Visualization of laser scanner point clouds as 3D panorama

Using laser scanning to reconstruct the facade of the Pellerhaus Nürnberg in its historic state

By

Adam Kalisz
2265000

A thesis presented for the degree of
Bachelor in Media Engineering

Elektrotechnik Feinwerkmechanik Informationstechnik
Georg-Simon-Ohm Technische Hochschule Nürnberg

Thesis advisors:
Prof. Dr. Stefan Röttger
Prof. Dr. (USA) Ralph Lano

Submission date:
July 15th, 2015

Location:
Nuremberg, Germany

Keywords:
LiDAR, point, cloud, laser, scanning, panorama, Blender

Declaration

Plagiarism Declaration in Accordance with Examination Rules

I herewith declare that I worked on this thesis independently. Furthermore, it was not submitted to any other examining committee. All sources and aids used in this thesis, including literal and analogous citations, have been identified.

_____

Signature

Abstract

This study examines a novel approach to convert point clouds generated via laser scanning into textured 3D-meshes. The title of this paper is "Visualization of laser scanner point clouds as 3D panorama". The approach is field-tested with a use case scenario where the interested reader will learn about our research on the 3D-model reconstruction of the historic Pellerhaus in Nuremberg, Germany, as it looked before its destruction during World War II.

The motivation behind the report, details about the project and existing solutions for creating virtual reconstructions are introduced in Chapter One. The background research that provided necessary fundamentals to start the project, for example how the Pellerhaus evolved or what exactly a 3D panorama is, is described in the second chapter. The third chapter presents the development process of the software tools applied to achieve the goal of reconstructing historic 3D models from various data, such as images and laser scans. To accomplish this, a custom converter software was written, which reads point cloud files and outputs the meshed and textured 3D-object file. The working title of this software is "PointCloud2Blender", *PC2B* in short. As a real world use case the creation of a photorealistic three-dimensional mesh from laser scans via LIDAR devices is described in detail in Chapter Four. Chapter Five concludes the work and presents future work. It contains the results, failures, and successes of this research. Furthermore, it discusses different possible ways to build upon the fundamental insights gained from this report.

Due to our modern open culture with several open software, hardware and movie projects - mainly inspired by the Blender Foundation - this research is being made available to the public. During the time of the writing of this thesis the progress is published online at http://bachelor.kalisz.co.

Acknowledgements

This research could not have been performed without the assistance, patience, and support of many individuals.

On behalf of the historical expertise required for this research, I would like to thank the Geschichtsarchiv Langwasser, including Mrs. Edith Schroth and Mr. Alfred Schroth for their constant support in providing old photographs, material and making contact to various institutions like archives, museums and companies. They initiated the contact with the Altstadtfreunde Nürnberg e.V. as well.
Additionally I would like to thank the Altstadtfreunde Nürnberg e.V. for a significant amount of historic pictures and professional guidance regarding the history of the Pellerhaus. I am thankful for the opportunity to be supported by chairman Mr. Karl-Heinz Enderle during my research.

Secondly, I must thank my thesis advisor, Mr. Prof. Dr. Stefan Röttger for mentoring me during my undergraduate studies. He proved his confidence in me by encouraging me to teach computer graphics to other students, letting me demonstrate how much fun it can be to create graphics with the open source 3D graphics suite Blender, and offered me several jobs in 3D animation. His insight lead to the initial proposal to examine the possibility of reconstructing the Pellerhaus facade. In addition I would like to extend my gratitude to Mr. Prof. Dr. (USA) Ralph Lano for supervision during my studies. His teaching style and enthusiasm made a strong impression on me and I have always carried positive memories of the classes I attended. Although the classes I took were not mandatory, they were unique, interesting and fun (e.g. XBox programming with Unity), and he was always very helpful and friendly. I would like to thank you very much for your support and understanding over these past four years.

Finally I would like to extend my deepest gratitude to my family without whose love, support and understanding I could never have completed this bachelor's degree.

Contents

1 Introduction **8**
- 1.1 Motivation . . . 8
- 1.2 Initial project specification . . . 9
- 1.3 Project schedule . . . 10
- 1.4 State-of-the-art methods for 3D reconstruction . . . 11
  - 1.4.1 Light Detection And Ranging (LiDAR) . . . 11
  - 1.4.2 Ultrasonic . . . 12
  - 1.4.3 Photogrammetry . . . 12
  - 1.4.4 Depth Cameras . . . 14
  - 1.4.5 Google Maps ® . . . 14
  - 1.4.6 Open Street Map ® . . . 14
  - 1.4.7 Bavarian State Office for Survey and Geoinformation . . . 15
  - 1.4.8 Autonomous mapping with UAV's and SLAM . . . 15
  - 1.4.9 Manual methods . . . 15
- 1.5 Defining the scope of this research . . . 15

2 Background Research **16**
- 2.1 Historical fundamentals . . . 16
  - 2.1.1 Renaissance . . . 16
  - 2.1.2 Pellerhaus . . . 19
  - 2.1.3 Architects of the Pellerhaus Nürnberg . . . 22
- 2.2 3D Panorama . . . 22
  - 2.2.1 Creating panoramas . . . 24
  - 2.2.2 Types of projections . . . 24

3 Conversion: From point cloud to Blender 3D **26**
- 3.1 Concept and preparation . . . 26
  - 3.1.1 Use case diagram . . . 27
  - 3.1.2 Laser scanning on location . . . 27
- 3.2 Generating data and testing algorithms . . . 28
  - 3.2.1 BlenSor . . . 28
  - 3.2.2 Test-Add-on for Blender . . . 29
- 3.3 How the conversion works . . . 30
  - 3.3.1 Pipeline overview . . . 30
  - 3.3.2 Point Cloud Importer . . . 30
  - 3.3.3 Determination of original point cloud resolution . . . 31
  - 3.3.4 Coordinate system representations . . . 32
  - 3.3.5 Converting from cartesian to spherical and vice versa . . . 32

3.3.6 Types of projections . . . 33
3.3.7 Saving textures . . . 33
3.3.8 Meshing . . . 34
3.3.9 Texture Coordinates and Normals . . . 34
3.3.10 Mesh Exporter . . . 36
3.4 How the conversion is implemented . . . 37
3.4.1 UML class diagram . . . 37
3.4.2 Working with the PC2B User Interface . . . 39
3.4.3 OpenGL Point Cloud Viewer . . . 40
3.4.4 Optimizations . . . 40

4 Production: Recreating the Pellerhaus from 1605 **43**
4.1 Modeling the current Pellerhaus facade . . . 43
4.1.1 Using the PC2B converter software . . . 43
4.1.2 Using UAV references with photogrammetry . . . 44
4.1.3 Using reference images . . . 45
4.2 Modeling the historic Pellerhaus facade . . . 46
4.2.1 Using historic images as guide . . . 46
4.2.2 Using historic stereoscopic images with photogrammetry . . . 47
4.3 The finished models . . . 48

5 Conclusion and Future Work **49**
5.1 Conclusion . . . 49
5.1.1 Mesh generation . . . 49
5.1.2 Handling non-LiDAR data . . . 50
5.2 Future Work . . . 52
5.2.1 Improving memory management and performance . . . 52
5.2.2 Multi-threaded PC2B Server . . . 52
5.2.3 Integration in the Blender core . . . 52
5.2.4 3D printing, lenticulars and stereoscopic movies . . . 52
5.2.5 Augmented and Virtual Reality . . . 53

A Appendix **54**
A.1 Pellerhaus Architect Biography . . . 54
A.1.1 German . . . 54
A.1.2 English . . . 54
A.2 Software used . . . 55
A.2.1 LaTeX . . . 55
A.2.2 FARO SCENE LT . . . 55
A.2.3 Blender 3D . . . 55
A.2.4 Meshlab . . . 56
A.2.5 Visual SfM . . . 56
A.2.6 CMP-MVS . . . 56
A.2.7 AgiSoft PhotoScan Professional . . . 56
A.3 Programming frameworks and libraries . . . 56
A.3.1 Qt 5.4 . . . 56
A.3.2 OpenGL . . . 56
A.4 Delaunay Tetrahedralization Texture Maps . . . 57

List of Figures

1.1 LiDAR Scanner Point Cloud of the Pellerhaus 11
1.2 Multi-view reconstruction point clouds generated from 356 photos . . 13

2.1 Pellerhaus around 1905 19
2.2 Pellerhaus 2015 20
2.3 The Evolution of the Pellerhaus 20
2.4 2D Panorama of the Pellerhaus 23
2.5 Mapping a two-dimensional panorama onto a three-dimensional sphere 23
2.6 Converting from cartesian to spherical coordinates, visually 24
2.7 Three sample projection types used in PC2B 25

3.1 Use Case Diagram 27
3.2 Scanning with FARO Focus^{3D} 28
3.3 Custom Blender Addon 29
3.4 The stages of the conversion process 30
3.5 Sample generated point cloud files 31
3.6 Panoramic images generated from the imported point cloud 33
3.7 PC2B Meshing Algorithm 34
3.8 PC2B Texture Coordinates viewed in Blender 35
3.9 PC2B Normal filtering concept 35
3.10 The generated mesh before and after normal filtering in PC2B 36
3.11 PC2B UML class diagram (simplified) 38
3.12 PC2B Graphical User Interface 39
3.13 PC2B OpenGL Viewer 40

4.1 The manually registered LiDAR scans of the Pellerhaus 44
4.2 Early draft of the Pellerhaus facade of around 1601 46
4.3 Historic stereo pair from around 1876 47
4.4 Automatic reconstruction results from historic stereo pairs 47
4.5 The finished 3D models of the Pellerhaus 48

5.1 Comparison of meshing in Photoscan vs. PC2B 50
5.2 Depth Camera images 51
5.3 Meshing of Depth Cameras with PC2B 51

A.1 Texture maps generated with Delaunay Tetrahedralization 57

Chapter 1

Introduction

This chapter introduces the project by providing a broad overview of the reason why this topic was chosen, consideration of various state-of-the-art technologies available for executing the research, and the definition what result is expected from this report.

1.1 Motivation

The field of 3D computer graphics has always been a fascinating subject to me. Creating virtual worlds and being able to inspect those from every possible viewpoint is a great way to present almost any object one can think of to a wider audience. I completed an apprenticeship as an A/V media designer, where computer graphics are a helpful tool to, for example previsualize camera work. The most impressive fact about 3D is that it has so many versatile applications in many fields. 3D information can be retrieved from 2D images, taken with a real photo camera, via photogrammetry and can, in turn, be rendered onto a flat computer screen by rendering a three-dimensional scene with a virtual camera. At the point an object is available as a 3D model, it can be postprocessed in various ways. It can be animated, physically simulated, and eventually rendered as a video. With modern display technologies the movie can be played out as a stereoscopic one and viewed with anaglyph (red, cyan), polarized, shutter or even without glasses by using, for example a parallax barrier display (see Wikipedia 2014, [Wik14]). Furthermore, objects can become tangible via 3D printing or can be inspected interactively in games with the help of virtual reality glasses like the Oculus Rift (see Oculus VR 2015, [Ocu15]). It is amazing that anyone can create and enjoy those virtual worlds today.

Additionally, I am highly interested in historical topics. As an active member of a local citizens association and representative of a settlement, where I am always available for any citizenship matters that people might have, I get to know many interesting people and the projects they are working on. Thus I am learning a lot about interesting historical facts and development of culture. Of course, this does not only include positive history. The history of the place where I live, Langwasser, district of Nuremberg, Germany, is particularly terrifying and shocking. The district was formerly used for tent cities and the Märzfeld ("March Field", a representation and parade ground) during the Reich Party Congress in Nuremburg, Germany, between 1933 and 1938 (see Wikipedia 2015, [Wik15a]). The construction of a railway

station, called Bahnhof Märzfeld, which is located right in the center of Langwasser, was partly finished in 1938. That station was used initially to transport the members of the Reich Party Congress to events. During World War II it was used for the deportation of about 940 people to concentration camps, where only 17 of them survived (see Stadtteilforum Langwasser 2015, [Sta15]). This railway station is in a ruinous condition at the moment. People go by without noticing that this is real history passing them by. This was a big concern for me, so I started to search for ways to present history in a modern way, making it educational on the one hand and enjoyable on the other hand.

Consequently, I talked to my professor, Mr. Dr. Stefan Röttger, about my wish to use laser scanning for historic 3D reconstruction. Suprisingly, my professor told me that we have a laser scanning device at the university which could be used for a thesis. The moment he told me that was the moment I made my decision to center my thesis around laser scanning.

Lastly, I was strongly motivated by researching how the laser scanner point cloud can be used to create a historic building model based off of a recent laser scan. 3D software enables a user to tweak automatically generated meshes or even to add new geometry. Due to my personal experience with the open source 3D graphics suite Blender and the decreasing interest in other software like Autodesk 3ds Max or Maya in favor of Blender (see Google Trends 2015, [Goo15]), I decided to use Blender for the 3D modeling and animation part of this research. According to the trend it is a better option since it is being used by a greater number of artists and therefore future work will be of help to a lot of people. In addition, the source code of Blender is open. Any research based on it might benefit other researchers due to its open nature. Inspired by the Blender Foundation, I wish to make my work available to the public as much as possible during and after the research. Every person should have the right to learn from the findings in my report. As a result, it was necessary to be able to work with laser scanner output, namely point clouds, in Blender. Unfortunately Blender is not designed to work with point clouds at the time of this writing. This research should adress this issue by providing a way to complete the laser scanning production pipeline for artists who want to use Blender, though not exclusively!

As will be described in greater detail hereafter, the aforementioned facts lead to an initial project specification.

