Tampere University

Lauri Varjo

VIRTUAL REALITY WALKTHROUGH OF LASER SCANNED ENVIRONMENTS

Faculty of Information Technology
and Communication Sciences
Bachelor’s thesis
May 2019

ABSTRACT

Lauri Varjo: Virtual reality walkthrough of laser scanned environments
Bachelor's thesis, 23 pages, 3 appendix pages
Tampere University
Bachelor's Degree Programme in Electrical Engineering
May 2019

Virtual reality has been continuously growing for a long time. The applications of virtual reality can range anywhere from industrial uses to entertainment, simulation and training.

There are numerous applications which would benefit from replicating an environment from the real world in the virtual reality. This has become possible with the development of surface reconstruction methods, laser scanning and virtual reality software and hardware. The required equipment has also become more affordable, even for personal use. There are many different methods and aspects to every part of the process.

This thesis explores the theory and practice of different steps of the whole pipeline from a real-world environment into virtual reality. The pipeline consists of scanning the environment, surface reconstruction from the scans, and the development of the virtual reality application. Existing methods, software and tools are used to achieve a functional virtual reality application, with an environment replicated from the real world. Different aspects of the process are discussed, and the results are analysed.

Keywords: laser scanning, surface reconstruction, virtual reality, Unity, walkthrough, mesh, point cloud, 3D

PREFACE

This thesis marks the closure of one chapter in my studies and opens the doors to new fields of curiosity in the academic domain of signal processing. Virtual reality has been a topic of interest for me for a long time, so it was a blessing to get to write a thesis regarding it.

I would like to give a huge thank you to my supervisor Jani Mäkinen for providing me with a motivational and interesting topic, and for giving me all the help, advise and time I needed for this thesis to become complete. Also, I want to express my gratitude and apologies to all the people who had to listen to me talk fanatically about the topic of my thesis for too long.

Tampere, 1 May 2019

Lauri Varjo

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

CloudCompare	a point cloud processing software, with capabilities for handling meshes
FARO SCENE	scan data processing and registration software
MeshLab	a mesh processing software
SteamVR	software development kit for VR
Unity	game engine, which aids in creating virtual reality applications
3D	three-dimensional
API	application programming interface
ASCII	American Standard Code for Information Interchange
CIVIT	Centre for Immersive Visual Technologies
GPS	global positioning system
HMD	head-mounted display
Lidar	light detection and ranging
PLY	Polygon File Format
RGB	red, green, blue
SDK	software development kit
VR	Virtual reality
VRTK	Virtual Reality Toolkit
c	speed of light
d	distance
τ	roundtrip propagation time

.

1. INTRODUCTION

Virtual reality has been continuously growing for a long time. Currently it is being utilized in many industries, as well as in entertainment. The list of practical applications is nearly endless, but it could include for example training and simulation (military, industry, medical), visualising (architecture, science, construction) and planning [20].

In many practices, it would be beneficial to replicate an environment from the real world and explore or utilize it in depth in the virtual reality. This has become possible with laser scanners and computers with competent hardware becoming more common and affordable [2]. Surface reconstruction methods and virtual reality hardware and software have also evolved over the years. The process requires a laser scanning device, a modern computer and means for displaying the end result.

Many methods have already been implemented for achieving different parts of the pipeline. This thesis aims to examine the whole process. Existing methods, tools and software are used to obtain an accurate and functional result. Different steps of the process are examined in a practical way, and the methods are analysed. The focus of this work is to provide clear solutions to different parts of the process, with little manual user input.

This thesis is structured as follows. In chapter 2 the theory behind the methods and the devices used for the process is explained. Chapter 3 examines the workflow of the pipeline and explains the choices made during the process. In chapter 4 the data and the results are analysed. Chapter 5 concludes the thesis by reviewing the most important observations and examining how the set goals were met.

2. THEORY

The pipeline includes applications of laser scanning, 3D (three-dimensional) data processing and virtual reality. The principles of the used technologies are presented here, at least in the depth necessary for the process.

2.1 3D data representations

All the points in the real world can be presented with three real numbers, usually *x*, *y* and *z* [11]. *X* and *y* are often used represent the two horizontal dimensions, while *z* represents the vertical dimension. The position of a point in space can be specified by the values of *x*, *y* and *z* with respect to some coordinate system. 3D data can be used to represent shapes, surfaces and colours of objects or environments.

2.1.1 Point cloud

One way to represent 3D data is a point cloud. It is a collection of data, which contains any number of data points in a space. Each individual point holds the data of its spatial location, represented with x, y and z. A data point can also hold additional scalar values such as intensity, colours or temperature. Some of the common point cloud formats are PLY, OBJ, PTX, XYZ, PCG and E57.

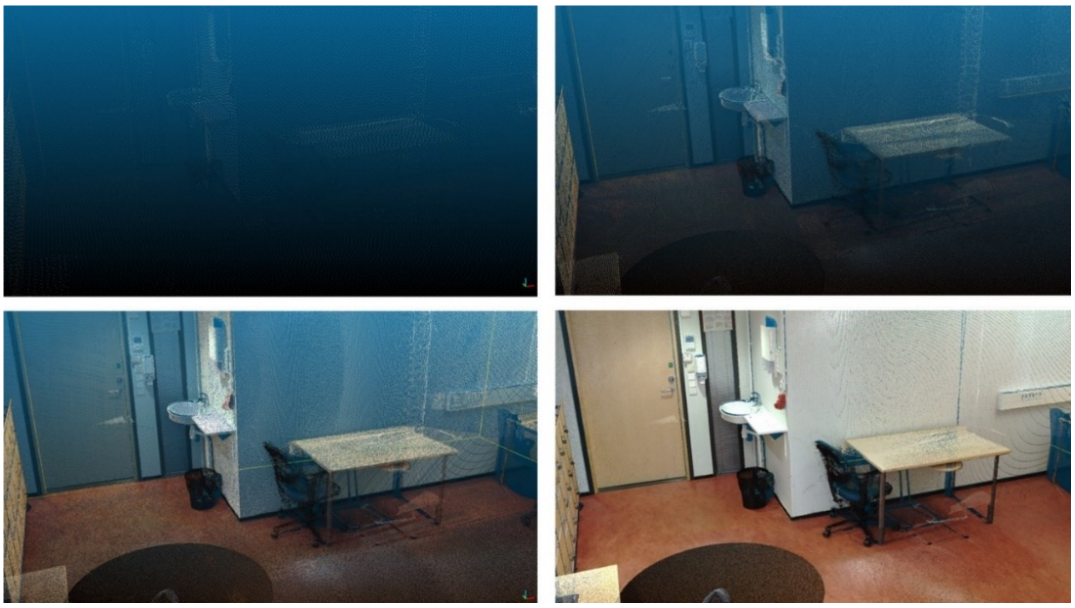


Figure 1. *Point clouds with different amount of points*

As demonstrated in Figure 1, point clouds can be recognized as shapes and objects if they have a sufficient amount of data points. Because of the sheer amount of data and the lack of association between the data points, large point clouds require considerable amounts of processing power. The bottom-right point cloud in Figure 1 has 42 million points with 3 values for spatial coordinates and 3 values for RGB (red, green, blue) colours.

2.1.2 Mesh

Another way for representing 3D data is a polygonal or triangular mesh. The data consists of vertices, edges and faces. Most 3D applications use meshes as the basis for their models because of their improved functionality over point cloud data. Some of the common mesh formats are STL, OBJ, FBX, COLLADA and 3DS.

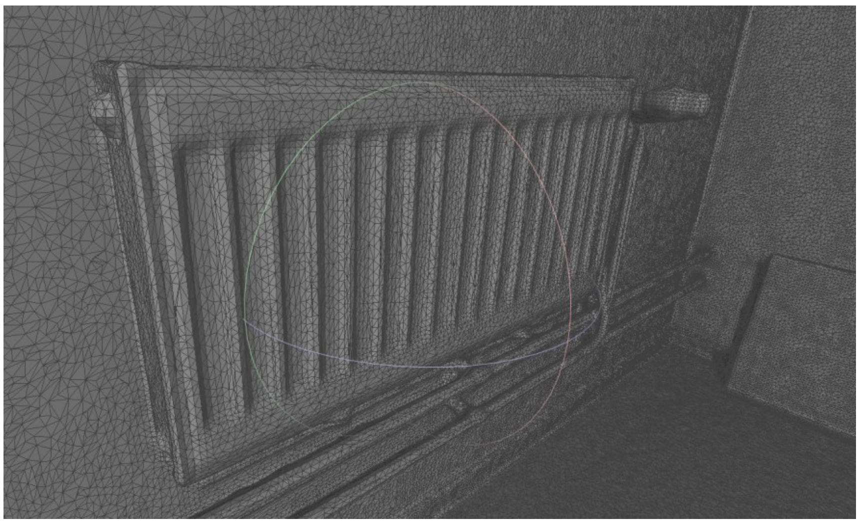


Figure 2. *Triangular mesh*

Figure 2 is a triangular mesh of a radiator and the background wall. As seen in the figure, it consists of many triangle shaped faces with different shades. The corners of the triangles are vertices. The vertices and the faces are connected to each other in the data, and together they form the mesh representation. It is possible to add texture to the faces from either internal or external data.

