dk/hardware-specification.

[41] Microsoft Documentation. *Map physical spaces with HoloLens*, 2021, Visited: 2022-05-01. https://docs.microsoft.com/en-us/hololens/hololens-spaces#mapping-your-space.

[42] Microsoft Documentation. *HoloLens Research Mode*, 2022, Visited: 2022-04-15. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/research-mode#usage.

[43] Microsoft Documentation. *Coordinate Systems*, 2022, Visited: 2022-04-20. https://docs.microsoft.com/en-us/windows/mixed-reality/design/coordinate-systems.

[44] Microsoft Documentation. *Scene Understanding*, 2022, Visited: 2022-04-20. https://docs.microsoft.com/en-us/windows/mixed-reality/design/scene-understanding.

[45] Microsoft Documentation. *Scene Understanding SDK Overview*, 2022, Visited: 2022-04-20. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/scene-understanding-sdk.

[46] Microsoft Documentation. *Spatial Mapping*, 2022, Visited: 2022-04-20. https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping#mesh-processing.

[47] Microsoft Documentation. Using the Windows Device Portal. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-the-windows-device-portal, 2022, Visited: 2022-04-22.

[48] Microsoft Documentation. *HoloLens environment considerations*, 2022, Visited: 2022-05-01. https://docs.microsoft.com/en-us/hololens/hololens-environment-considerations.

[49] Microsoft Documentation. SpatialSurfaceInfo Class. https://docs.microsoft.com/en-us/uwp/api/Windows.Perception.Spatial.Surfaces.SpatialSurfaceInfo?view=winrt-22000, Visited:

2022-04-22.

[50] Microsoft Documentation. SpatialSurfaceMesh Class. https://docs.microsoft.com/en-us/uwp/api/Windows.Perception.Spatial.Surfaces.SpatialSurfaceMesh?view=winrt-22000, Visited: 2022-04-22.

[51] Microsoft Documentation. SpatialSurfaceObserver Class. https://docs.microsoft.com/en-us/uwp/api/Windows.Perception.Spatial.Surfaces.SpatialSurfaceObserver?view=winrt-22000, Visited: 2022-04-22.

[52] Microsoft Documentation. OpenXR. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/native/openxr, Visited: 2022-04-26.

[53] Microsoft Documentation. Scene Understanding in Unreal. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unreal/unreal-scene-understanding, Visited: 2022-04-26.

[54] Microsoft Documentation. Spatial Mapping in Unreal. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unreal/unreal-spatial-mapping, Visited: 2022-04-26.

[55] Microsoft Documentation. Holographic Remoting Overview. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/native/holographic-remoting-overview, Visited 2022-04-27.

[56] Microsoft Documentation. Holographic Remoting Samples. https://github.com/microsoft/MixedReality-HolographicRemoting-Samples, Visited 2022-04-27.

[57] Microsoft Documentation. What is the Mixed Reality Toolkit. https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05, Visited: 2022-04-27.

[58] Microsoft Documentation. What's a Universal Windows Platform (UWP) app? https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide, Visited: 2022-04-27.

[59] Microsoft Documentation. Setting up your XR configuration. https://docs.microsoft.com/en-us/windows/mixed-reality/

develop/unity/xr-project-setup?tabs=openxr, Visited: 2022-04-28.

[60] Microsoft Documentation. Spatial mapping in Unity. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/spatial-mapping-in-unity?tabs=xr, Visited: 2022-04-28.

[61] Microsoft Documentation. *Spatial Coordinate System*, Visited: 2022-05-02. https://docs.microsoft.com/en-us/uwp/api/windows.perception.spatial.spatialcoordinatesystem.trygettransformto?view=winrt-22000#windows-perception-spatial-spatialcoordinatesystem-trygettransformto(windows-perception-spatial-spatialcoordinatesystem).

[62] Milgram, P., and Kishino, F. A Taxonomy of Mixed Reality Visual Displays. *IEICE Trans. Information Systems vol. E77-D, no. 12* (12 1994), 1321–1329.

[63] Newcombe, Richard A. et al. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality* (2011), pp. 127–136.

[64] Pollefeys, M. MARSS2021 - Prof. Dr. Marc Pollefeys - "HoloLens, Mixed Reality and Spatial Computing". https://www.youtube.com/watch?v=gW-nkyF_s48, 2021, Visited: 2022-02-22.

[65] Pollefeys, M. TUM AI Lecture Series - HoloLens, Mixed Reality and Spatial Computing (Marc Pollefeys). https://www.youtube.com/watch?v=45r5VRHfGrs, 2021, Visited: 2022-02-22.

[66] Qualcomm Technologies. Snapdragon 850 Mobile Compute Platform. https://www.qualcomm.com/products/application/mobile-computing/snapdragon-8-series-mobile-compute-platforms/snapdragon-850-mobile-compute-platform, Visited: 2022-04-21.

[67] Roberts, L. G. *Machine Perception of Three-Dimensional Solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[68] Shen, J.-G., Zhang, S.-y., Chen, Z.-y., Zhang, Y., and Ye, X. Mesh sharpening via normal filtering. *Journal of Zhejiang University - Science A: Applied Physics and Engineering 10* (04 2009), 546–553.