1.2 Initial project specification

The idea for this research started with the personal concern of reconstructing a historical site, like the old railway station in Langwasser, in its historic state. Due to the fact that this railway station has never been fully finished and therefore there exists only poor historical documentation, a 3D reconstruction would not be complete. Luckily the famous Pellerhaus was the perfect candidate for this research[1]. After its destruction during World War II, it was rebuilt quite differently from the original state. While construction of the inner courtyard is almost finished at the time of this writing, the facade still looks modern. It became clear that the main research topic would examine ways to reconstruct the Pellerhaus facade in its historic

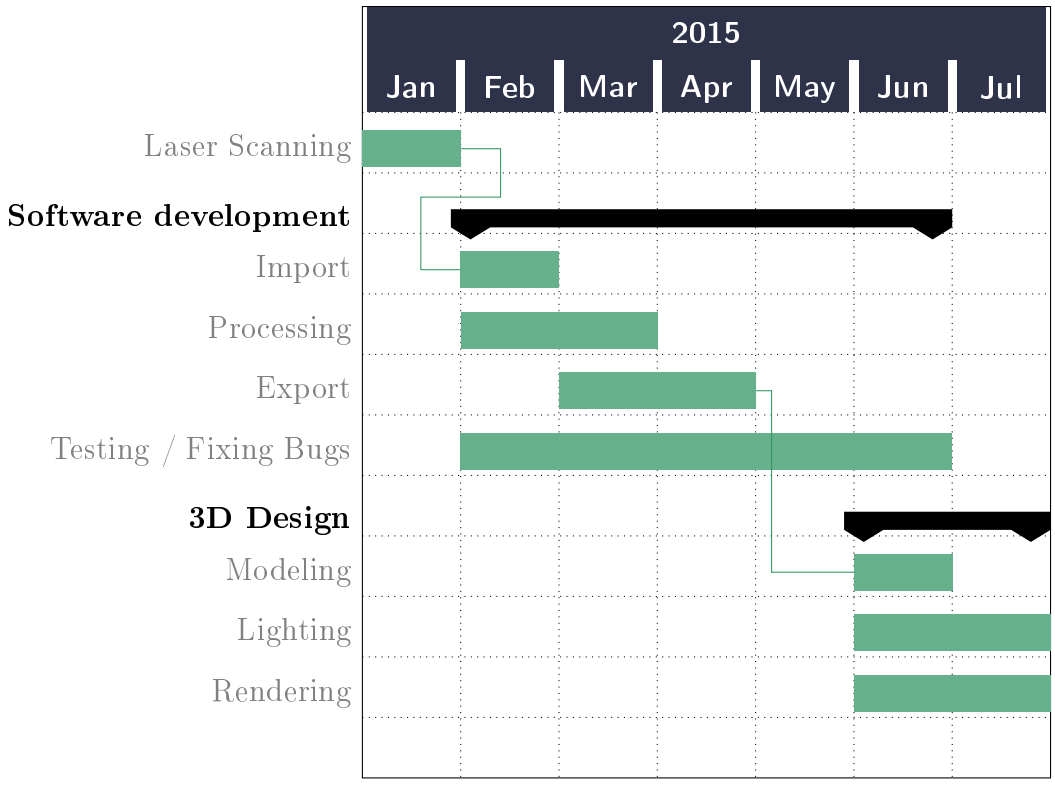
[1]Number of existing historical photos: Bahnhof Märzfeld: 9; Pellerhaus Nürnberg: 190

state. A more concrete specification was defined by considering how this would be done. The current state of the building has to be captured with laser scanning technology to get the correct measurements from the real world reference. We use a FARO Laser Scanner Focus^{3D} X Series device for this project. This point cloud data then needs to be processed. To do so, a custom software must be written, which can read a file format exported from the proprietary FARO SCENE application, create a panoramic image representation of the data, use it to generate a 3D mesh surface and export this mesh to a widely supported file format. This research will mostly rely on the open source software Blender to model and animate the historic state of the Pellerhaus, thus it is crucial to provide a compatible output to be used as a basis for the design process. By creating a textured surface from the point samples, this research will provide a way for the artist to overcome a bad design decision in Blender, which makes it incapable of displaying or rendering colored point clouds at all (see thread by author on BlenderArtists 2014 [Ble14a]). The goal of this research is to produce a 3D model of the Pellerhaus in its historic state from the year 1605 by utilizing point clouds generated via laser scanning as described above.

1.3 Project schedule

This project is divided into two main phases. The first phase is developing the software for converting laser scanner point clouds as 3D panorama meshes. The second stage is designing the historic 3D model from this initial mesh.

This is visualized in the following GANTT chart:



1.4 State-of-the-art methods for 3D reconstruction

There are several methods that allow for the generation of 3D meshes from various data. One can either use several still images or videos, sample the real world with modern sensor technology, or use open data for generating geometry of varying complexity. This is described as follows:

1.4.1 Light Detection And Ranging (LiDAR)

The term Light Detection And Ranging (in short, LiDAR) is commonly used with high precision applications, such as scanning and mapping of indoor and outdoor environments. It uses a laser beam emitter and receiver. By using the Distance-Speed-Time formula it is very easy to compute how far away an object is:

$$speed = \frac{distance}{time} \iff distance = time * speed$$

The time between sending a signal and receiving it is measured and multiplied by the speed of light ($c = 299,792,458\ \frac{m}{s}$, see Wikipedia 2015 [Wik15b]). This returns the meters the light traveled from the emitter to the obstacle and back. Dividing this distance by two yields the range to the obstacle in meters (see Schroeder 2014 [Sch14]).

As this gives the meters to only one specific point, it is necessary to keep measuring from different view orientations. This can be done by rotating the scanning device horizontally and vertically simultaneously. To avoid mechanical imprecision or cables from becoming tangled, most devices usually use one motor for the horizontal and another motor to control a flat mirror on an elliptical mount for the vertical rotation. That way it is possible to sample a lot of points around the device position quickly and effectively.

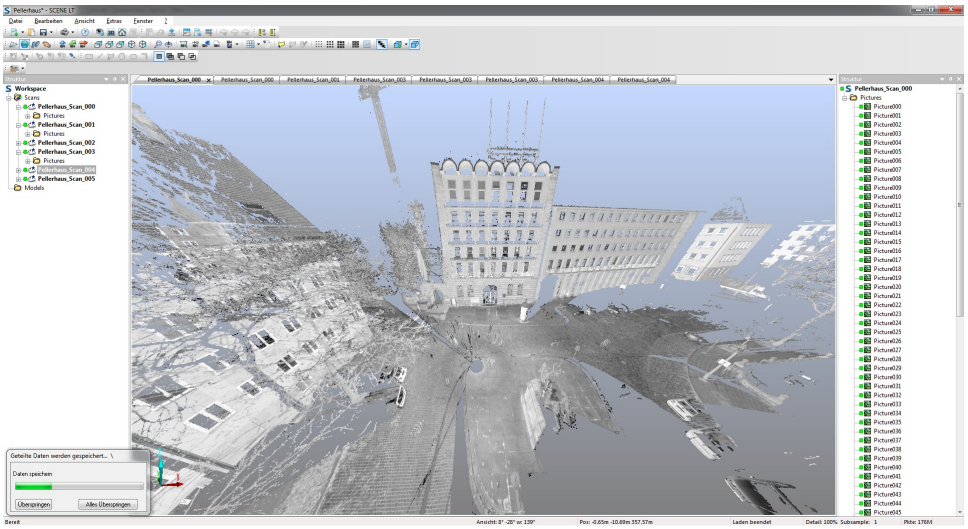


Figure 1.1: LiDAR Scanner Point Cloud of the Pellerhaus

This work utilized the LiDAR scanner FARO Focus^{3D}. It is capable of capturing 976,000 points per second with a vertical and horizontal field of view of 305 and

360 degrees, respectively (see Techsheet FARO Focus^{3D} 2013, [FAR13]). To allow a better registration, additional sensors can be used, such as GPS for localization and a barometer for height measurement. The measured points can be colored using an automatic color overlay from a built-in camera at a 70 megapixel resolution. The price for the Focus^{3D} totals at 61,404.37 Euro (see Opti-cal Survey Equipment Ltd. 2015 [Opt15]).

Besides using a stationary device, portable devices are also available. Recently, a new technology has been revealed by Csiro called *Zebedee*. This handheld laser scanner can be used in challenging environments where a stationary device would require several scans to cover the whole area (e.g. caves, staircases) while the operator is walking. It samples over 40,000 range measurements every second and consists of a 2D laser scanner mounted on a spring system (see Mail Online 2014, [Vic14]). The visual effects field in particular has a great use for this device, since the environments can vary a lot during video shootings and a 3D mesh representation is ubiquitous today. The price for the ZEB1 handheld laser scanner is 17,000 Euro[2].

Although measuring with laser technology can be found in household devices as an alternative for tape measuring, it is still quite complicated to reverse engineer such devices to get the raw distance reading. Fortunately a group of engineers tried to bridge the gap by starting a crowd funding campaign for a low-cost laser range finder, called the LiDAR-Lite (see PulsedLight 2015 [Pul15]). It has a total range of 40 meters with a resolution of 1 cm. This research utilizes this sensor with a custom arduino build to examine how it can be used as a cheap alternative to the examples mentioned initially. The price for one module is 82 Euro.

1.4.2 Ultrasonic

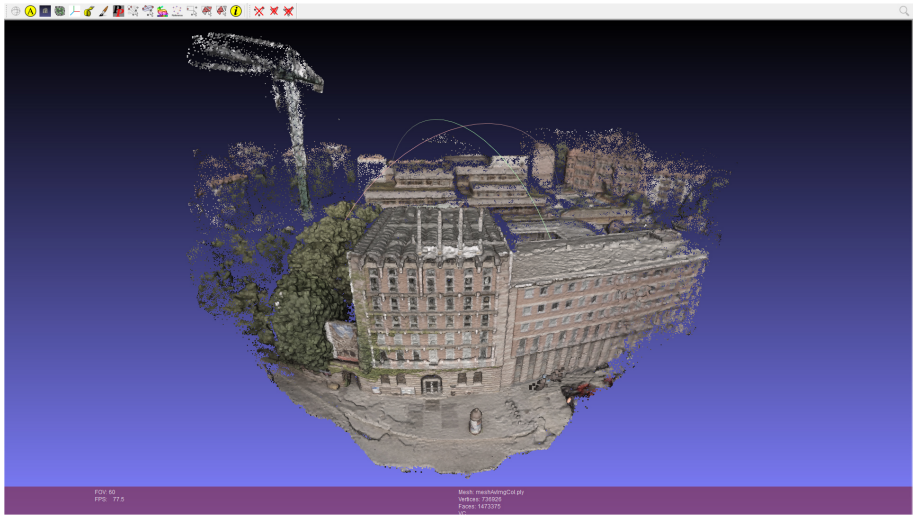
In contrast to LiDAR, most ultrasonic sensors are cheap, but generally are not used for higher distances at several tens of meters (though there are products for a range higher than 100 meters; see VEGAPULS 69 [VEG15]). The reason for this is that sound is usually affected more strongly by environmental properties than light (see Sensors Magazine 2015 [Sen15]). For this reason ultrasonic sensors are often used for shorter distances e.g. for near field obstacle recognition in robotics or in small desktop 3D scanners (see Dinh 2013 [Huy13]). Typical ultrasonic sensor modules with a maximum range of around 5 m can be purchased for just 5 Euro.

1.4.3 Photogrammetry

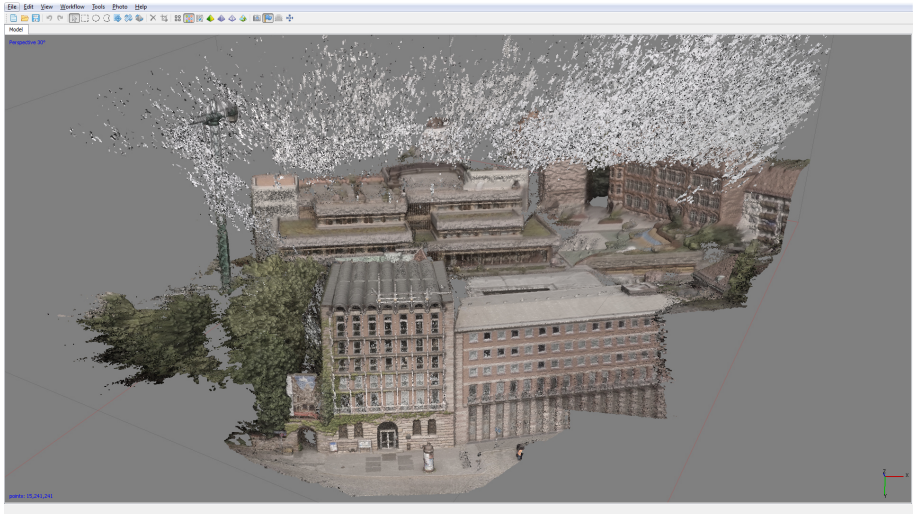
Photogrammetry (also referred to as multi-view reconstruction) is a technique from the Computer Vision field that presents a cost-effective alternative to laser scanning. A real 3D object can be reconstructed as a virtual 3D model by using photographs of the scene and feeding them into the software. This works by detecting image features (for example by using Harris Corner Detector or SIFT algorithms), matching those between image pairs, computing the respective camera positions and re-projecting the reconstructed 3D points to get a point cloud representation of the real photograph (see Solem 2012 [Sol12, p29]). The Computer Vision algorithms get better each day and there is plenty of software that uses them. Basically we can distinguish between open source, free, or commercial software for this task. Usually open source

[2]Source: Personal contact to sales team

software is free to use, too. However, it might have some limitations defined by its license, e.g. only granting non-commercial use. On the contrary, some licenses even allow users to sell the software under a different brand name as is the case with Blender. In this example the GNU GPL Version 3 license allows a company to sell Blender with prices starting at $ 47.00. As this is only a side note, more information on that topic can be found in [Ble15a]. To compare the results of open source and commercial photogrammetry software we processed 356 photos of the Pellerhaus with two applications. One trial used VisualSFM for generating a sparse point cloud in conjunction with CMP-MVS for the dense point cloud generation via open source tools. The other trial used the commercial software Agisoft Photoscan Professional which costs $ 3,499.00 but can be tested with a fully functional 30 day trial, as was done in this research.