2.2 Laser scanning

In laser scanning, laser beams are emitted and deflected from surfaces. It is a fast, reliable and safe way to gather 3D information from the surrounding environment in coordinate measurement systems.

2.2.1 Lidar

In lidar (light detection and ranging) systems, light pulses of high intensity are emitted from a laser and are then deflected back from the target to sensitive optical detectors. The time the light pulse takes for the round trip propagation is measured with a precise time interval unit. [10] Knowing the time it takes from the light pulse to travel the round trip and the angle from which it was emitted, it is possible to calculate accurate coordinates for the target point along with the intensity of the returning signal [14].

The lidar configuration used in ranging and altimeter systems is usually monostatic (Figure 3). The computer controls the whole process, which begins by emitting a light pulse with the laser. A small amount of the light is split apart and directed to an optical detector which starts the precision clock, while the rest of the beam is projected at the target. The reflected beam from the target is collected by the primary mirrors and focused to the secondary mirror. The collected reflection beam is focused into a collection of lenses and filters that remove interference and noise from the signal. An optical photodetector detects the laser pulse and stops the precision clock. The elapsed time is sent to the computer, which calculates the distance to the target. [10]

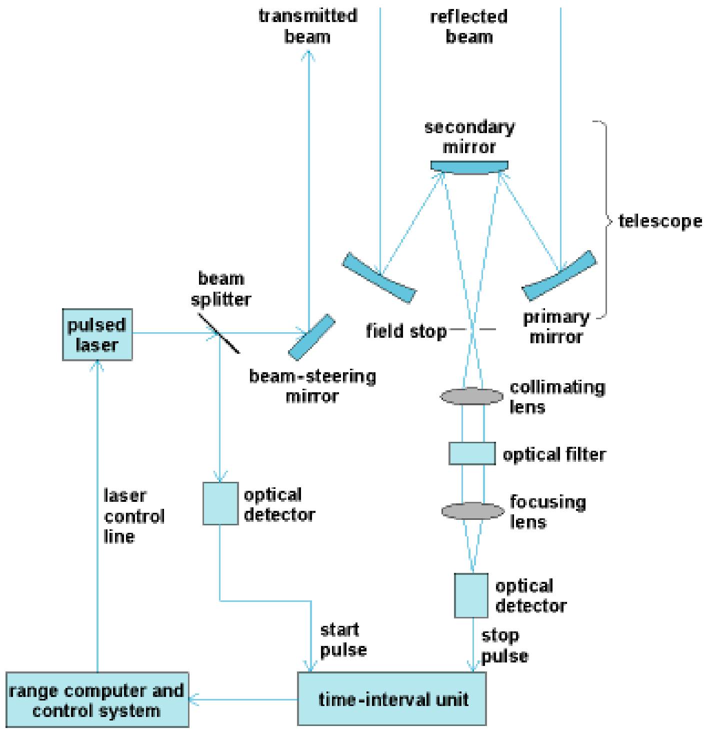


Figure 3. *A simple demonstration of a monostatic system [10]*

The equation for measuring the distance d of the target from the scanner is,

$$d = \frac{c\tau}{2}, \tag{1}$$

where the c is the speed of light and τ is the time it takes for the light pulse to propagate to the surface and back [10].

2.2.2 Field of view

Most laser scanners at the present are made for dome scanning [14]. There are also panoramic and cone-shaped scanners, which work well for different purposes. For example, 3D hand scanners can have a cone-shaped field of view, which is suitable for scanning individual objects.

When scanning a space, especially a closed one, a dome-shaped field of view is preferred (Figure 4). The dome scan can accurately measure the surrounding environment, including the ceiling, but is unable to scan the area below it due to technical limitations. However, this flaw can be easily negated by performing multiple scans.

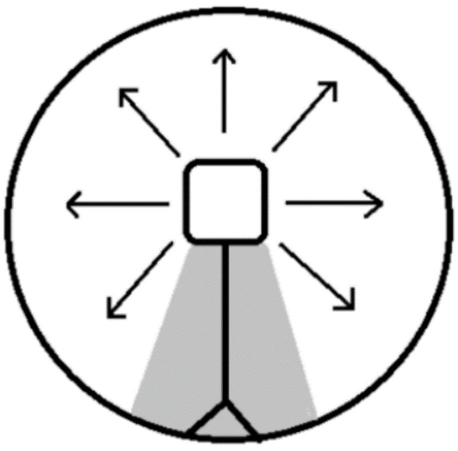


Figure 4. *Dome scan*

Dome scanning is enabled by having a mirror on the scanner. The mirror rotates rapidly along its own axis, employing laser beams and collecting data from the environment. The scanning device also rotates along the horizontal axis to acquire a full 3D dome scan.

2.2.3 Acquired data

Laser scanning an environment generates a point cloud containing spatial information from the surrounding space. Each data point contains coordinates calculated by the scanner from the distance and angle, with regards to the base point, which is usually the scanner. Colours can also be included in the point cloud by the scanner taking pictures of the scanned area and mapping the colours from the images to the data points.

If multiple scans are performed, it is beneficial for the scanner to also have GPS (Global Positioning System) to acquire the location of the scanner in each individual scan. Knowing the locations of each of the multiple scans makes it trivial to merge the acquired point clouds into a single one.

2.3 Surface reconstruction

The process of reconstructing a solid surface from a set of individual data points is called surface reconstruction. Many algorithms and methods have been developed for the conversion of point cloud data to a triangular or polygonal mesh. One of those is the Poisson surface reconstruction algorithm [15]. The same researchers also developed an improved version of the algorithm called Screened Poisson surface reconstruction [16].

Poisson surface reconstruction is a widely used, reliable and stable method for surface reconstruction [1]. It is highly resilient to data noise, which makes it a good choice for processing 3D scans from the real world. It is based on casting the surface reconstruction from a set of oriented points as a spatial Poisson problem [15].

The algorithm works by computing a 3D indicator function, that defines the points inside the model as 1 and 0 at the outside. The reconstructed surface is obtained by extracting an appropriate isosurface from the indicator function. The input data is assumed to consist of points that have inward-facing normals and lie on or near the surface of the model. [15]

2.4 Virtual reality

Virtual reality (VR) is a computer simulation that strives to produce an image of the real world that is as authentic as possible. It produces synthetic stimuli to our senses and responds accordingly to our inputs, resulting in the feeling of being immersed in the simulation. Through virtual reality, humans can share ideas and experiences. [20] Normally a VR (virtual reality) system utilizes visual and aural stimuli, but occasionally other senses can also be stimulated.

Usually a head-mounted display (HMD) is used to provide the images for the vision and the sounds for the hearing. The device features two screens, one for each eye, to produce a stereoscopic 3D view for the user. Binaural audio is used with head-related transfer functions to provide a stereoscopic sound system. The point of view and the distance to virtual audio sources change when the user moves their body or rotates their head. To follow the actions of the user, the location and orientation of the HMD are tracked by tracking cameras. The tracking system enables refreshing the images and sound according to the position of the user and their head.

VR applications are usually created and executed with the help of a game engine. The engine handles rendering the images to both eyes along with audio, collision detection, physics, animation and memory management. Different SDKs (software development kits) and APIs (application programming interfaces) are also often used to aid in the VR development process.

3. PROCESS

In this section we cover the methods used during the process and rationale for the choices. The progression of the practical steps is presented here, along with observations gathered during the process.

3.1 Scanning the environment

The first step of the process is to acquire the spatial data from the environment we want to replicate in VR. A 3D laser scanner is used to obtain the data. If the device has an option to integrate colours into the scan, it can be used to gather more information into the data.