[69] STMicroelectronics. iNEMO 6DoF inertial measurement unit (IMU), for smart phones with OIS / EIS and AR/VR systems. Ultra-low power, high accuracy and stability. https://www.st.com/en/mems-and-

sensors/lsm6dsm.html, Visited: 2022.04.24.

[70] Terry, E. Silicon at the Heart of HoloLens 2. In *2019 IEEE Hot Chips 31 Symposium (HCS)* (2019), pp. 1–26.

[71] Tölgyessy, Michal et al. Evaluation of the Azure Kinect and Its Comparison to Kinect V1 and Kinect V2. *Sensors 21*, 2 (2021).

[72] UiT The Arctic University of Norway. DTE-3604 Applied geometry and special effects. https://en.uit.no/education/courses/course?p_document_id=765883&ar=2022&semester=H, Visited: 2022-05-15.

[73] UiT The Arctic University of Norway. DTE-3605 Virtual reality, graphics and animation - project. https://en.uit.no/education/courses/course?p_document_id=765882&ar=2022&semester=H, Visited: 2022-05-15.

[74] UiT The Arctic University of Norway. DTE-3607 Advanced game and simulator programming. https://en.uit.no/education/courses/course?p_document_id=745558&ar=2022&semester=V, Visited: 2022-05-15.

[75] UiT The Arctic University of Norway. DTE-3609 Virtual reality, graphics and animation - theory. https://en.uit.no/education/courses/course?p_document_id=743948, Visited: 2022-05-15.

[76] UiT The Arctic University of Norway. DTE-3610 Finite element methods, programming. https://en.uit.no/education/courses/course?p_document_id=743947, Visited: 2022-05-15.

[77] Ungureanu, Dorin et al. HoloLens 2 Research Mode as a Tool for Computer Vision Research. *arXiv:2008.11239* (2020).

[78] Ungureanu, Dorin et al. HoloLens 2 Research Mode as a Tool for Computer Vision Research. https://arxiv.org/abs/2008.11239, 2020, Visited: 2022-04-20.

[79] Ungureanu, Dorin et al. HoloLens 2 Research Mode: API Overview. https://github.com/microsoft/HoloLens2ForCV, 2021, Visited: 2022-04-20.

[80] Ungureanu, Dorin et al. Research Mode API. https://github.com/microsoft/HoloLens2ForCV, 2021, Visited: 2022-04-20.

[81] Unity Technologies. Unity. https://unity.com/, Visited: 2022-04-25.

[82] Velodyne. VLP 16 'Puck' Set Up Overview. https://www.youtube.com/watch?v=Pa-q5elS_nE, Visited: 2022-05-15.

[83] Velodyne Lidar Inc. Velodyne LiDAR PUCK VLP-16. https://www.goetting-agv.com/dateien/downloads/63-9229_Rev-H_Puck%20_Datasheet_Web.pdf, 2018, Visited: 2022-05-02.

[84] Wikipedia. Kinect. https://en.wikipedia.org/wiki/Kinect, 2022, Visited: 2022-02-22.

[85] Wikipedia. Lloyd's algorithm. https://en.wikipedia.org/wiki/Lloyd%27s_algorithm, Visited: 2022-05-14.

A

Source code and setup

The source code for spatial mapping can be found here: `https://github.com/Cghost96/Thesis-UWP-Spatial-Mapping`. It is included a ".vsconfig" file in the repository which will help automatically install all the required tools and settings. [36] is a nice guide for how to run an application (and also how to manually set up all the tools).

The most relevant files and folders are listed below, otherwise the code and structure is quite self explanatory.

- SpatialMappingMain.cpp – responsible for handling surface updates and exporting the meshes when the app is about to shut down. The meshes can be found using the Device Portal and navigating to System → File Explorer → LocalAppData → SpatialMapping → LocalState → Meshes

- Common → Settings.h – contains all the relevant settings that can be tuned, such as triangles per cubic meter

- Content → SurfaceMesh.cpp – responsible for extracting and storing the data in the correct way

- Data – this folder contains all the meshes and data extracted from HoloLens

- Python – this folder contains the python scripts used in the project, the one of most interest is probably "Improvement.py" which contains the improvement-algorithm described in chapter 5

The source code for extracting the IMU-data can be found here: `https://github.com/Cghost96/Thesis-UWP-CV`. There are 4 projects in the "Sample" folder, the one used in this project is "SensorVisualization". The files responsible for handling the data are "BasicHologramMain.cpp" found in the "SensorVisualization" folder, and "AccelRenderer.cpp", "GyroRenderer.cpp", and "MagRen-

derer.cpp" found under "SensorVisualization → Content". If the setup-steps for the spatial mapping project have been followed then only some additional settings need to be adjusted for HoloLens, this guide explains how to do it [80].

The files and data related to the LiDAR experiments are attached to this delivery as a zip-folder. See this guide for how to connect with the LiDAR [82] (the video is about VeloView but the steps are identical).

B

Original project description

The project description in its original form is listed over the next 4 pages.