(a) Open Source: Visual SFM + CMP-MVS (736,926 points)



(b) Commercial: Agisoft Photoscan (15,241,241 points)

Figure 1.2: Multi-view reconstruction point clouds generated from 356 photos

Comparing the results, the point cloud output from Agisoft Photoscan is much more detailed, approximately by a factor of 20 (see Figure 1.2). Furthermore VisualSFM created a bent facade, whereas Photoscan preserved all important straight lines. All in all it can be observed that the algorithms of Photoscan are more sophis-

ticated and suited better for images taken with a great amount of lens distortion, though this is something that should be avoided when considering using the footage for multi-view reconstruction (see Balletti et al. 2014 [Bal+14]). Both applications generate a model that can provide a good initial mesh of a scene, but the computation takes a significant amount of time. Photoscan used all resources of an eight core Intel i7 workstation with 16 GB of RAM running for about 4 days.

Photogrammetry will be used in this project to try reconstructing surfaces from historical images. Fortunately historical stereographic image pairs are provided through the Altstadtfreunde Nürnberg e.V. By matching the laser scanner data with the Photogrammetry output, a good groundwork is expected for the final surface reconstruction.

1.4.4 Depth Cameras

Instead of using photogrammetry software to retrieve 3D information from images, Depth Cameras can be used, which encompass the same functionality in hardware. Popular devices are the Microsoft Kinect v1 and v2 or the Asus Xtion Pro Live, typically ranging between 100 and 200 Euro. Using stereo matching algorithms those devices can determine the distance, or depth, of a certain point. First, an infrared projector emits a speckle pattern which an infrared camera analyzes to match points between the emitter and camera. By using a mathematical process based on trigonometry called Triangulation (see Wikipedia 2015 [Wik15c]) it is possible to calculate the distance to a point if certrain properties are known, such as the distance of a fixed baseline between two observing points and the angle from the baseline to the observed point. There are some problems known with those sensors which limit their use to mostly indoor applications. Direct sunlight can wash out the speckle pattern or multiple sensors can confuse each other. Despite those issues, Depth Cameras provide a simple and fast way to get 3D point clouds of real objects. Custom software can be written to access this data directly from the depth sensor. The Microsoft Kinect SDK provides some examples of how this can be accomplished, and the Kinect Fusion project presents a complete solution for creating 3D surfaces of high resolution in real-time (see Newcombe et al. 2011 [New+11]).

1.4.5 Google Maps ®

The commercial application allows viewing cities from the sky with a rough representation of 3D building shapes (see Zamora 2014 [Ric14]). While this service had gray boxes some years ago, today the visualization is getting more accurate. Nowadays it is possible to see small details with better modeled and textured buildings.

1.4.6 Open Street Map ®

The open source alternative to the commercial service above offers the basic functions for map viewing and navigation. OpenStreetMap (OSM) offers very detailed access to its data, like boundaries, streets and building footprints. That way it is possible to extract simple building shapes (see F4 2014 [F414]) that can be used in custom software free of charge.

To allow for a better mapping of buildings there are also proposals for an indoor version of OSM (see OpenStreetMap Wiki 2015 [Ope15]). Having this data available

is a helpful asset for various applications such as indoor navigation at railway and subway stations, mobile emergency exit information and robotics.

1.4.7 Bavarian State Office for Survey and Geoinformation

Geodata and city plans are usually provided officially through governmental institutions. They provide various types of data, including historical aerial photographs, digital elevation models (DEM) and also 3D building shapes. For educational purposes (like this research) they offer a university discount for the data of 25 percent. A usual dataset without any discounts containing 7580 buildings of Langwasser, district of Nuremberg in Germany, costs 1,158 Euro[3].

1.4.8 Autonomous mapping with UAV's and SLAM

Drones, or unmanned aerial vehicles (UAV's), are getting more popular each day. Most of them are also equipped with a camera which allows for taking pictures or videos from viewpoints a human cannot reach easily. More expensive drones have LiDAR systems attached (Shen et. al. 2010 [She10]) which allow, together with the IMU (Inertial Measuring Unit) and GPS (Global Positioning System), to localize it and map its environment. A popular term for retrieving the current position based on various sensor data while creating a virtual representation of the environment at the same time is Simultaneous Localization And Mapping (SLAM).

1.4.9 Manual methods

If all other methods fail, there is still the chance to get a reconstruction done roughly by taking measurements of real objects with measuring tapes or eyeballing. Loading reference pictures from the front, side and top view into a 3D software can already yield decent results. Furthermore, this is the standard way a 3D artist would begin to model a digital human or character that a concept artist provided by sketching those three main views. Concept art that is accurate and matches every other view can help a 3D artist to block out the shapes very quickly. In certain circumstances it can even be faster than setting up a scanning environment or generating a mesh via photogrammetry, because the generated meshes need to be retopolgized (that is, re-modelled with a strong focus on the clean layout of the mesh grid) as soon as they are considered for use, for example in animation.

1.5 Defining the scope of this research

Although this work uses a combination of several techniques (briefly presented above), the main focus is on examination of whether panoramic projection and meshing of laser scanner point clouds would be an aid for 3D reconstruction. This will be evaluated by using the result from the custom converter software in a real world use case of using the generated mesh in the design process.

[3]Personal research and contact

Chapter 2

Background Research

Before we can discuss how the Pellerhaus will be reconstructed by the projection of point clouds as 3D panoramas, it is necessary to build up some fundamental knowledge about the history of the Pellerhaus, and define what a 3D panorama is and what properties it has. Those fundamentals will enable us to start with the creation of software tools or 3D models that build upon this basic information. First, we establish a brief historic review of the art epoch in which the Pellerhaus was built to understand the underlying principles of the way it was designed and what it communicated by its authentic style. Secondly, this report presents a definition of the term "3D panorama" and explains how such a panorama is created from a high-level perspective which prepares the reader for the low-level details in the next chapter.

2.1 Historical fundamentals

2.1.1 Renaissance

We can group certain historical time periods by name. For example, the time period between the years 400 to 1499 (often referred to as the 5th and 15th century, respectively) are called the Middle Ages. The time period after the year 1500 (or in other words after the 16th century) is known as Modern History. We want to take a closer look inbetween those two time spans, namely the period from the 14th to the 17th century. This is where the Renaissance art epoch was active and the Pellerhaus Nürnberg was built. Renaissance literally means "rebirth" when translated from French. In the Late Medieval period the Renaissance started as a cultural movement in Italy first, and subsequently spread across the rest of Europe. Because it is located between the aforementioned time epochs it is considered to be the bridge between them. The start of the Renaissance in Italy was fostered by the support for artistic movements from powerful and dominant families like the Medici, comparable to today's art patronage of digital artists by larger companies (see Voss 2014 [Geo14] in The Guardian). The Medici family in Florence poineered the banking system and therefore introduced a commercial revolution to finance the Renaissance.

The Fall of the Constantinople in 1453 at the hands of the Ottoman Turks caused Greek scholars to migrate towards the west. When they arrived in Italy they spread their wisdom and knowledge of ancient Greece and Rome through all the major city states across the Italian peninsula, such as Florence, Venice, Milan, and Rome,

during the Renaissance papacy. Thus, the influence of the Renaissance affected the questioning of many aspects in life, like literature, philosophy, art, music, politics, science and religion. Scholars established new methods of study and introduced realism and human emotion in art.

The principles expressed by the Renaissance are a cultural revival of the ones developed in the ancient Greek and Roman Empires, where for instance in architecture the most representative building was said to be a temple (see Wikipedia 2015 [Wik15d], Architecture section). Since these principles were not applied uniformly all over Europe, widespread educational reform was gradual, though it had a major impact in all aspects of life. In politics it was the base of the conventions of diplomacy, and in science the renaissance brought an increased reliance on observing nature instead of pure superstition. The greatest impact, however, was on arts, with discoveries made by famous artists like Leonardo da Vinci and Michelangelo. To summarize, the Renaissance could be considered as an attempt to study and improve the physical world by reviving ancient ideas and principles on the one hand and new approaches to thoughts on the other hand.

This revival based on ancient Greek and Roman culture is observed in Renaissance Architecture as well. It followed Gothic architecture and was succeeded by Baroque architecture. The typical Renaissance style in architecture is described by Wikipedia [Wik15e] as follows:

> Renaissance style places emphasis on symmetry, proportion, geometry and the regularity of parts as they are demonstrated in the architecture of classical antiquity and in particular ancient Roman architecture, of which many examples remained. Orderly arrangements of columns, pilasters and lintels, as well as the use of semicircular arches, hemispherical domes, niches and aedicules replaced the more complex proportional systems and irregular profiles of medieval buildings.

Renaissance Architecture in Germany

The arrival of Renaissance architecture in Germany was inspired by German philosophers and artists such as Albrecht Dürer and Johannes Reuchlin. They travelled to Italy, where they learned more about its advanced artistic world. Although the religious turmoil caused by the Protestant Reformation was frequently depicted in art or literature, gothic and medieval scholastic philosophy remained dominant until the end of the 15th century. When Emperor Maximilian I of Habsburg (1493-1519) rose to power, the Renaissance became popular in Germany, too. Maximilian I was the first true Renaissance monarch of the Holy Roman Empire.

There are several examples of early Renaissance architecture in Germany, such as the Landshut Residence or the Augsburg Town Hall, both located in the Free State of Bavaria in the south of Germany. Furthermore the largest Renaissance church was also built in Bavaria. Sir Wiliam V, Duke of Bavaria, had to raze 87 houses, ignoring the protests of the citizens, to construct the St. Michael's Church in Munich between 1583 and 1597.

Along the Weser river in central Germany a specific regional variant of architectural style called the "Weser Renaissance" has been preserved in unusually high density in towns and cities of that region. This was caused by the slow economic

recovery of the region from the effects of the Thirty Years War (1618 - 1648) which made them unable to transform their architecture to baroque style (see Wikipedia 2015 [Wik15f]).

Nuremberg had substantial achievements in the field of architecture around 1600 as well. Some of the most popular projects are described by Mährle (2000) in his book "Academia Norica" [Mäh00] which is translated by the author from German as follows:

> Significant public and private buildings have been built between the end of the Second Margrave War and the beginning of the Thirty Years' War.
>
> The first big construction project after the end of the war against Albrecht Alcibiades was the fortification of the defense structures. During 1556-1564, the wall ring was improved and the towers of the five main gates (Laufer Tor, Spittlertor, Frauentor, Neutor, Vestnertor) were surrounded by a stone wall. This was inspired by the towers of Castle Sforza in Milan, Italy.
>
> Additional important public buildings were realized by the city builder Jacob Wolff der Ältere (1596-1612) and his son Jacob Wolff der Jüngere (1612-1620) during that time. The most important ones have been the construction of the Fleischbrücke inspired by the Ponte Rialto in Venice (after 1596), the Wöhrder Torbastei (1613/1614), the master builders' house on the Peunt (1615) and especially the city hall, which was inspired by late renaissance style palaces in Italy (1616-1622).
>
> Besides the public buildings there were several considerable private structures created around 1600. They mostly haven't been commissioned by patricians but rich merchants. The most important ones have been the Toplerhaus (1590), the Fembohaus (1591) and the Pellerhaus (1602-1607).
>
> At the same time many manors in the land domain of Nuremberg have been rebuilt in the following decades after the Second Margrave War.

2.1.2 Pellerhaus

The Pellerhaus was built between 1602 and 1605 and considered one of the most magnificient examples of a town house in German Renaissance architecture.

The house was commissioned to be constructed by the wealthy trading company Viatis-Peller, which was in the posession of the greatest assets at that time. Bartholomäus Viatis gave the Pellerhaus to his son-in-law Martin Peller where it remained in the posession of the Peller family until 1828.

Credit: Altstadtfreunde Nürnberg e.V.

Figure 2.1: Pellerhaus around 1905

The house changed owners several times during the subsequent 100 years until it was purchased by the mayor of Nuremberg Hermann Luppe in 1929. Acquring the house assured a proper maintenance of this historic landmark by the city. It was estimated that a full reconstruction would need the budget of a Martin Peller to succeed. Nuremberg felt responsible for maintaining the beauty of the Pellerhaus at that time and financed a restoration of critical parts of the building. Hence, it started a refurbishment program between 1931 and 1934 for the Pellerhaus, where the focus was put on restoring the delightfulness of the court yard and the rear facade. The red facades have been cleared up and new stone details have been redone by hand. Artists were careful to keep all of the small details and not to recreate the house in a more modern style. The Pellerhaus was saved. Luckily it was at that time that a vast documentation of the historic Pellerhaus was created. This 1930's restoration is incredible worthy today. Hundreds of plans and photos document every detail of its facade. Without that documentation a reconstruction would have been extremely difficult today (see Pellerhaus Magazin 2013 [Alt13]).