When scanning an environment there are many things to take into consideration. The size of the space should correspond to the capabilities of the used laser scanner. If the purpose of the end result is to be explored in VR, the space should also be large enough for movement. For greater detail and coverage of the blind spots, multiple scans should be performed. The scanning locations should be carefully thought upon, taking into consideration the obstacles present in the space and the field of view of the scanner. There should be no movement or changes in the space between multiple scans to ensure a successful merging of the scans.

The room TC411 in Hervanta campus of Tampere University was chosen as the environment to be captured. The scanning was performed with a FARO Focus 3D laser scanner [8]. In addition to high-quality scans, the device also takes pictures of the scanned areas to include colours to the data. The scanner has a dome-shaped field of view and it features GPS, that tracks the locations of scans outdoors.

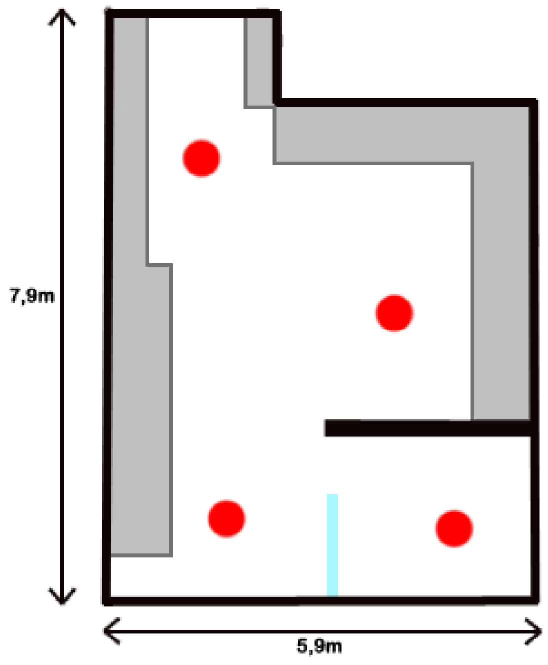


Figure 5. *Floor sketch of the scanned space*

Four scans were performed to cover the whole space properly. The floor sketch of the space is shown in Figure 5. Grey areas indicate furniture and red circles indicate the chosen scanning locations. Multiple scans were required for an otherwise relatively small space due to the furniture near the walls and the partition in the room.

Final part of this step is to process the raw data from the scans into formats that are compatible with processing software, if the scanner does not do it automatically. In our case, the scans were processed and registered with FARO SCENE [9]. It can clean up ghost points from the data, export it into various point cloud formats and align the multiple scans. The data was exported into four separate PTX format point cloud files, which included the information of scan location, points and colours.

3.2 Data processing

After the data from the environment has been collected and converted into an appropriate format, it is still not ready for being applied to VR. The data is currently a point cloud, which has no surfaces or shapes recognizable by a computer. It is simply a collection of data points in space.

In this step we process the data to make it suitable for VR applications. The purpose of this step is to create a mesh from the point cloud data that can be imported into a suitable game engine. We use two different software for the processing of the data. CloudCompare [4] is first used to clean and process the point cloud data into a triangular mesh. The mesh is then opened in MeshLab for decimation and hole filling.

The four scans from the room were first aligned and merged into a single point cloud in CloudCompare. The scans were aligned automatically during the registration in FARO SCENE, but the alignment can also be done manually by choosing corresponding points from the clouds. The redundant points outside of the room were segmented out with the segmenting tool.

Merging four point clouds with around 10 million points resulted in a single cloud with 42 million points. The scans partially overlapped, which means that there was unnecessary information in the merged cloud. We cleaned the data by subsampling it with a spatial method, where points are picked from the original cloud by a set distance between the points. The distance was set to 2 mm. Subsampling made the cloud cleaner and reduced its size, which also made it easier to process.

The final procedure before meshing is to compute per-vertex normals for the point cloud. We set the local surface model as plane for the computation, because it is the most robust to noise amongst the options in CloudCompare [5]. Orientation of the normals was done with a minimum spanning tree, where k-nearest neighbors parameter was set to 12.

After the point cloud data is cleaned and the normals have been computed, the next step is to mesh it. The screened Poisson surface reconstruction method [16] was used, because it is robust to the noise present in 3D scans. The method is implemented in CloudCompare according to the version 7 of the PoissonRecon library [6][19]. The reconstruction implementation has many parameters, but octree depth makes the most considerable difference to the result. It directly affects the quality of the mesh, while also increasing memory usage and computing time. An octree depth of 11 was chosen, since it produced the highest achievable detail with the provided computing capacity. While the method created a very detailed mesh, it also produced some unnecessary shapes in low density areas. The mesh was cleaned by filtering it by density, cutting the unnecessary shapes out.

The mesh could be exported into the game engine already, but it is very large with 22 million faces and it still contains holes. The complexity of the mesh had to be reduced and the holes filled due to the requirements of the end use. The mesh was exported as PLY (Polygon File Format) from CloudCompare to MeshLab [3] for the decimation and hole filling.

The mesh was decimated with quadric edge collapse decimation algorithm implemented in MeshLab. The method reduces the number of faces in a mesh, while preserving the shapes and normals. The mesh was decimated from 22 million faces to 1 million faces for it to work smoothly in the application. After decimation, we deleted manifold edges and vertices and closed the holes in the model with the algorithms implemented in MeshLab.

The resulting mesh fulfilled the requirements imposed by VR, though an appropriate file format had to be chosen. The Unity game engine [22] was utilized in this work; thus the compatible formats were mainly FBX and OBJ. Texture generation for the OBJ file in MeshLab produced textures with black sections that did not belong to the scene. Different parametrization methods for the texture were experimented with, but ultimately the choice was made to use per-vertex colours. Since only FBX supports per-vertex colours, it was chosen as the format to export.

3.3 Developing a virtual reality application in Unity

Importing the data into VR is done with the help of a game engine. Unity is used to make an application with the generated mesh as the environment. Two versions are created, one for the VR and one for testing the environment with a normal display, mouse and a keyboard. Depending on the VR hardware for the end use, different toolkits are used to aid in creation of the VR application.

The first step is to start a new project in Unity. We created an application to be used with VIVE Pro [12] HMD and Omnideck [17] treadmill. The software packages needed for this hardware are VRTK (Virtual Reality Toolkit) [7], SteamVR [23] and Omnifinity/Unity-SteamVR-API [18]. All the packages were imported into Unity along with the FBX mesh that was made in the previous part of the process. For the 3D testing application, only VRTK is needed.

Standard shaders in Unity do not support per-vertex colours properly. To get the colours of the mesh to show properly, a custom shader was created with the code of jhorikawa [13] from GitHub. A new material set to use the vertex colour shader was also created.

Import options of the environment mesh were changed for it to work properly. The original unit of measure was in meters, but Unity interprets the values as millimetres. Setting the scale of the model to 100 corrected the size. Colliders were generated for the mesh to avoid the player object from falling through the floor or walking through walls. The material of the model was changed to the previously created vertex colour material for the colours to show properly. Unity flipped the model sideways because of different coordinate systems, so it had to be rotated -90 degrees in x axis for it to be level.

A simple 3D application was made first for testing the environment with a keyboard and mouse. For the application, a new scene was created in Unity. The default camera was removed, and *VRSimulator_CameraRig* was added to the scene from the VRTK along with the environment model. The *VRSimulator_CameraRig* handled the camera and the mapping of controls, so the application could be built straight into an executable for testing.

To create the VR application, an example scene was used from the Omnifinity package for the camera and player rig. All the pre-set objects were removed, and the environment mesh was added to the scene. The model was then centred in the middle so that the player and camera rig were inside it. The VR application was ready, and it was built into an executable for testing with the hardware.

4. RESULTS AND ANALYSIS

In this section the results obtained during and after the process are presented and analysed. Noise and errors in the data and their causes are examined. Improvements to the process are also proposed.

4.1 Environment scans

The scanning of the environment was successful, with each scan taking around 10 minutes to finish. Four scans were acquired from different locations in the room.