Faculty of Engineering Science and Technology
Department of Computer Science and Computational Engineering
UiT - The Arctic University of Norway

Augmented reality.
Spatial mesh approximation using Hololens 2

Casper Andrè Levoll-Steen

Thesis for Master of Science in Technology / Sivilingeniør

Problem description

The objective of this master thesis project is to explore limitations of the mixed reality device Hololens 2 [1] and propose a new method of the spatial mesh processing.

One of the basic features of the Hololens 2 is the ability of virtual objects to interact with the real world. To obtain this interaction, there exists a technology called spatial mapping [2]. It makes a coarse triangulation of its surroundings, for example, the room space and the furniture in the room. The goal of the proposed topic is to process this triangulation so that it approximates real-world objects more precisely and smoothly. The task consists of two main challenges: hardware issues and algorithm development.

Hardware part:
- Identify possible hardware limitations.
- Extract mesh values and parameters.
- Insert the processed mesh back into the device.

Development part:
Use the blending spline approximation techniques [3] to process the triangulated spatial mesh and represent it as a continuous surface.

The task can be extended towards experimental user interface for shape modeling with Hololens 2 with special focus on volume and surface modeling.

Tools:
- Hololens 2
- Unreal Engine (https://www.unrealengine.com)
- MRTK (Mixed Reality Toolkit) (https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unreal/unreal-mrtk-introduction)

References:
1. Hololens 2: https://www.microsoft.com/en-us/hololens/hardware
2. Spatial mapping: https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping
3. Tatiana Kravetc, Børre Bang, Rune Dalmo. Regression analysis using a blending type spline construction. Springer Publishing Company 2017 (10521) ISBN 978-3-319-67885-6. ISSN 0302-9743.s 145 - 161.s doi: 10.1007/978-3-319-67885-6_8.

Dates

Date of distributing the task:	<10.01.2022>
Date for submission (deadline):	<16.05.2022>

Contact information

Candidate	Casper Andrè Levoll-Steen
Supervisor at UiT-IVT, IDBI	Tatiana Kravetc (tatiana.kravetc@uit.no)
Co-supervisor at UiT-IVT, IDBI	Tanita Fossli Brustad (tanita.f.brustad@uit.no) Aleksander Pedersen (aleksander.pedersen@uit.no)

General information

This master thesis should include:

- Preliminary work/literature study related to actual topic
  - A state-of-the-art investigation
  - An analysis of requirement specifications, definitions, design requirements, given standards or norms, guidelines and practical experience etc.
  - Description concerning limitations and size of the task/project
  - Estimated time schedule for the project/ thesis
- Selection & investigation of actual materials
- Development (creating a model or model concept)
- Experimental work (planned in the preliminary work/literature study part)
- Suggestion for future work/development

Preliminary work/literature study

After the task description has been distributed to the candidate a preliminary study should be completed within 3 weeks. It should include bullet points 1 and 2 in "The work shall include", and a plan of the progress. The preliminary study may be submitted as a separate report or "natural" incorporated in the main thesis report. A plan of progress and a deviation report (gap report) can be added as an appendix to the thesis.

In any case the preliminary study report/part must be accepted by the supervisor before the student can continue with the rest of the master thesis. In the evaluation of this thesis, emphasis will be placed on the thorough documentation of the work performed.

Reporting requirements

The thesis should be submitted as a research report and could include the following parts; Abstract, Introduction, Material & Methods, Results & Discussion, Conclusions, Acknowledgements, Bibliography, References and Appendices. Choices should be well documented with evidence, references, or logical arguments.

The candidate should in this thesis strive to make the report survey-able, testable, accessible, well written, and documented.

Materials which are developed during the project (thesis) such as software / source code or physical equipment are considered to be a part of this paper (thesis). Documentation for correct use of such information should be added, as far as possible, to this paper (thesis).

The text for this task should be added as an appendix to the report (thesis).

General project requirements

If the tasks or the problems are performed in close cooperation with an external company, the candidate should follow the guidelines or other directives given by the management of the company.

The candidate does not have the authority to enter or access external companies' information system, production equipment or likewise. If such should be necessary for solving the task in a satisfactory way a detailed permission should be given by the management in the company before any action are made.

Any travel cost, printing and phone cost must be covered by the candidate themselves, if and only if, this is not covered by an agreement between the candidate and the management in the enterprises.

If the candidate enters some unexpected problems or challenges during the work with the tasks and these will cause changes to the work plan, it should be addressed to the supervisor at the UiT or the person which is responsible, without any delay in time.

Submission requirements

This thesis should result in a final report with an electronic copy of the report including appendices and necessary software, source code, simulations and calculations. The final report with its appendices will be the basis for the evaluation and grading of the thesis. The report with all materials should be delivered according to the current faculty regulation. If there is an external company that needs a copy of the thesis, the candidate must arrange this. A standard front page, which can be found on the UiT internet site, should be used. Otherwise, refer to the "General guidelines for thesis" and the subject description for master thesis.

The supervisor(s) should receive a copy of the thesis prior to submission of the final report. The final report with its appendices should be submitted no later than the decided final date.

“Remember kids, the only difference between screwing around and science, is writing it down.”
–Adam Savage