Unfortunately, Nuremberg suffered from an Allied aerial bombing on January 2nd, 1945, as a result of World War II. It was the most severe attack made by aerial bombardment at that time. Nearly 1,800 people were killed that day. This Area Bombing Directive was issued by the British Air Ministry which directed the Royal Air Force to concentrate their attacks on factories and industry buildings in general. The objective of the directive was to focus the attacks on the enemy morale. Hence, the bombardment is often called „Morale Bombing“. Many buildings were transformed into a leveled surface after the removal of the debris remaining from the attack. The bombed areas today present either completely new buildings or reconstructed ones. The Pellerhaus is a building that has been reconstructed after World War II.

Figure 2.2: Pellerhaus 2015

The reconstruction was initiated in 1955. It only preserved the base floor in its historic state, due to the fact that major parts of it survived the bombing. From the first floor upwards, the facade of the Pellerhaus changed dramatically and only served a pure functional purpose, as room was needed to accomodate the new City Library and Archive. It was almost decided to completely embed the Pellerhaus into the Library which had been built to the right of it, but a movement within the city prevented that. An old arch was completely destroyed to connect the Pellerhaus with the Library, which was criticized as being senseless. Only some column bases and capitals remained in the inner courtyard, ready to be build into the southern part of the court. So all six arcs, the small passage next to the front-facing house and the adjacent facade part of the northern court facade needed to be recreated. In the end, the old style Pellerhaus was combined with a new style to allow at least some experience of the old state. The reconstruction of the base floor was finished in 1957. The upper floors - and therefore all reconstruction efforts - were finished by 1960. At that time people realized that a full reconstruction might happen some day, although the individual storey heights differ from the original. Conversely, when a secondary school was built on top of the back-facing house in 1972/73, almost any hope of a full reconstruction of the Pellerhaus was dashed (see Pellerhaus Magazin 2012 [Alt12]).

Credit: Carl Simon United Archives

(a) before 1945

Credit: Life Archive

(b) Pellerhaus 1945

(c) after 1945

Figure 2.3: The Evolution of the Pellerhaus

In 2005 a new initiative was launched, which had the goal of reconstructing the Pellerhof, which is the inner courtyard of the Pellerhaus. With its groined vaults, it is an artistic piece of architecture. The association Altstadtfreunde e.V. created a flyer [Alt] which includes a wonderful description of the Pellerhaus that has been

translated by the author from German:

> Before destruction, the Pellerhaus was one of the main sights of Nuremberg. The architecture seems to be the most honorable performance of the local art of construction. Its inner court was considered the probably most beautiful arcade court.
> As the city descended into shatters in 1945, there were only a few remains of the Pellerhaus. The front-facing house was rebuilt in a modern form 1957 on top of the reconstructed hall. An enourmous effort was done by complementing the courtyard, it was discontinued 1959, though.
> Not until 2005, 60 years after the destruction, the Altstadtfreunde took the initiative to continue the former abandoned construction of side wing and rear house facade.
> With the accurate documentation of the pre-war level it is possible to do those court additions with extraordinary accuracy. October 2008 layed the foundation block of building the courtyard completely via donations. Since then with the well corner, side wings and eastern backyard gallery crucial parts have been able to get restored from the old building.
> With your donation or by purchasing a symbolic block of stone you can help to make one of the greatest achievements of German Renaissance come alive in its historic state.
> At the time when the merchant Martin Peller started with building his house in 1602, he also layed the foundation block to what later entered as the most magnificient bourgeois house into the history of art. The notion of building an arcade court was not new in Nuremberg. There have been hundreds of gallery courts in the city. Many of them with tracery breastwork made of stone. Though, the Pellerish courtyard bested everything that has been known at that time:
> On the two long sides it was flanked by noble three-story arcades, with a clear and symmetric structure, though with a rich and filigreed ornamentation. While skimming along it, ultimately the show façade caught the eye with a glorious gable. Seldom one can find forms of the italian renaissance merged with local sensuous enjoyment in such a happy way. Antique style pillars accompany the individual floors, obelisks stretch up into the sky and still the appearance was entirely different than in Italy. The Pellerhof, as a Middle European counterpart to the wonderful arcade courts of Italy, is an indispensable part of european architecture.

This project is still active today and the Pellerhof has almost been fully reconstructed by the Altstadtfreunde Nürnberg e.V. at the time of this writing. The build process was mainly based on photos and the remains of the western side. Measurements have been extracted by examining the remains including profiles, design of capitals and ending stones. Overall forms were reconstruced with the help of historic photos. New constructions were needed for the differing tracery of balustrade areas. It was a stroke of luck that the historic documentation of the house is extensive. This helped reconcile differences of geometrically correct constructions with the new build. The Chörleins (a type of bay window) are also documented well enough to allow for a reconstruction. For example, there was an incorrect ornamentation of

Chörleins at window lintels, sockets, and volutes when comparing the rebuild from 1950 with the original. The new build comes much closer to the Renaissance original. The Altstadtfreunde Nürnberg e.V. can proudly say that, with their restored state, the two time layers 1605/07 and 1957/59 form a harmonic unit. In April 2013 they moved newly produced stone blocks and a fully donated arc into the Pellerhaus. Once again, they noticed reckless deviation from consistancy. All of the six arcs have different spans and the alignment of the arcarde row is not straight, but has been – in its old parts – slightly bulged out. Though this might be due to the bombing destruction, the fact that the arc row doesn't continue horizontally but considerably descends from the front-facing house into the courtyard is notable.

Many of the facades of the buildings in Nuremberg have been painted red with white rectangles. The reason for this was that the look of mined stones varied considerably, so by painting them the houses had a united look. This color is also called the „Nürnberger Rot" (Terra Norimbergensis rubra), because it looks like the local sand stone and the color powder comes from the rural area of Nuremberg. Unfortunately only a few color remains are left but it is enough to prove the colorfulness of the facade. After finishing the reconstruction in the courtyard, repainting the facades in the „Nürnberger Rot" would be the right decision (see Pellerhaus Magazin 2014 [Alt14]).

With the ongoing progress of the reconstruction of the Pellerhof, our research will examine ways to reconstruct the historic Pellerhaus facade as a 3D model.

2.1.3 Architects of the Pellerhaus Nürnberg

The Pellerhaus was constructed by the German architect Jakob Wolff der Ältere who was born in Bamberg in 1546. He accomplished several famous building projects in Nuremberg, such as the Fleischbrücke. He had two sons, Hans and Jakob Wolff der Jüngere. They were educated by their father and built the city hall of Nuremberg.

A detailed biography about the architects can be found in the Appendix (A.1).

2.2 3D Panorama

The term "panorama" is defined by Wikipedia [Wik15g] as follows:

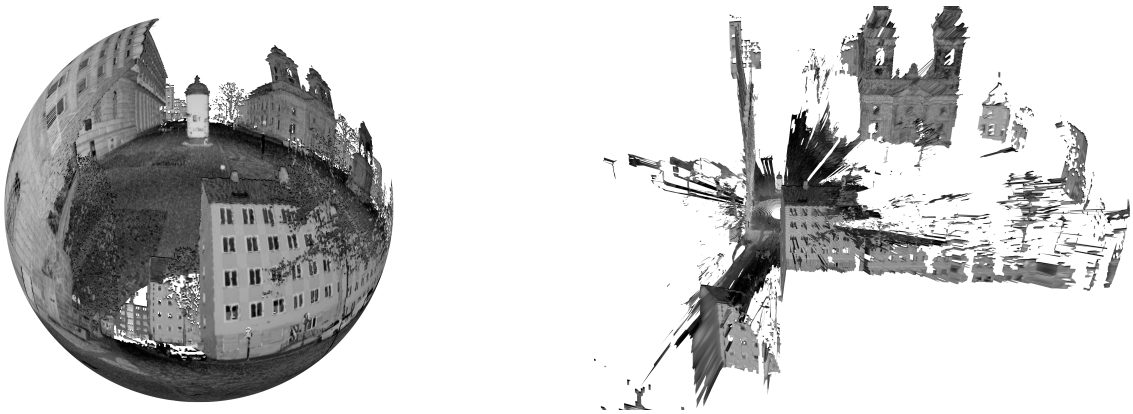
> A panorama (formed from Greek $\pi\tilde{\alpha}\nu$ "all" + $\ddot{o}\rho\alpha\mu\alpha$ "sight"), is any wide-angle view or representation of a physical space, whether in painting, drawing, photography, film, seismic images or a three-dimensional model. The word was originally coined in the 18th century[1] by the Irish painter Robert Barker to describe his panoramic paintings of Edinburgh and London. The motion-picture term panning is derived from panorama.

In simple words this means that we can create a two-dimensional image containing every view one can see from a fixed position. Below is an example how this could look for the Pellerhaus (see Figure 2.4):

Figure 2.4: 2D Panorama of the Pellerhaus

The image above was generated with the custom PointCloud2Blender (PC2B) converter software which will be discussed in detail in the next chapter. A two-dimensional panorama image depicts a full horizontal and vertical view of the location where it was taken. Please note that the example above is not a full vertical view, the reason being that this image was generated from a point cloud created by a laser scanner. Due to the tripod on which the scanner was mounted, the lower part of the scan is useless and therefore has been omitted by the scanning device. Additionally panorama images generated in PC2B are flipped horizontally (see Chapter 3.4.4).

This two-dimensional panorama can now be converted to a three-dimensional panorama by simply mapping it onto a sphere, as shown here:



(a) 2D Panorama mapped on a unit sphere

(b) The same panorama mapped on a sphere using individual distances

Figure 2.5: Mapping a two-dimensional panorama onto a three-dimensional sphere

Placing a virtual camera in the center of the sphere makes it possible to visualize the complete three dimensional environment from one viewpoint.

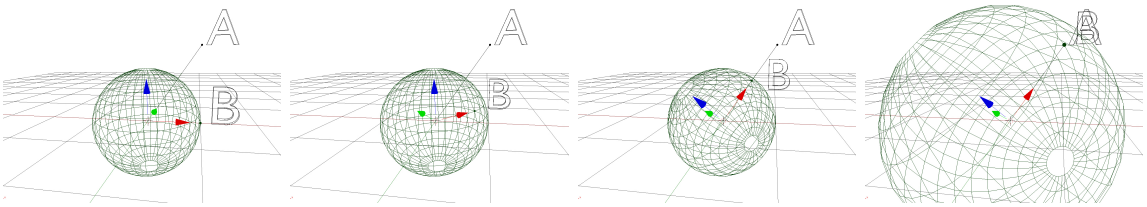
3D Panoramas were given a great new use by the introduction of the Oculus Rift (Oculus VR, [Ocu15]), and are increasingly used in film production as well. While

this technique was used rather seldom in, for example, the Circle-Vision theaters in Walt Disney Theme Parks in 1955 (see Wikipedia 2015 [Wik15h]), today's hardware is becoming cheaper and faster to create such panoramas in real-time. The German FMX 2015, one of the biggest international conferences on animation, effects, games and transmedia, presented several software and hardware solutions that were able to stitch multiple video feeds into one panorama (see Kolor 2015 [Kol15]). A very sophisticated example of using video panoramas combined with visual effects is Google ATAP 'HELP' (see CGMeetup 2015, [CGM15]), which is a video that can be viewed interactively on mobile devices from every angle by simply pointing the device in the desired direction.

2.2.1 Creating panoramas

To create panoramic images it is necessary to map a three-dimensional environment onto a two-dimensional plane. This is accomplished by computing the spherical coordinates of every point in three-dimensional space. Without going into too much technical detail, we can either define a point in 3D space with its x, y and z coordinate (known as the cartesian coordinates) or with their horizontal angle displacement, vertical angle displacement and distance from the origin (which is known as spherical coordinates). The latter can be imagined as rotating and scaling a unit sphere until it intersects the point.

To clarify the process a little further, we provide a visual example. Consider the 3D point A at $(1.0, 1.0, 1.41421)$ in cartesian coordinates (x, y, z). We need to align the x-axis of a unit sphere, labeled Point B, with point A (Figure 2.6a). To do so, we first rotate the sphere along its local z-axis by 45 degrees (Figure 2.6b). Secondly, we rotate it along its local y-axis by 45 degrees (Figure 2.6c). Finally we scale the unit sphere by 2 to align both points with each other (Figure 2.6d). Now we have calculated the spherical coordinates $(\theta, \varphi, radius)$ which are expressed as $(45.0°, 45.0°, 2.0)$ in degrees or as $(\frac{\pi}{4}, \frac{\pi}{4}, 2.0)$ in radians. This process is visualized below:



(a) (b) (c) (d)

Figure 2.6: Converting from cartesian to spherical coordinates, visually

The final image can be formed by mapping these spherical coordinates $(\theta, \varphi, radius)$ to the image coordinates (x, y). This is called panoramic projection.

There is a set of various types of projections used for panorama generation. A subset of them is introduced in the following section.

2.2.2 Types of projections

It is not possible to create a perfectly "flat" or two-dimensional representation of a sphere. There will always be distorted areas; thus it is necessary to choose the right

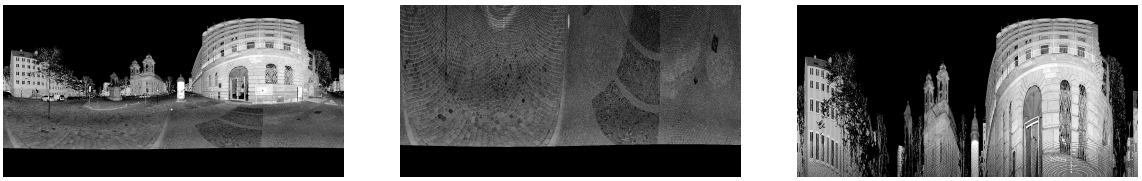
projection type for every new task (see Furuti 2014 [Car14]).

A study of implementing seven different projections was conducted by Houshiar et al. [Hou+15] in 2015. The findings of that study are the basis for our implementation in the converter software. We have implemented three projections, namely the equirectangular, cylindrical and mercator projections.