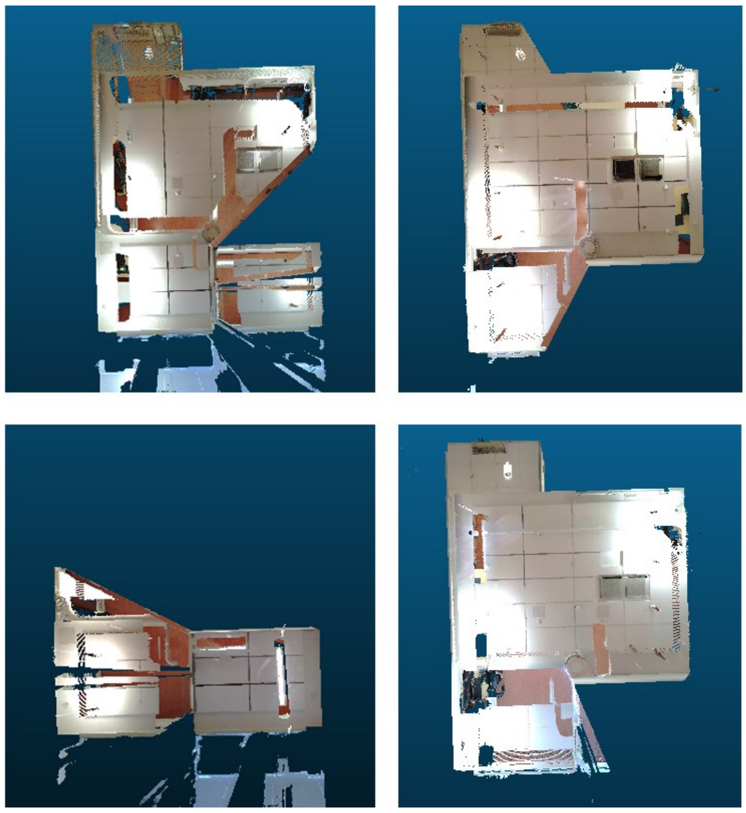


Figure 6. *Point clouds acquired from scanning*

The scans covered parts of the whole area from different points of view. Figure 6 shows the multiple scans from the top view in RGB colours. A single point cloud resulted from merging the scans, shown on Figure 7.

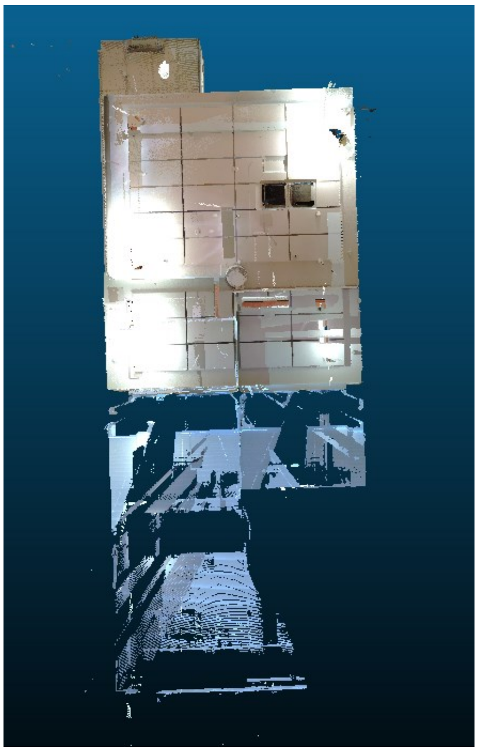


Figure 7. *The merged point cloud*

The scanner captured unnecessary points through the windows and other individual stray points outside of the room. The point cloud was very dense at some parts due to the scans partially overlapping. While the scans were generally high quality and a large amount of data was collected, they still failed to capture all the little details inside the room.

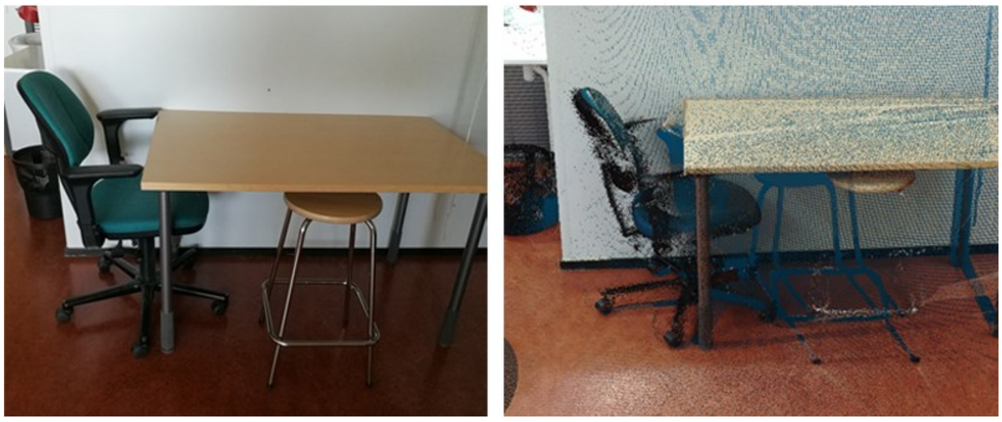


Figure 8. *Photo to cloud comparison of details*

In Figure 8, a photo of the space and the scan are compared. While the scanner captured larger shapes and details well, the narrow legs of the stool have almost no points at all. The areas behind objects were also scarce of points. In addition to some areas not appearing on the scan, some unnecessary noise was also present on the scans. For example, on the chair there are some points outside of its outlines.

All things considered; the scanner can capture the general shape of the room very well. Objects and furniture that partly block the scanners field of view can cause problems, especially in a crowded space. Windows and other reflective surfaces can also prove to be a challenge when scanning. The windows partly reflect and partly let the laser beams pass, so the points acquired from them can be inconsistent.

Problems with field of view could be remedied by increasing the number of scans or reducing the number of objects present in the space. The objects could also be scanned separately and added to the scene at a later part of the process. However, if the purpose is to make the process straightforward and effortless, simply moving the objects to the sides of the room is sufficient.

4.2 The surface reconstructed data

Many different methods and orders for the processing were experimented with. For example, computing normals separately for each scan or filtering the data. Small adjustments with most parameters did not make any considerable difference. Most of the problems in the mesh were caused by the noise and errors in the original cloud. However, the most important parameter was the octree depth in the Poisson algorithm.

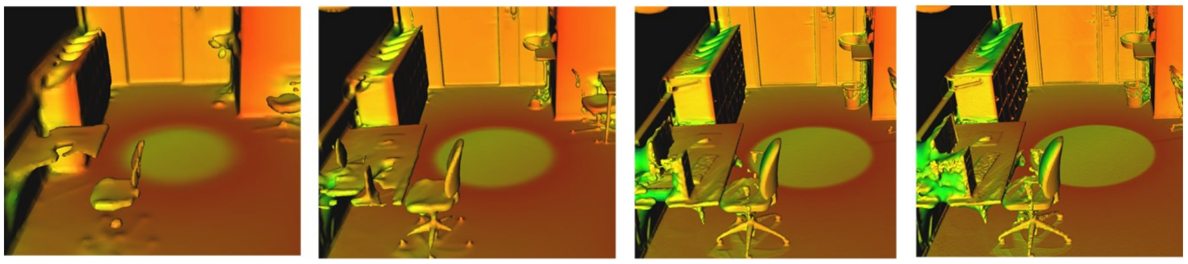


Figure 9. *Meshes with different octree depths from left to right 8, 9, 10, 11*

The differences between different octree depth levels were considerable. Figure 9 shows that the smaller depths easily leave even larger details out of the resulting mesh. In the octree depth 8 mesh, almost the whole desk is left out. Higher octree depth levels produce shapes and edges better, they also make smooth surfaces like the floor more detailed and rougher. Increasing the octree depth also increases the amount of faces and the computing time.