First, the equirectangular projection is the simplest type and can be implemented very quickly, as the spherical coordinates θ and φ are mapped directly to the x and y coordinates of the two-dimensional image without any transformation.

Second, the cylindrical projection is similar, but the vertical mapping is transformed. This can be envisioned by placing a sphere inside a cylinder. If light is emitted from the center of the sphere it is projected onto the cylinder. The projection keeps vertical lines straight, but horizontal lines are still curved. Furthermore, objects are stretched vertically, which becomes more prominent the closer they are to the north and south poles of the sphere.

Lastly, the mercator projection is a mapping that preserves angles locally (conformal projection). The distortions are less pronounced than equirectangular or cylindrical projections.



(a) Equirectangular (b) Cylindrical (c) Mercator

Figure 2.7: Three sample projection types used in PC2B

We have observed that the equirectangular projection is suited best for our purpose (see Figure 2.7a). It is one of the most widely spread types and provides full coverage of the scene. Furthermore, it does not apply any transformation or scaling. During testing it proved to be the only useful projection, since the other projections caused many points to stretch into negative infinity. For this reason we used primarily equirectangular projection in this project.

Chapter 3

Conversion: From point cloud to Blender 3D

This chapter presents the implementation of a custom software tool, which creates panoramic images based on point clouds captured, for example, via laser scanning. We discuss every step in the development phase from concept to finished prototype. It is developed with the version control system git and hosted in a public repository on GitHub.com. The software development progress is recorded via screen capturing and will gradually be uploaded online. Additionally, this information is provided on the official website which accompanies our research.

3.1 Concept and preparation

Based on the initial idea to somehow move from a dense point cloud to a 3D mesh surface that can be used in the 3D graphic suite Blender, it was important to plan ahead.
It is possible to mesh a 3D point cloud with several algorithms by, for example, trying to find the nearest neighbour of a point in 3D space. One such algorithm is called Delaunay Tetrahedralization (see Shewchuk 2002 [She02]) and is used in the free multi-view reconstruction software "Visual SFM" (see Appendix A.2.5), for instance.
With our method we try to utilize the characteristics of laser scanners in such a way that we know every aquired point can be described by scaling and rotating a unit sphere. In mathematical terms we can determine the spherical coordinates of every point in our point cloud. The 3D points need to be converted from their cartesian coordinate system to the spherical coordinate system first (see Wikipedia 2015 [Wik15i]). Using this simple principle we can not only mesh a point cloud generated by a laser scanner, we can texture it, too. With the coordinates ranging from 0 to 360 degrees horizontally, 0 to 180 degrees vertically and a depth coordinate ranging from 0 to the maximum scan distance we are able to create two images, namely a depth map and a color map. Those images are then used to create a regular grid which is used for meshing and texture coordinates. By applying the inverse transform to the spherical coordinates it is possible to get the vertices of the textured 3D mesh in cartesian space and export it to any 3D file format. This process is explained in more detail in this chapter.
Furthermore it was neccessary to know how the user will be operating with the

software. Usually a use case diagram is created to determine the required functions the software must provide in order to let a user accomplish his or her desired goals.

3.1.1 Use case diagram

The use of the converter software should be easy and fast. A user needs to do at least two actions, namely load a point cloud file and easily start the conversion process. Users should be able to adjust various settings for the file import, to meet the needs for their specific project. The initial use case diagram for this software looks like this:

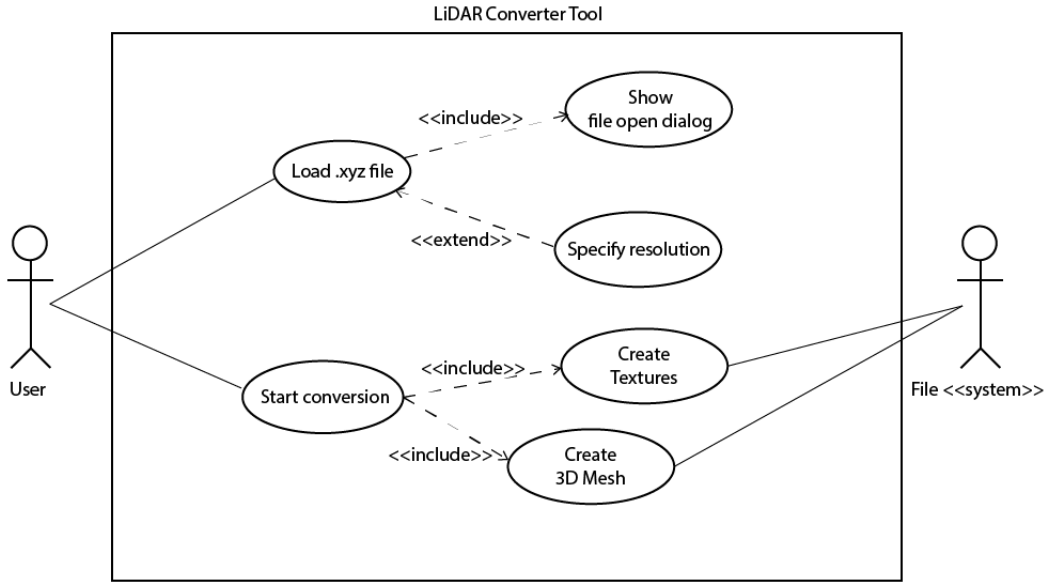


Figure 3.1: Use Case Diagram

3.1.2 Laser scanning on location

For data aquisition we used the FARO Focus^{3D} laser scanner on January 21st 2015. First, the device should be configured by setting the desired resolution (both for scan and photos), maximum scan distance (which results in a change of the eye-safety distance), a project name and GPS location (if no GPS module is available, as in our case). This configuration can be done in the office or on-site. After the device is set up, the scan process is started. During scanning, the main body of the device rotates horizontally and a mirror mounted inside the body rotates vertically. This creates the uncolored point cloud. After the scanning process, several pictures are taken by the built-in camera to color the point cloud. The scanner also measures the exposure to avoid under- or overexposured photographs. Finally, the inclination is measured with an inclinometer to level the point cloud properly. Scanning took about 40 minutes. This procedure was repeated five times to get additional scans covering viewpoints that have been obstructed by obstacles.

During scanning we faced problems we didn't expect. We encountered people walking or stopping in the laser beam (resulting in vertical lines in the final scan), a

suprising crash of the device's operating system leading to a terminal output window (wrecking the sd card with all previous scans), and a man asking if we took a photo of him while he was entering the building (obviously blocking the scanner while talking to us).

Figure 3.2: Scanning with FARO Focus^{3D}

3.2 Generating data and testing algorithms

3.2.1 BlenSor

The Blender Sensor Simulation Toolbox (see Gschwandtner et al. 2011 [Gsc+11]) is a custom version of the open source software Blender that allows simulations of different ways of scanning within a virtual 3D scene. It is being developed by the Department of Computer Sciences, University of Salzburg, Austria. The goal of this project is to provide a tool, mostly aimed at researchers, that can help with testing algorithms for fields such as obstacle detection and tracking, range data segmentation or surface reconstruction.
We found this software very useful to begin the development of PC2B. With a number of scanner presets it is possible to generate a point cloud of a virtual environment from different types of scanner devices.

3.2.2 Test-Add-on for Blender

During the beginning of the software development process the point cloud projection did not seem to be correct. Testing the algorithm responsible for projecting from cartesian to spherical coordinates was very tedious, because it involved importing the files, waiting for the images to get generated and then either looking at the produced image files to find mistakes or continuing with the meshing process. This pipeline was prone to errors, because one tiny mistake might affect the overall result. Hence, a custom add-on for Blender was developed to test the equirectangular projection algorithm for correct mathematics. The language used for add-ons is Python 3 and enables developing powerful extensions to the Blender core.

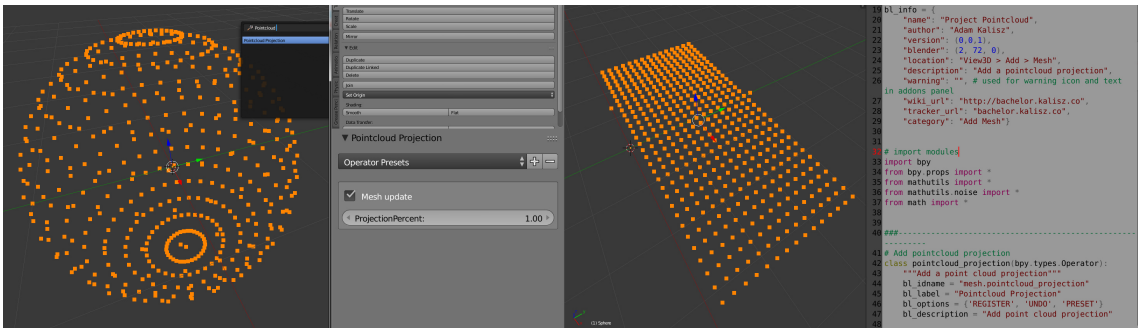


Figure 3.3: Custom Blender Addon

A more efficient and elegant approach would be to develop a new modifier in C/C++ that integrates directly into Blender and can perform mesh manipulation fast. Unfortunately diving into Blender Core Development is not very easy due to its huge code base. In addition the time constraint did not permit experimenting further with this approach. Overall, it was helpful for tweaking the algorithm.

3.3 How the conversion works

The working title of this converter software was defined as "PointCloud2Blender", PC2B in short, because converting point clouds to a Blender compatible file format was the main goal of the software project. The prototype consists of three main parts: the importer, the 3D panorama, and the mesher.

3.3.1 Pipeline overview

We provide a high level overview of how the conversion works. The individual conversion stages are explained in more detail in the following sections.

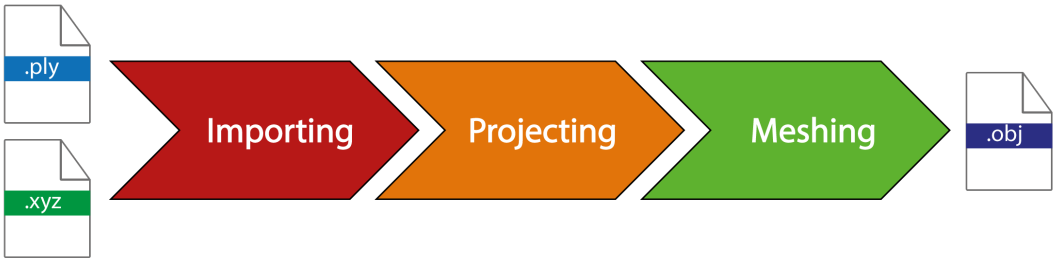


Figure 3.4: The stages of the conversion process

When point cloud files are provided by the user, they are imported into PC2B. That is, the individual lines of the files are parsed and split up into small components, such as x, y, and z coordinate information. This is done in a separate module as soon as the user starts the conversion. Subsequently the imported three-dimensional points are projected onto a two-dimensional grid in order to prepare them for meshing in the next stage. The two-dimensional points are used for the creation of the mesh surface which is exported as a file format that can be imported into Blender.

3.3.2 Point Cloud Importer

A crucial part of the PC2B converter software is the ability to import point clouds saved as files. There is a huge amount of file types that can accomodate such a data structure. Points can be stored in ASCII, Binary or a hybrid form of both. ASCII files are human-readable, while binary are not. Importing binary formats requires one to know the exact structure of the file and the byte lengths used for certain values. This is important to avoid mixing and misinterpretation of data. A precise import function needs to adhere to the exact file structure rules. Documentation was limited for many of the file formats, so using ASCII files was a better choice from the beginning. Initially, it was decided to only import the .xyz file format, since this is a very simple file format that can be exported in ASCII form from the proprietary FARO SCENE 5 software which is needed for preprocessing the raw point cloud stream produced by the FARO Focus^{3D}.
During development it turned out that support for the .ply file format is desirable, since scientific websites that provide models (see Stanford Computer Graphics Laboratory 2014 [Sta14]) widely provide this file type. Also Blender can export a 3D

model to this file format. This fact was extremely useful for testing the algorithm, because test data can be generated quickly.

Point Cloud data formats

Working with such file structures, as in our case, is very easy (See Figure 3.5).

```
-59.43620000 -31.36650000 302.80950000 59 46 55
-60.34600000 -31.80190000 302.85280000 58 45 54
-60.51810000 -31.88870000 302.71900000 58 45 54
-59.50470000 -31.39880000 302.68240000 56 47 50
-60.32350000 -31.79130000 302.89580000 59 46 54
-59.40940000 -31.35360000 302.85200000 61 48 57
-67.58220000 -29.73320000 302.12780000 73 55 58
-67.51000000 -29.59980000 302.04520000 63 43 54
-66.18880000 -29.78650000 299.88830000 100 87 80
-67.54620000 -29.66590000 302.08650000 64 47 56
-67.50660000 -29.59690000 302.00010000 62 43 51
-67.51970000 -29.60540000 302.09000000 64 46 57
...
```

(a) Sample .xyz file

```
ply
format ascii 1.0
comment VCGLIB generated
element vertex 2900882
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue
element face 1473375
property list uchar int vertex_indices
end_header
-37.037369 -0.850013 -1.831357 119 108 103
-37.063740 -0.755348 -1.886383 130 118 113
-37.446922 -0.902575 -2.421639 155 146 142
-37.512043 -1.051430 -2.535675 135 125 122
-37.546394 -0.904681 -2.498151 140 130 127
-37.340092 -0.932499 -2.274875 138 128 124
-37.292992 -0.973793 -2.336324 162 153 149
-37.166294 -0.780378 -2.111366 138 127 121
-37.341915 -0.864522 -2.316645 173 165 160
-36.981441 -0.691213 -1.978962 127 114 108
-36.918392 -0.767583 -1.994380 127 114 107
-37.038353 -0.931582 -2.136851 134 122 117
...
```

(b) Sample .ply file

Figure 3.5: Sample generated point cloud files

While the .xyz file type solely lists the x, y, z coordinates and optionally the red, green and blue color components, the .ply file type begins with a header describing how the file is structured. This has advantages, because we noticed that the .xyz file format is not documented. And to make things more complicated the free version of the proprietary FARO SCENE software, FARO SCENE LT, exports a different .xyz file than the paid version. With the free version two additional columns are added at the beginning of each row, such as the row and column number, and other settings can alter the file format even more. Those special cases are considered in the implementation of our prototype.

3.3.3 Determination of original point cloud resolution

Users of PC2B have the option to automatically determine the resolution of a panorama based on the point cloud file. The panorama resolution can either be set to fixed multiples of 360 by 180 pixels or set to a custom resolution, which can be filled out by the software. We have implemented an algorithm to help the user

find the best resolution for his particular point cloud. It works by creating a histogram for counting the number of fixed steps of horizontal angles. First, we define an angle accuracy Δw:

$$\Delta w = \frac{\frac{360}{30000}}{4}$$

We limit the maximum horizontal scan points to 30,000 which returns an angle accuracy of 0.003 degrees. The histogram is created with n values based on the accuracy:

$$n = \left\lceil \frac{360}{\Delta w} \right\rceil$$

For every scan point in the point cloud file, we compute its position (or index) in the histogram:

$$i = \left\lfloor \frac{w}{\Delta w} \right\rfloor$$

We then increment the histogram value at the index and repeat this procedure for every point in the point cloud file. The horizontal scan resolution is determined by counting the non-zero values in the histogram.

3.3.4 Coordinate system representations

We can express points in different coordinate systems, including cartesian, cylindrical and spherical.

Almost all point cloud files use a cartesian coordinate system (at least the ones we are using). To get the coordinate of a point in an image plane from a point in cartesian space, we simply convert its coordinate space from cartesian to spherical and project it onto the image plane. We then have the points horizontal position in a range of 360 degrees and its vertical position in a range of 180 degrees. In that way we create the image files from the point samples and then use those images to convert back to the cartesian space when creating the 3D mesh.

3.3.5 Converting from cartesian to spherical and vice versa

To convert from Cartesian space (x, y, z) to spherical coordinates $(\theta, \varphi, radius)$, we use the following equation:

$$radius = \sqrt{(x^2 + y^2 + z^2)}$$
$$\theta = \operatorname{atan2}(y, x) + \pi$$
$$\phi = \cos^{-1}(z/radius)$$

Conversely, to get from spherical $(\theta, \varphi, radius)$ to cartesian (x, y, z) coordinates (see Pharr et al. 2010 [PH10], page 114), we use:

$$x = radius \cdot \sin\theta \cdot \cos\phi$$
$$y = radius \cdot \sin\theta \cdot \sin\phi$$
$$z = radius \cdot \cos\theta$$

3.3.6 Types of projections

To calculate the image pixels (x, y) from spherical coordinates, we have the option to use various projection types, as already briefly introduced in Section 2.2.2. Their respective formulas are as follows (see Houshiar et al. 2015 [Hou+15]):

Equirectangular projection

$$I_x = \theta$$
$$I_y = \varphi$$

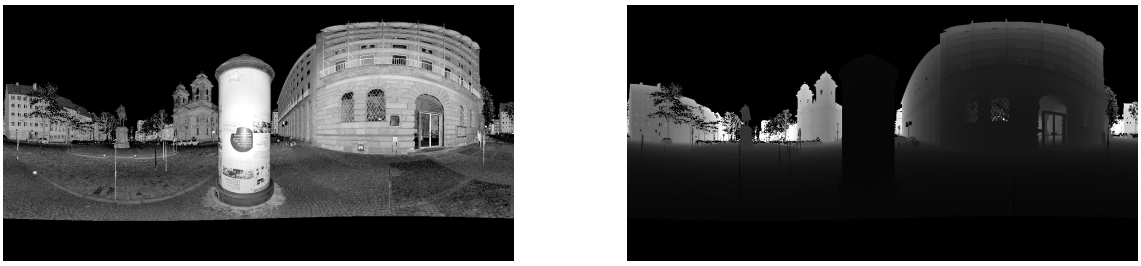
Cylindrical projection

$$I_x = \theta$$
$$I_y = \tan(\varphi) + \pi$$

Mercator projection

$$I_x = \theta$$
$$I_y = \ln\left(\tan(\varphi) + \left(\frac{1}{\cos(\varphi)}\right)\right)$$

3.3.7 Saving textures



(a) Colormap (b) Depthmap

Figure 3.6: Panoramic images generated from the imported point cloud

At this point, each new row in the point cloud file returns the following data: x, y, z, r, g, b, radius, θ, φ, I_x and I_y. This enables us to create the panorama image files. The colormap image is formed from I_x, I_y, r, g and b. The depthmap is created from I_x, I_y and radius. As the names imply, the colormap is necessary to color the 3D panorama (it will be applied as a texture for the 3D panorama) and the depthmap is used for displacing the individual vertices from the center by their depth value. The generated depth and color maps are stored with 8-bit unsigned integer values ranging from 0 to 255. After the point cloud file has been imported they are automatically saved as .jpg image files.

3.3.8 Meshing

After the panorama images have been created, the depthmap image is used for meshing. This is accomplished by looping through the individual pixels of the image and creating four-sided polygons (dashed line in red) clockwise from the current pixel (shaded green) and the neighboring pixels to the right, bottom-right and bottom.

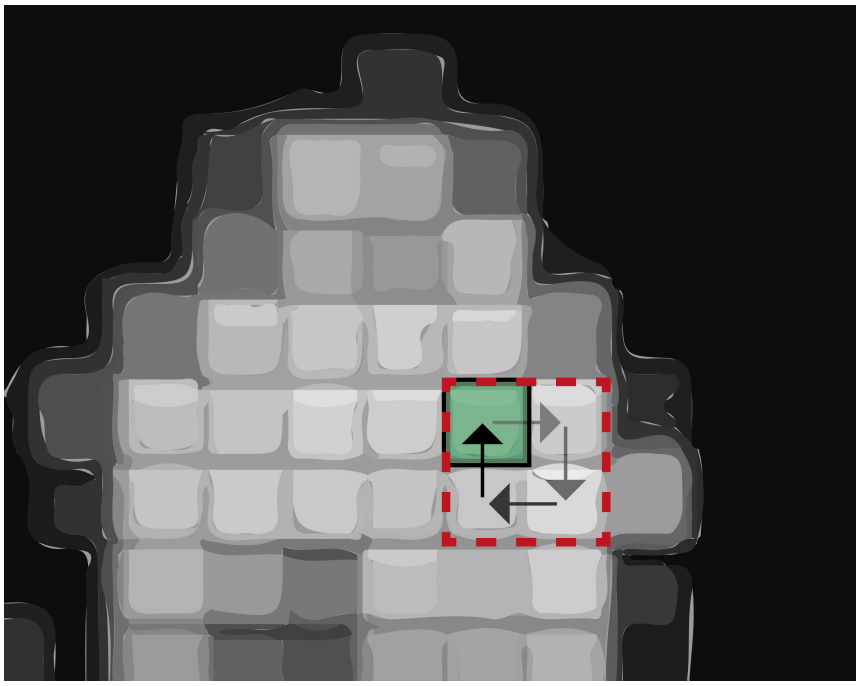


Figure 3.7: PC2B Meshing Algorithm

Each of the four vertices is converted back from the image plane to the cartesian coordinate system and contributes to build up the 3D surface. This process is repeated until every pixel of the image has been processed.

3.3.9 Texture Coordinates and Normals

While the mesh is being generated, texture coordinates and normals are being calculated, too. Texture coordinates vary from 0.0 to 1.0 in the x and y direction, respectively. Real time computer graphics denote the texture coordinate axes as s and t, 3D software applications tend to name them u and v. By dividing the current pixel coordinate on the image plane by the width and height of the image, respectively, the coordinates can be normalized to get the texture coordinate. 3D Applications use the texture coordinate information to properly map an image texture (in our case the colormap) onto the 3D surface.

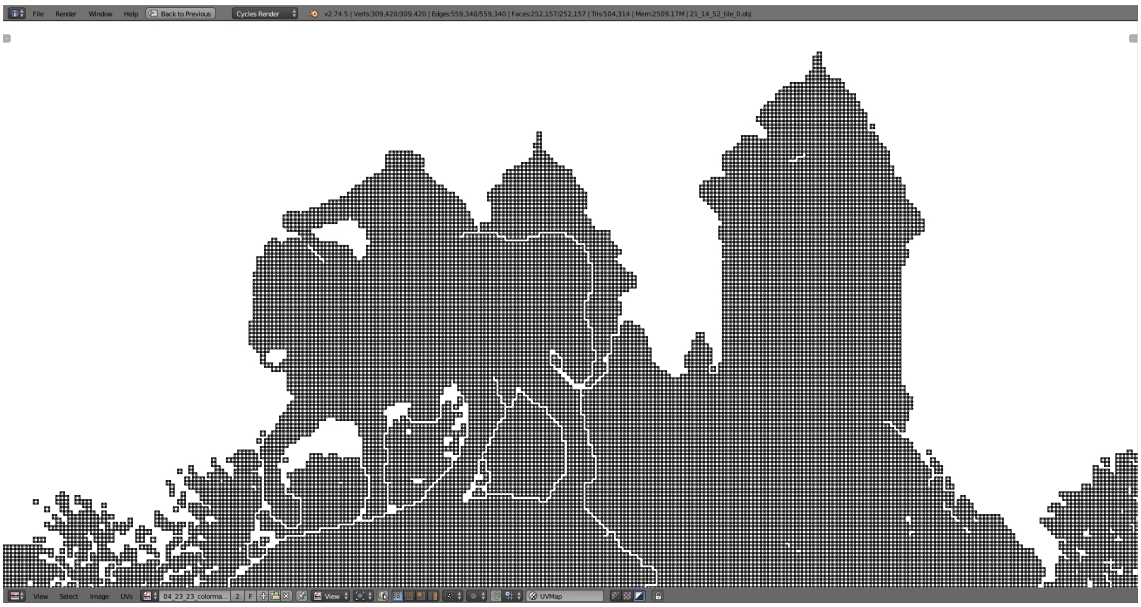


Figure 3.8: PC2B Texture Coordinates viewed in Blender

Calculating normals is accomplished by determining the cross product of the two vectors forming the current quad. It may happen that a polygon will get distorted at locations where the depth image values change quickly. Hence, the four-sided polygon is divided into two triangles and normals are generated for each of them. Normals are very useful to test certain properties of the mesh. This research uses normals to determine if a face is oriented perpendicular to the center of the panorama. Computing the dot product of a normal and a vector from the origin to the face returns the cosine of the angle between those two vectors. This is used in PC2B to discard polygons which would normally distort the mesh, as shown here:

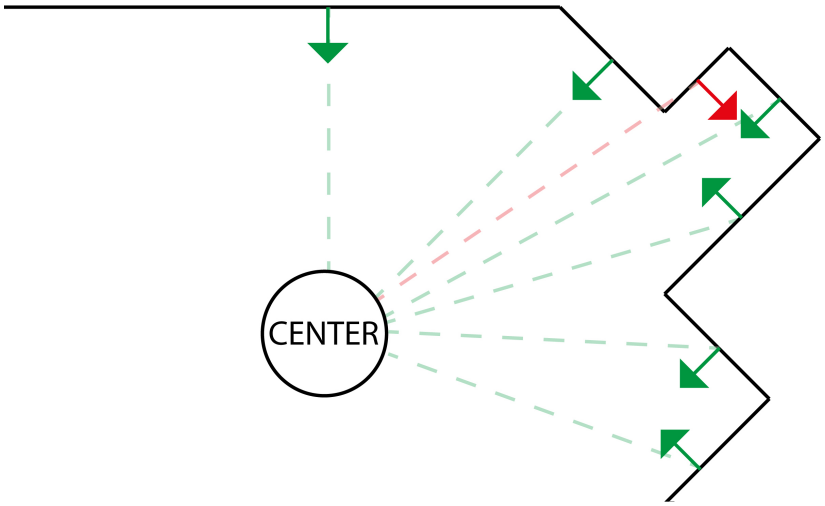
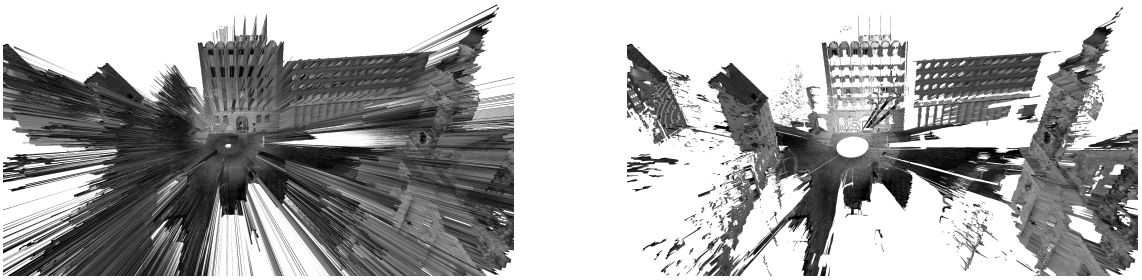


Figure 3.9: PC2B Normal filtering concept

This improvement cleaned up the generated mesh, since a lot of faces have been created in undesirable places.