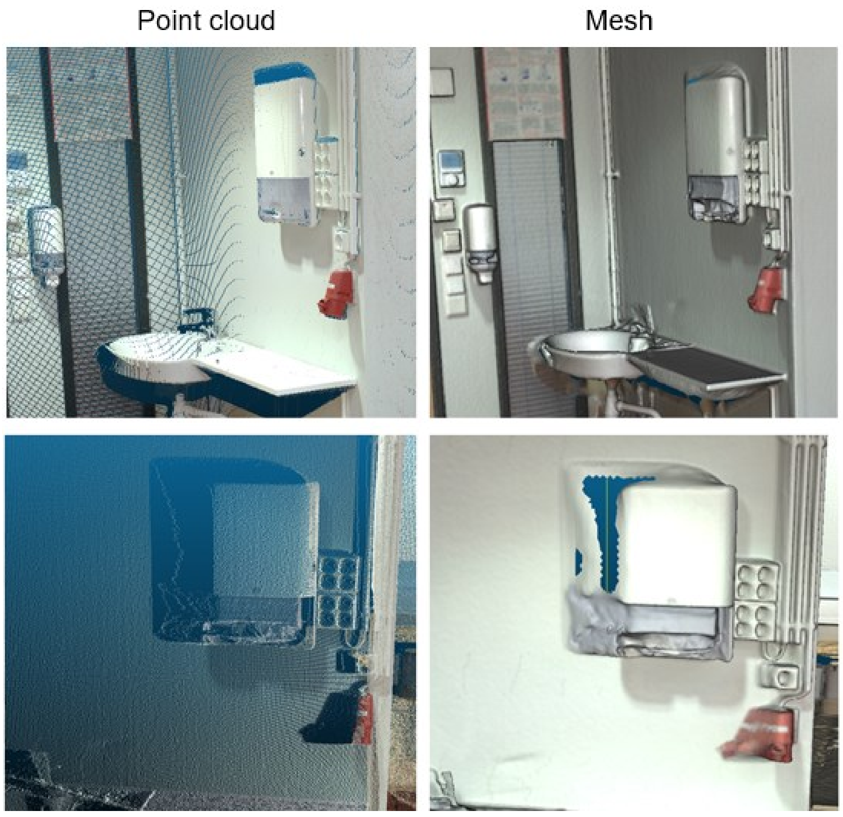


***Figure 10.** Example of scanning occlusions' effect on mesh*

On the upper row of Figure 10 the objects on the wall are shown from the perspective of the scanner, while on the lower row they are shown from the front. The occlusions left due to the limited point of view of the scanner caused the mesh to extend some of the objects in an unnatural way. Some larger occlusions also left a hole on the mesh, as seen from the picture on the bottom right.

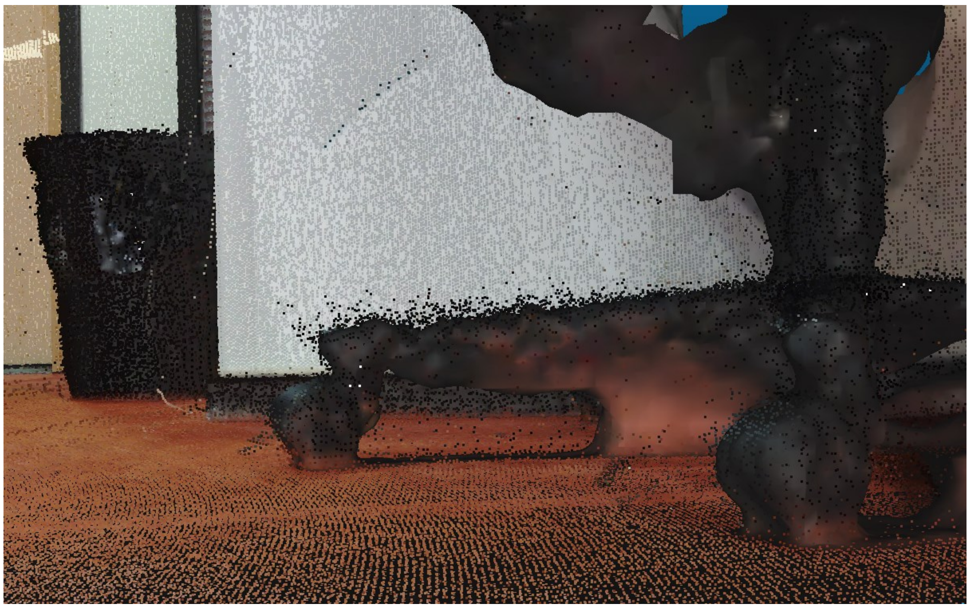


Figure 11. *Original point cloud noise on top of the mesh*

The generated mesh can only be as accurate as the point cloud it was derived from. Figure 11 shows the noise of the point cloud on top of the generated mesh of a chair leg. While the meshing algorithm does handle the noise well, it is not perfect. As can be seen from the picture, the algorithm follows the points very well, but the result is bumpy.

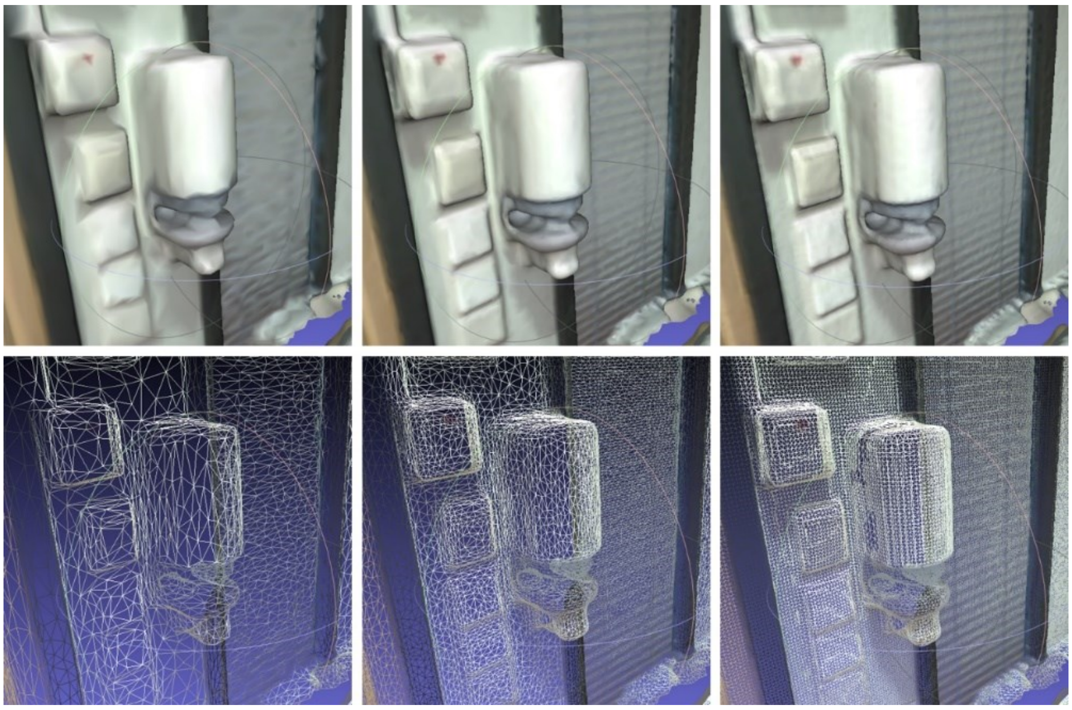


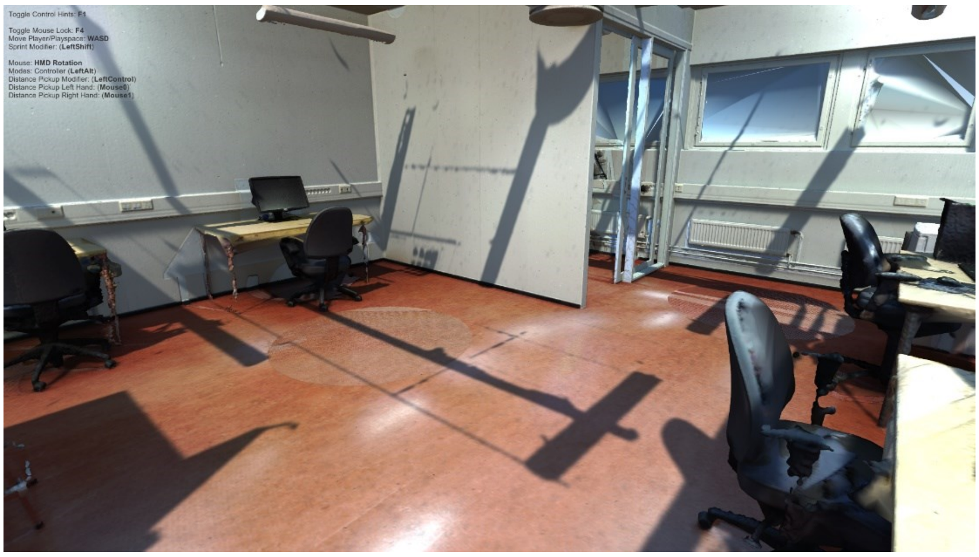
Figure 12. *Effects of decimation on the mesh from left to right 1, 5 and 22 million faces*

Figure 12 shows the effects of decimation on a small object in the model. The original mesh had 22 million faces, from which the five and one million face meshes were decimated. While decimation does decrease the quality and sharpness of the mesh, with enough faces it does not affect the whole model too much in this case. Smoothing some rougher areas might even be more visually pleasing in the result and reducing the number of faces reduces the load on the system significantly.