(a) Without normal filtering (b) With normal filtering

Figure 3.10: The generated mesh before and after normal filtering in PC2B

3.3.10 Mesh Exporter

As a last step the generated mesh needs to be exported. Just like for importing, there is a huge amount of file formats available to export to. One format had to be chosen that supported at least vertices, texture coordinates and faces.

.obj

The .obj format is one of the most popular, and possibly easiest to understand, file formats to save 3D geometry with not only vertices, but normals, texture coordinates, parametric surfaces, splines and much more. With .obj files, usually .mtl files are saved. Those Material Template Library files store additional information about the 3D material, such as ambient, diffuse or specular color, transparency and reflection settings, or filepaths to image textures. It was the first choice when testing the mesh export from the converter software and examining it in Blender. Furthermore this file format can be used with almost any other 3D software application.

.blend

A personal goal for this research was to implement a .blend export feature to allow for a native importing of the panorama mesh into Blender. However, this goal was not reached in this project. As it turned out, exporting the binary Blender file format was quite complicated, due to its versatile structure. An experienced Blender Developer, Jeroen Bakker, stated in 2009: "[...] When implementing loading and saving blend-files in a custom tool the difficulty is the opposite. In a custom tool loading a blend-file is easy, and saving a blend-file is difficult. [...]" (see Bakker 2009 [Bak09]). At least, implementing the feature with the limited time for the thesis was not feasable.

custom format

Even the Blender community suggested to not use the .blend format directly, but rather try a custom binary format (see thread on BlenderArtists [Ble14b] by the author). Implementing such a binary file format would result in a fast and memory efficient way to exchange the mesh between PC2B and Blender as well as a comfortable solution for users, since the file could automatically be openend with Blender after meshing is finished. In this research the custom file format was omitted in

favor of the aforementioned .obj file format. However, it is under consideration to be implemented eventually.

3.4 How the conversion is implemented

Additionally, an OpenGL viewer is implemented to visualize the data while it is being processed. A programming language and framework that provide the necessary performance and graphical user interface were needed. We decided to use C++ with Qt 5 for this task.

3.4.1 UML class diagram

PC2B was developed using object-oriented programming (OOP). To encapsulate certain properties and actions by creating and using objects, those are represented as classes in the source code. PC2B consists of several classes.

Helper classes, such as *Point3D* and *FileType*, provide clear representations of entities while still being flexible for future extensions. Graphical User Interface (GUI) code is contained in *MainWindow*, which inherits from *QMainWindow*. This *MainWindow* is made visible in an instance of a *QApplication* when the program is launched and creates user interface elements as well as instances of PC2B's main components responsible for processing LiDAR data. One of those component parts is the *ImportWorker* which inherits from *QRunnable* and enables it to run on a separate thread in a *QThreadPool*. This is important, because reading and parsing a 6 GB point cloud file can take several minutes which freezes the main GUI loop if on the same thread. The same applies to *MeshWorker* which implements the meshing algorithm and exports the final textured mesh for use in 3D applications. The *Panorama3D* class encapsulates methods to convert a 3D point to a 2D pixel and vice versa. Additionally, it stores the panorama images. Finally the OpenGL viewer is implemented in two classes, *GLWidget* which inherits from *QGLWidget*, and *GLMesh*. While *GLWidget* provides a canvas to draw 3D graphics onto it via OpenGL, *GLMesh* represents the 3D point cloud mesh and groups every property such as vertices, colors and shaders.

This relationship between individual objects is usually depicted in the Unified Modeling Language (UML) as a class diagram. Below is an example of what a simplified UML class diagram for PC2B could look like:

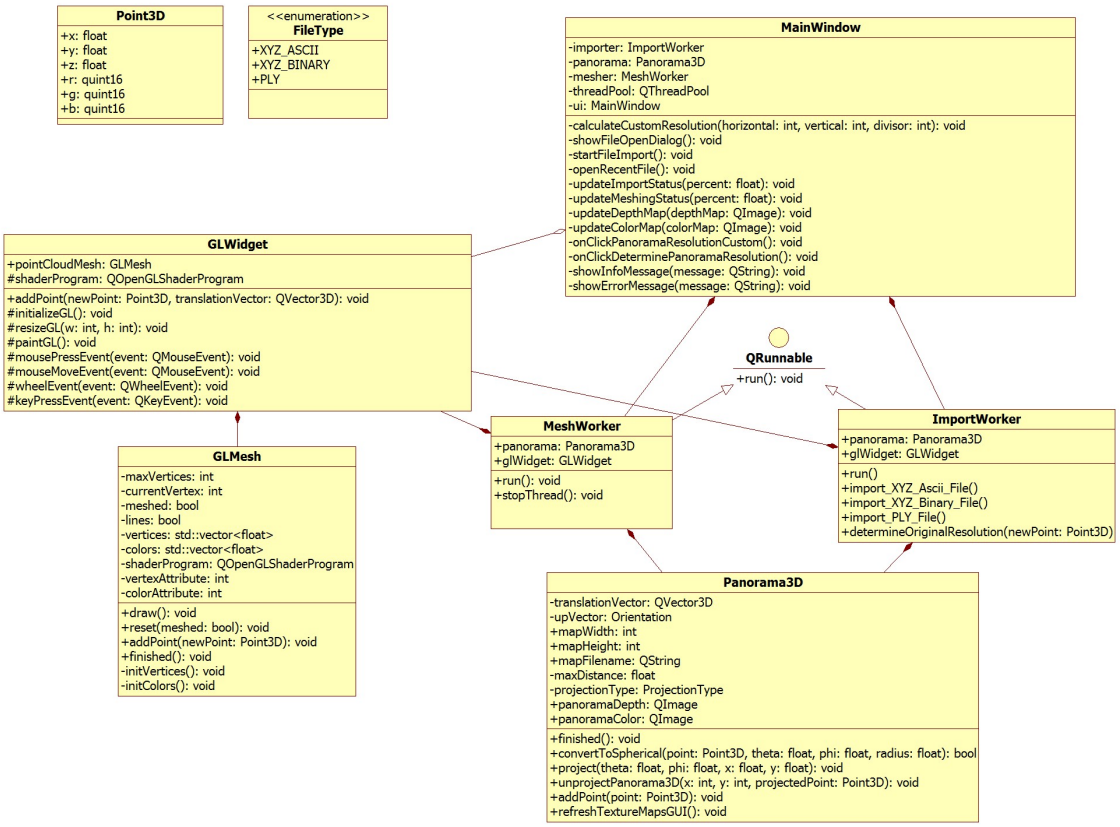


Figure 3.11: PC2B UML class diagram (simplified)

3.4.2 Working with the PC2B User Interface

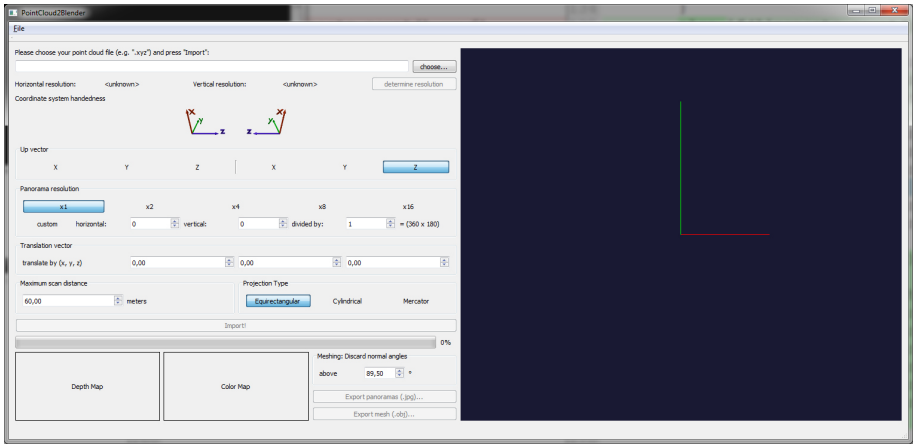


Figure 3.12: PC2B Graphical User Interface

The Graphical User Interface (GUI) of PC2B is divided into two parts. On the left hand side it allows to setup and load point cloud files that PC2B will process. On the right hand side it visualizes every step of the conversion procedure in a 3D viewer, which is described in more detail in the following chapter.

To convert a point cloud file the user can either browse to a file on the local machine or open the most recent file from the file menu. This is the only prerequisite to start the conversion in PC2B. However, the file may need additional tweaking which a user can perform by easily adjusting the respective setting in the user interface.

Users are able to define the coordinate system handedness and its corresponding Up-Vector. This provides a way to handle point clouds with varying orientations. Furthermore the point cloud can be translated by a user defined 3D vector and it is possible to set the maximum scan distance to work with files from diverse point cloud generators.

Lastly, panorama settings are provided. On the one hand the projection type can be set to "Equirectangular", "Cylindrical", or "Mercator". On the other hand the final panorama resolution can be defined by using presets or a custom resolution. The final mesh can be cleaned up by adjusting the threshold at which a normal angle will discard a polygon.

3.4.3 OpenGL Point Cloud Viewer

During the development of PC2B, it turned out that debugging would be easier if data is visualized simultaneously while importing and meshing. To that end we implemented a point cloud viewer. Initially it displays the coordinate axes. While importing it displays, at maximum, three million points from the point cloud file and during meshing it displays the final 3D panorama mesh.

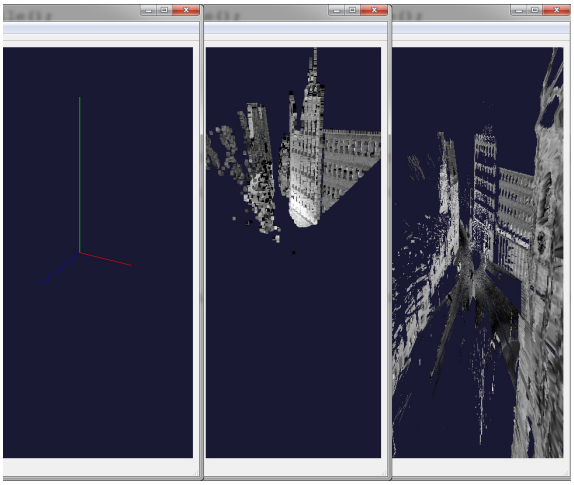


Figure 3.13: PC2B OpenGL Viewer

The Qt framework offers its own implementations to utilize OpenGL for drawing. Setting up the basic functionality within Qt was done with the help of a Russian video tutorial (see Enzhaev 2014 [Enz14]). Using modern OpenGL drawing code was important to be able to handle millions of points and render high-resolution 3D meshes in real time.

3.4.4 Optimizations

The initial algorithms and approaches had some flaws, which needed to be eliminated to achieve a clean mesh out of the converter. Further, there are still some issues with the converter software which we would like to address briefly:

Various import settings

The point cloud file which is going to be meshed could need some minor adjustments, such as a predefined translation along a 3D vector, or it might have a different orientation. In addition, the maximum scan distance or the normal angle threshold for discarding normals (see Section 3.3.9) could need tweaking. Those settings are exposed to the user in PC2B.

Panorama pixel depth testing

It might happen that two or more points from the point cloud fall into the same pixel in the 2D panorama. On the one hand this might result in a noisy image. On the other hand, this affects the generated mesh, if not handled with care. To avoid any errors, it is important to take only the closest point to the camera, instead of

just letting every 3D point override the corresponding pixel in the image. This is implemented in the converter software.

Panorama flipped horizontally

The panorama images are currently flipped horizontally. This is not a serious issue, since the texture coordinates are created based on these images and therefore the texture is eventually applied correctly. However, this might be something to consider changing.

Panorama noise reduction

Since there is only a limited number of points, the panorama texture can get quite noisy or gaps can occur, especially with a higher resolution option set in the converter. A distinct change from light to dark gray values in the depth map will result in a noisy 3D surface as well. To solve this issue, the image pixels could either be formed by averaging the values in a certain region or blurred by a user-defined amount.

Remove doubles

The meshing algorithm currently produces a very high point count in the .obj file. For Example: A quadruple resolution panorama with 2,198,528 vertices will be automatically decimated by 2,100,716 vertices using the "remove doubles" option in Blender 3D. This could be solved by buffering the vertex data and using several passes for writing vertices, texture coordinates, and normals in the mesher algorithm. Every vertex would only be saved once and get referenced by its index when defining the individual faces.

Tiling

Due to the higher-resolution meshes being several megabytes in size and taking some time to import in Blender, this could also be optimized. Depending on the resolution set by the user, the converter could create tiles. If, for example, a quadruple resolution is set, four tiles get created (that is four separate .obj, .mtl and image files). Another idea might be to provide users a tool to define custom borders in the panorama to only export specific parts of the scan.

Quantized 3D surface

The panoramic image pixels are stored as 8-bit unsigned integer values at the moment. This does not allow depths in very fine steps, as 8 bit can only store $2^8 = 256$ values. A good solution would be to store the values with at least floating point precision. This would also eliminate the need for the user to specify a maximum scan resolution. Most importantly the mesh quality would be greatly improved, which we believe would be highly appreciated by artists.

Better memory management

Almost all of the aforementioned issues have one reason in common: they require quite a lot of memory, and thus better memory management in order to be resolved. PC2B cannot be run on older systems with 1 GB RAM at the moment, because of the allocation of a fixed memory block for the OpenGL point cloud viewer. A better approach would be a dynamic memory allocation. In this regard the use of the C++11 standard with smart pointers might lead to significant improvements.

Although PC2B works well without those optimizations, they would greatly benefit the output of the converter software, in our view.