Overall, octree depth makes a considerable difference in the fine details during the processing of the mesh. The quality and accuracy of the original scans mostly dictate the result of the surface reconstruction. The mesh can be decimated safely, without downgrading the quality significantly. Additional accuracy comparison figures of the resulting mesh and real-world photos can be found on the appendix A.

4.3 Visualization applications

Two applications were created from the scanned environment. The VR application was tested with an HMD and Omnideck, and the 3D application was tested with a standard display, mouse and a keyboard.



Toggle Control Hints: **F1**

Toggle Mouse Lock: **F4**
Move Player/Playspace: **WASD**
Sprint Modifier: (**LeftShift**)

Mouse: **HMD Rotation**
Modes: Controller (**LeftAlt**)
Distance Pickup Modifier: (**LeftControl**)
Distance Pickup Left Hand: (**Mouse0**)
Distance Pickup Right Hand: (**Mouse1**)

Figure 13. *Snapshot of the 3D testing application*

The creation and testing phases of the 3D application were successful. A mesh with 11 million faces was used for the model, and the application ran smoothly on a desktop computer. W, A, S and D keys were used for moving in the horizontal directions and mouse was used to look around. This application allowed the movement through walls, so the scanned room could also be looked at from the outside. The model looked like it should, and the colours were shown correctly.

Several problems with version compatibility of the APIs and the mesh size were encountered at first with the VR application. The Omnifinity API used for the Omnideck did not-work with the latest SteamVR version. However, downgrading SteamVR to 2.0.1 solved the issue. The computer used also had problems with displaying the environment properly. This was discovered to be because of the size of the mesh, which had 11 million faces. It was decimated to 1 million faces, which made the VR application work smoothly.

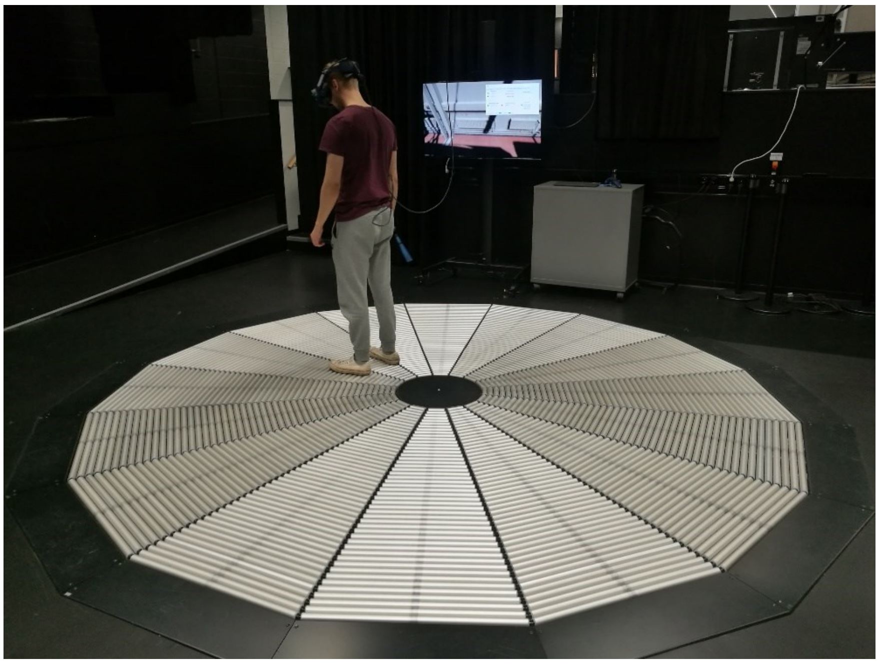


Figure 14. *The VR application testing environment*

The finished VR application was tested with HTC Vive and Omnideck treadmill in CIVIT (Centre for Immersive Visual Technologies) [21] laboratory of Tampere University, Hervanta campus. The tracking system of VIVE followed the movement of the player. The information was input to the Omnideck software through SteamVR API. It made the Omnideck return the player to the middle when walking on the platform, while simultaneously moving forward in the VR application. The VIVE Pro HMD handled displaying the 3D environment in a stereoscopic view. The view that the player was seeing was also shown on the screen.

All in all, the testing of the applications was a success. The Omnideck platform is an exceptional and expensive device, not usually found in personal or common use. The scarcity of the user base causes the compatibility and other issues be more common. Leaving the Omnideck out of the VR application simplifies the process.

When developing an application only for the HMD and the tracking system, the room could be scaled to the tracking area, or teleportation methods could be implemented in the application. Various supplementary features, such as object grabbing or door opening, could also be implemented to further enhance the application. Additionally, more environments could be scanned and combined in the game engine to create a larger integrity.

5. CONCLUSION

The process of replicating a space from real life into virtual reality is possible with the utilization of modern technologies and algorithms. With knowledge and skills to use the available tools and software, the process becomes straightforward and relatively fast. Achieving adequately accurate results is completely feasible.

The purpose of this thesis was to shed light on the whole process. Scanning, processing and transferring the data to virtual reality were covered by practical steps, analysis and discussion of the data. The goal of replicating an environment from the real world in VR was achieved. The process could be done in a straightforward way using pervasive methods for the whole data. Many observations were made regarding different parts of the process.

In the scanning phase it is important to gather enough data, as accurately as possible. Multiple scans should be used to cover blind spots of the scanner. The scanning of spaces crowded with line of sight blocking objects or partitions should be performed profoundly by planning the scanning locations and preparing the space in advance. The quality of the scans makes the most considerable difference in the final environment model.

The goal of the processing phase is to reduce the amount of redundant data and achieve an accurate mesh from the data acquired from the scans. Poisson surface reconstruction algorithm is a good choice for meshing 3D scanner data due to it being robust to noise. When using the algorithm, mainly the octree depth dictates the level of fine details in the resulting mesh. The mesh should be decimated for it to be lighter to utilize in VR.

The final part of the process is to create a VR application with the produced mesh. A game engine and toolkits are used to make the process effortless. The final product is an executable file that can be transferred and executed anywhere with suitable VR equipment and software.

REFERENCES

[1] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, C. Silva, State of the Art in Surface Reconstruction from Point Clouds, Eurographics 2014 - State of the Art Reports, 2014, pp. 161-185. Available (accessed 29.4.2019): https://hal.inria.fr/hal-01017700/document

[2] F. Bernardini, H. Rushmeier. The 3D model acquisition pipeline. Computer Graphics Forum, Vol. 21, Iss. 2, 2002, pp. 149-172.

[3] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, MeshLab: an Open-Source Mesh Processing Tool, Sixth Eurographics Italian Chapter Conference, 2008, pp. 129-136, Available: http://www.meshlab.net/

[4] CloudCompare (version 2.10.2) [GPL software], 2019. Available: https://www.danielgm.net/cc/

[5] CloudCompare, CloudCompare documentation (Normals/Compute), Available (accessed 29.4.2019): https://www.cloudcompare.org/doc/wiki/index.php?title=Normals%5CCompute

[6] CloudCompare, CloudCompare documentation (Poisson Surface Reconstruction plugin), Available (accessed 29.4.2019): http://www.cloudcompare.org/doc/wiki/index.php?title=Poisson_Surface_Reconstruction_(plugin)

[7] Extended Reality Ltd, VRTK. Available (accessed 11.4.2019): https://github.com/ExtendRealityLtd/VRTK

[8] FARO, FOCUS. Available (accessed 19.4.2019): https://www.faro.com/products/construction-bim-cim/faro-focus/

[9] FARO, SCENE. Available (accessed 19.4.2019): https:/www.faro.com/products/construction-bim-cim/faro-scene/

[10] C.S. Gardner, Lidar, AccessScience, McGraw-Hill Education, 2014. Available (accessed 20.3.2019): https://www.accessscience.com/content/lidar/380750

[11] S.J. Gortler, Foundations of 3D computer graphics, Cambridge, MA: The MIT Press, 2012.

[12] HTC, Valve Corporation, VIVE Pro. Available (accessed 29.4.2019): https://www.vive.com/us/product/vive-pro/

[13] J. Horikawa, VertexColor shader for Unity, GitHub. Available: (accessed 17.4.2019) https://gist.github.com/jhorikawa/7a236ae0b801bf2aca2d8a3038b9bf40

[14] V. Joala, Laserkeilauksen perusteita ja mittauksen suunnittelu, Leica Nilomark Oy, Espoo, 2006. Available (accessed 20.3.2019): https://docplayer.fi/7209674-Laserkeilauksen-perusteita-ja-mittauksen-suunnittelu.html

[15] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, Proceedings of the fourth Eurographics symposium on geometry processing, 2006, pp. 61-70.

[16] M. Kazhdan, Hugues Hoppe, Screened Poisson surface reconstruction, ACM Transactions on Graphics (TOG), Vol. 32, Iss. 3, 2013, pp. 1-13.

[17] Omnifinity, Omnideck. Available (accessed 30.4.2019): http://omnifinity.se/

[18] Omnifinity, Omnifinity/Unity-SteamVR-API, Available (accessed 18.4.2019): https://github.com/Omnifinity/Unity-SteamVR-API

[19] PoissonRecon library, Available (accessed 18.4.2019): http://www.cs.jhu.edu/~misha/Code/PoissonRecon/Version11.02/

[20] W.R. Sherman, J.D. Will, A. Craig, Developing virtual reality applications: Foundations of effective design, Elsevier Science & Technology, 2009.

[21] Tampere University, CIVIT laboratory. Available (accessed 30.4.2019): http://www.tut.fi/civit/

[22] Unity technologies, Unity, Available (accessed 11.4.2019): https://unity.com/

[23] Valve Corporation, SteamVR Plugin. Available (accessed 11.4.2019): https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647

APPENDIX A: ACCURACY COMPARISONS

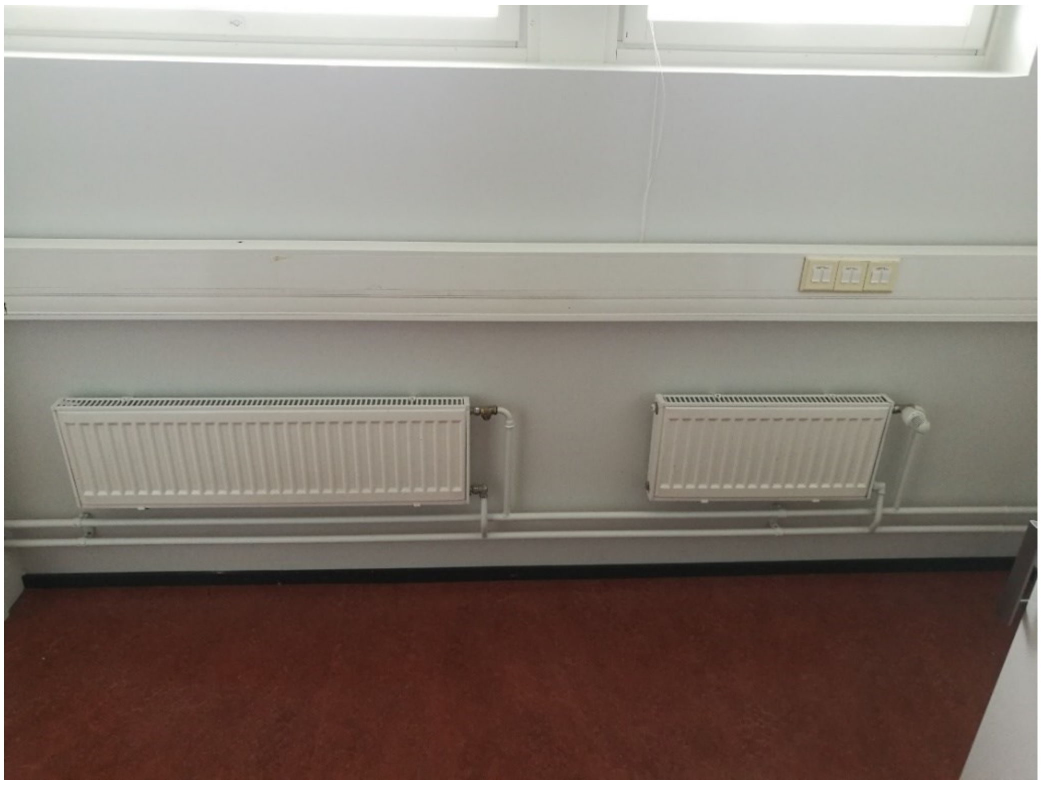


Figure 15. *Accuracy comparison 1: Photo*

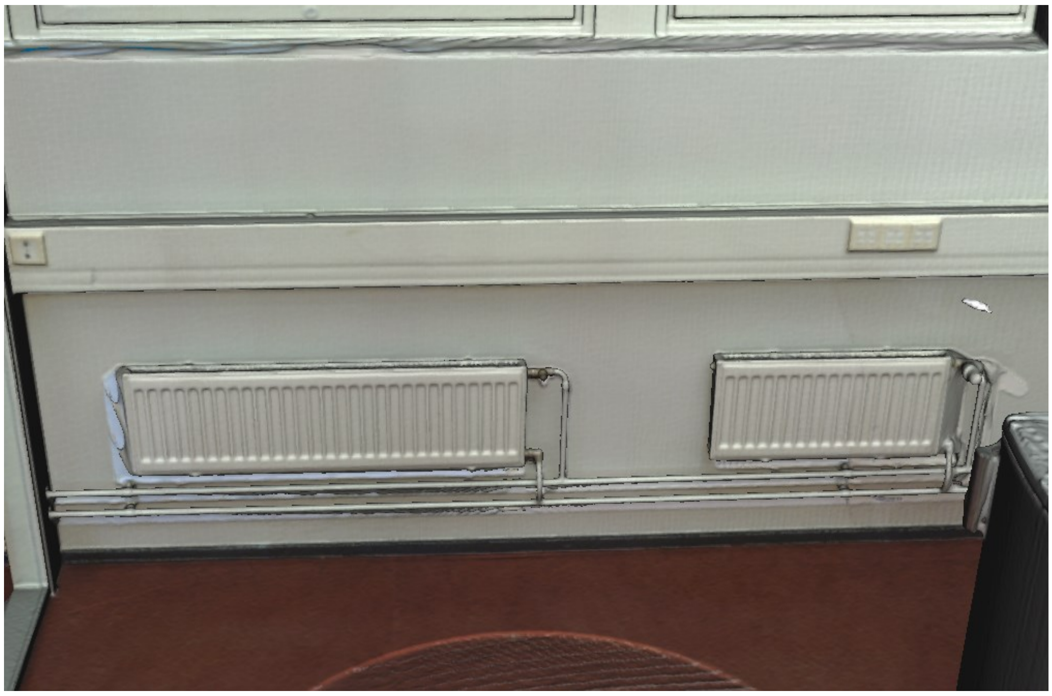


Figure 16. *Accuracy comparison 1: mesh*

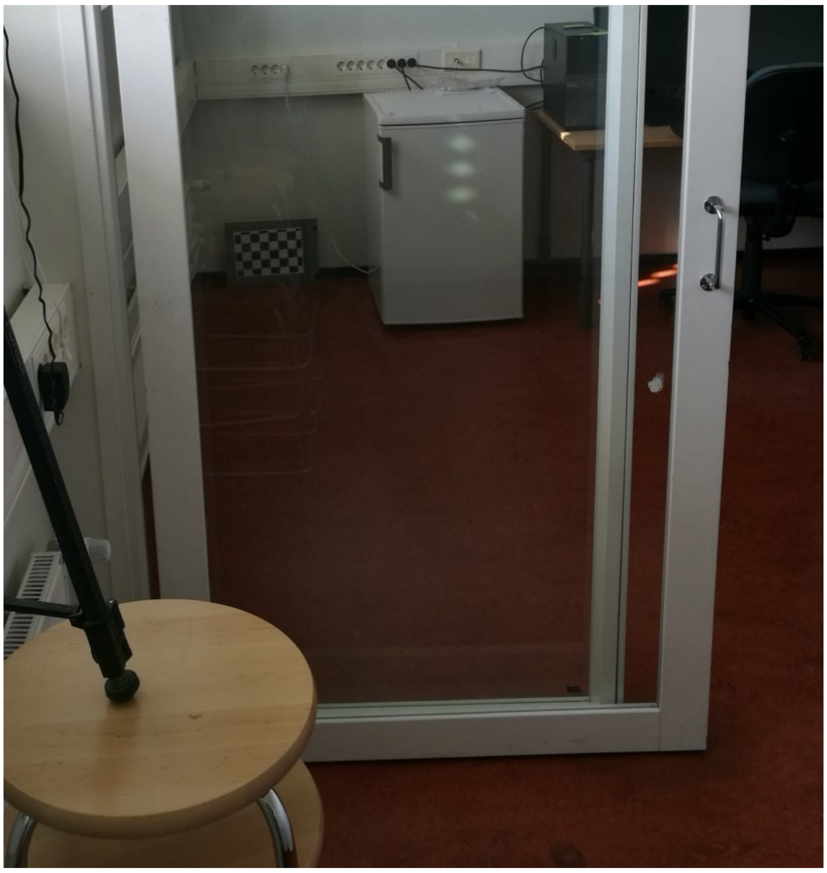


Figure 17. *Accuracy comparison 2: Photo*

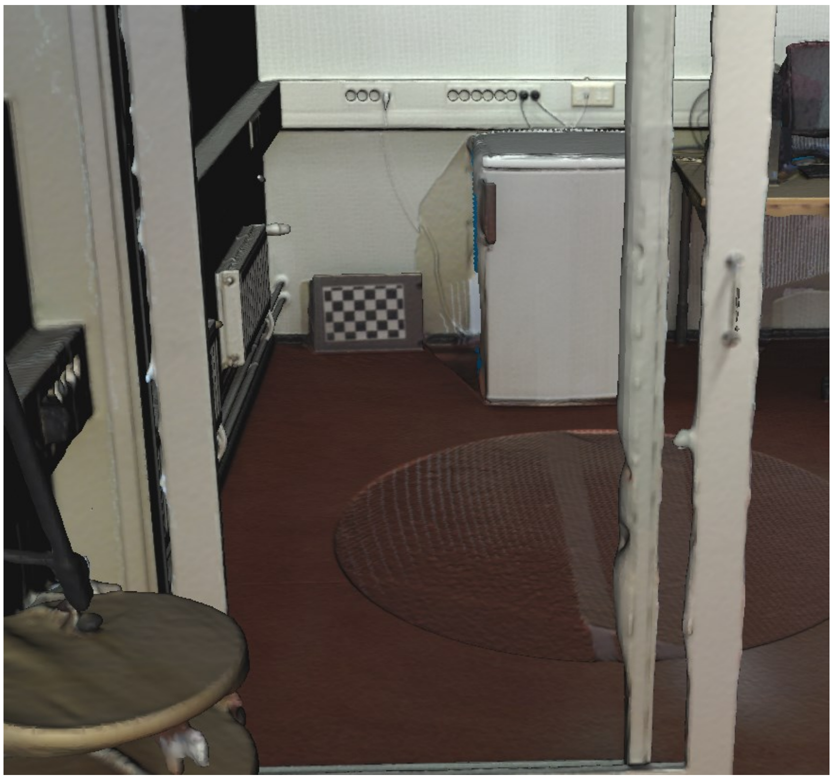


Figure 18. *Accuracy comparison 2: Mesh*

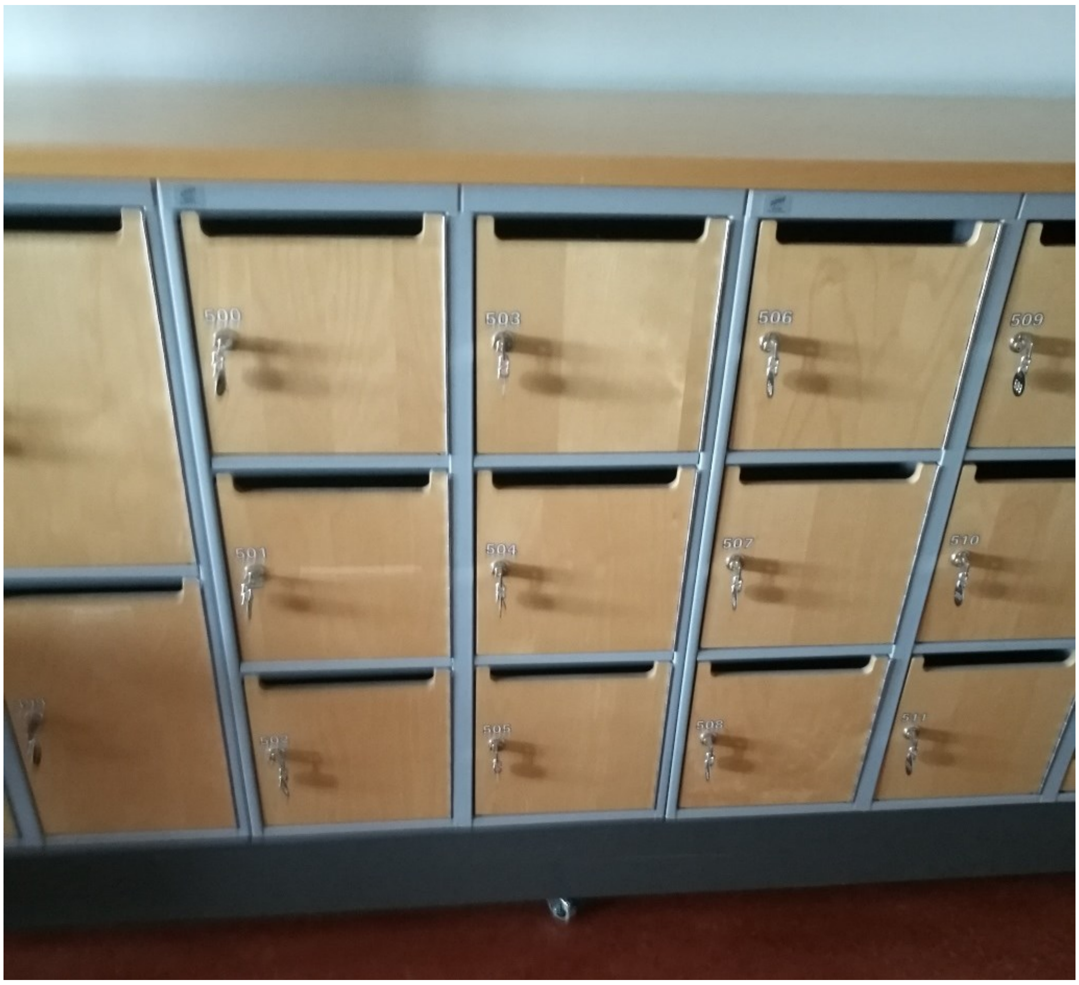


Figure 19. *Accuracy comparison 3: Photo*

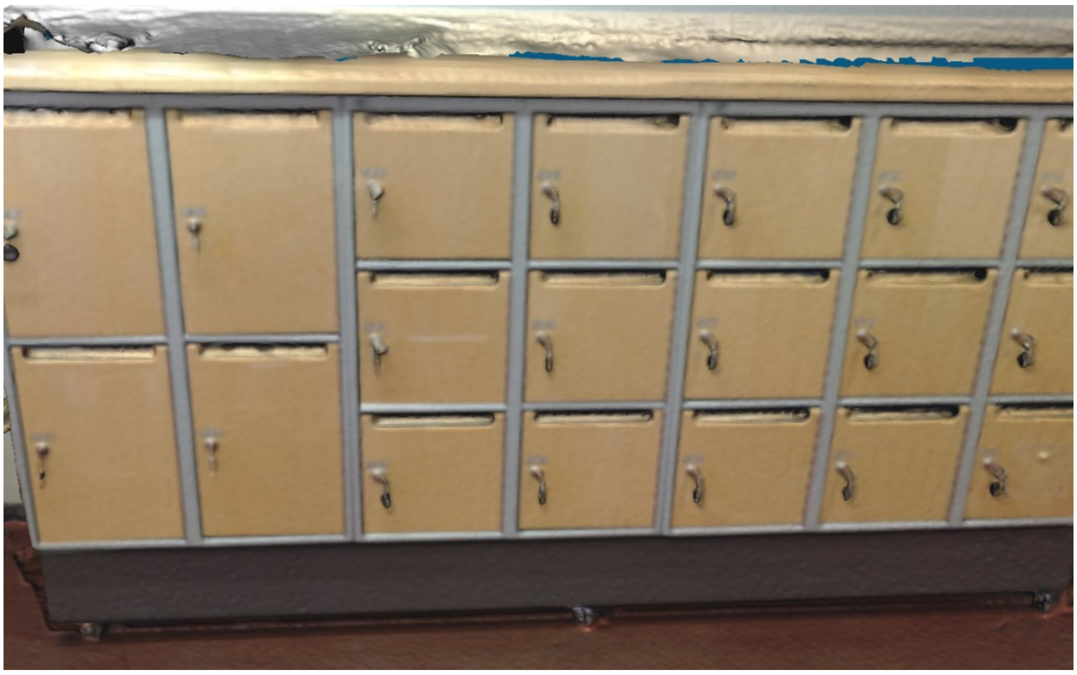


Figure 20. *Accuracy comparison 3: Mesh*