Chapter 4

Production: Recreating the Pellerhaus from 1605

The generated mesh from PC2B may still need additional cleanup and detail. The mesh topology is dependant on the meshing algorithm used, naturally. It might not be ideal for an artist to have thousands of polygons build up a surface that could be represented by only one polygon. Furthermore, the laser scanner beam cannot sample reflective or transparent structures, such as windows. Therefore it can only provide artists a reference for manual tracing.

4.1 Modeling the current Pellerhaus facade

The lower part of the current Pellerhaus facade is very similar, almost identical, to the historic one. Thus modeling the modern facade based on LiDAR, photogrammetry, and photographic reference appeared to make sense.

4.1.1 Using the PC2B converter software

First, the LiDAR data was preprocesed in PC2B. We created five scans on location, so all the scans have been meshed with PC2B and imported into Blender. A manual registration of the scans was necessary in Blender, because the scans have been aquired at different locations. Fortunately, all the scans were oriented similarly due to the compass sensor in the laser scanner, and the Pellerhaus facade was a good reference point to align them with each other.

It turned out that the workflow for artists is relatively straightforward. Each scan took approximately 5 minutes to be processed by PC2B. Importing the results in Blender worked flawlessly and created all the necessary data blocks, like the named object and texture. We noticed that distinct names for the panorama textures need to be used to avoid multiple uses of one texture in different scans. The mesh is very helpful to get a sense of the scale of objects and, where the detail is not sufficient, the texture information can help in finding details. Moreover, due to the clear texture, the artist can select unnecessary polygons by simply choosing them from the image texture coordinates (which are usually called UV's in 3D applications) and delete them (see Section 3.3.9). Other meshing algorithms, such as the Delaunay Tetrahedralization, create textures with very small UV islands and store them efficiently (see Appendix A.4).

All in all, PC2B generated a mesh that was helpful for tracing. This is how the imported scans looked in Blender:

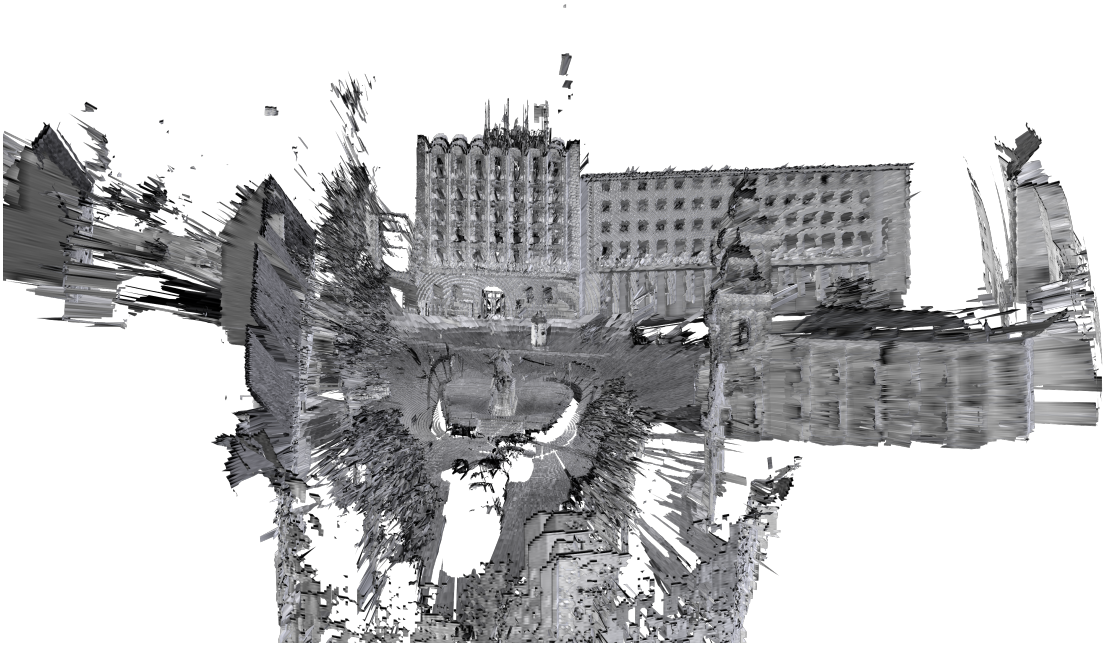


Figure 4.1: The manually registered LiDAR scans of the Pellerhaus

4.1.2 Using UAV references with photogrammetry

Unmanned Aerial Vehicles (UAV's, or simply drones) are becoming affordable, and this is true even for very good quality models. In our research we use the DJI Phantom 2 with a GoPro Hero 4 Black mounted on a 3-axis DJI Zenmuse H3-3D Gimbal to create photographic aerial references of the Pellerhaus Nürnberg.

Flying a drone legally is not as easy, as one might initially assume. Before even being able to take off with a UAV, German law requires general permission just to enter the air space. In addition, every UAV pilot needs UAV insurance.

Regarding the usage of drones inside the city center of Nuremberg, there are more restrictions. The city center of Nuremberg is covered by a controlled air space. Flying in that air space is not permitted until the initial permission from the bavarian aviation authority and UAV insurance are upgraded to commercial versions. Furthermore, pilots need a clearance from Air Traffic Control (ATC). Additionally, the starting and landing procedure requires cordoning off of pedestrians and a special license from the traffic authority. Lastly, the owner of the property needs to be asked for permission to allow the takeoff and landing of the aircraft.

Luckily there are some laws that permit video shoots and photography. For example, the Freedom of Panorama Law (§ 59, German Urheberrechtsgesetz) allows taking photos from pavements and roads permanently located in a public place. This right to freely take and share photographs of buildings and works of public art was almost abolished on July 9th, 2015 by the European Parliament (see Blacker 2015 [Owe15]). Fortunately, members of the European Parliament have voted against this controversial EU plan (see Cheesman 2015 [Chr15]) with an overwhelming majority. We would expect immense consequences for historic documentation of buildings if this would have happened.

In total we made three flights on location. We planned to use the second flight for single photos shot in an interval of 1 second whereas the other two flights were videos in 4K. Unfortunately, it turned out that the third flight wasn't recorded at all and the second flight was captured as a video file. Luckily we noticed the third video missing while still on location, with a bit of battery life left and about one hour left to use the air space, so we made at least two impressive aerial shots before ending the mission.

Having video files instead of still images, the files needed to be exported as still frames to be able to be processed by the software. We tried both the free "Visual SfM" and commercial "Agisoft Photoscan Pro" software solutions to generate additional colored meshes of the Pellerhaus. The total processing time was about 20 hours for Visual SfM and 40 hours for Photoscan to get a 3D point cloud from the images. Comparing the results we noticed that Visual SfM generated a bend facade while Photoscan Pro kept it very straight. Although not recommended (see Balletti et al. 2014 [Bal+14]), we used a GoPro camera with a short focal length and thus a strong lens distortion for 3D reconstruction. We can confirm that the use of this camera type is not a good choice, since the radial distortion can produce errors in the feature matching phase of photogrammetry. Still, the output should be a good reference for the rough shape of the reconstructed object. If available, LiDAR data should be preferred over Photogrammetry for accurate building reconstruction.

4.1.3 Using reference images

In addition to the LiDAR data and Photogrammetry models, photos were taken on location for additional reference. Very small details can be inspected in photographs while creating an accurate surface. Based on the versatile data available, the model was built up slowly, first by creating the rough shapes for the building and then constantly refining them to get the final model.

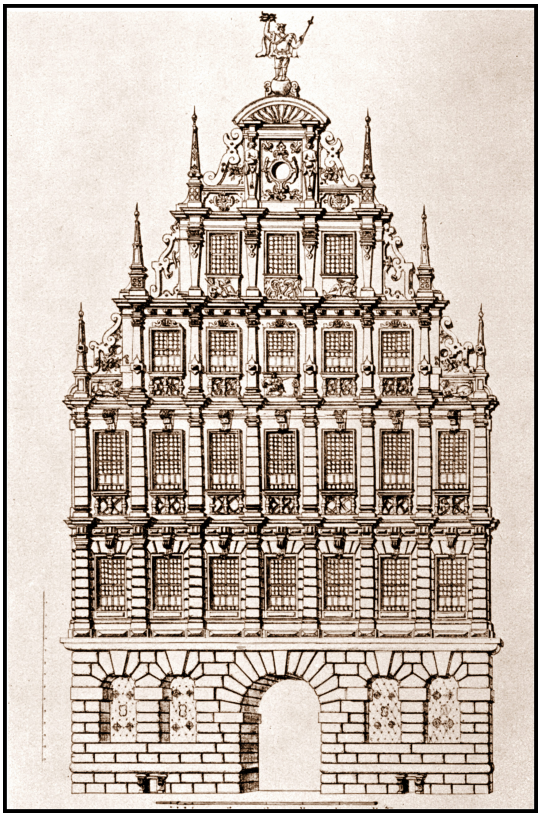
4.2 Modeling the historic Pellerhaus facade

Based on the modern facade it was possible to get a good feel for the size of the Pellerhaus in the Renaissance. Still the main parts above the ground floor of the facade differ dramatically from the modern one. They can only be extracted from photographs.

4.2.1 Using historic images as guide

It is quite lucky that the Pellerhaus history has been documented by an enormous amount of pictures. A fascinating number of about 190 pictures has been provided for free through the association Altstadtfreunde Nürnberg e.V., not including about 100 additional pictures depicting the space in front of the building, to be used in this research.

This collection contains a few early drafts of the facade, likely created in the year 1601. Frontal, side or top views are best suited for modeling, especially when they are orthogonal (not distorted by perspective).



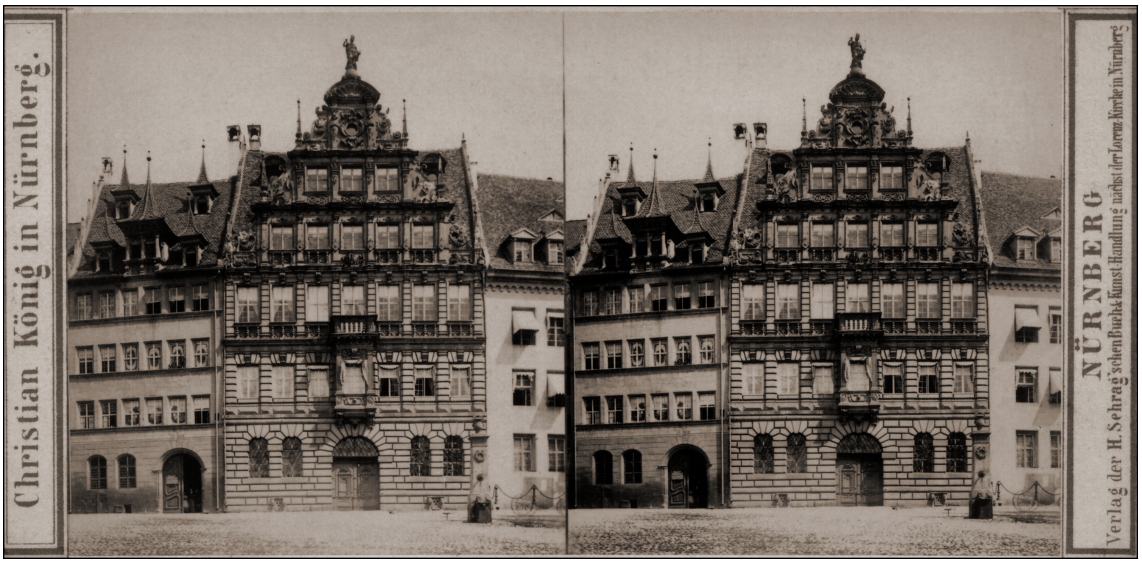
Credit: Altstadtfreunde Nürnberg e.V.

Figure 4.2: Early draft of the Pellerhaus facade of around 1601

The modeling process began with adjusting the size of the image to the current Pellerhaus provided by LiDAR. The ground floor of the modern Pellerhaus was duplicated and served as the first guide for the creation of the upper floors. Floor by floor, bottom to top, was modelled by tracing the reference image (Figure 4.2). This image was the clearest and best to work with to block out the historic facade shape. Unfortunately this was only possible to a certain degree. The Pellerhaus has a lot of tiny details, artistic ornaments and even sculptures at the top. These are so complex that the overall three-dimensional shape can only be estimated. Without historic experts it is very time-consuming to model the forms correctly. For this reason the historic model in this research only contains the obvious, rough forms of the facade. We hope that this model can be refined by working closely under the supervision of the association Altstadtfreunde Nürnberg e.V. in the future.

4.2.2 Using historic stereoscopic images with photogrammetry

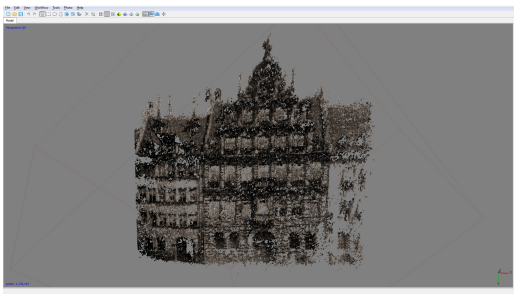
Although the historic pictures were already a blessing, there are two additional images aquired by stereophotography.



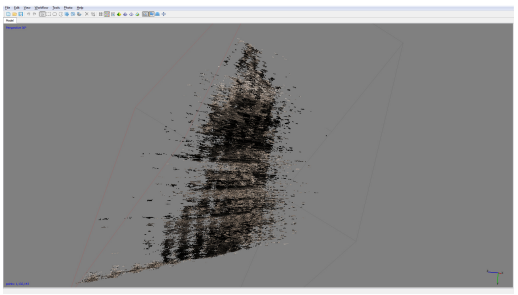
Credit: Christian König

Figure 4.3: Historic stereo pair from around 1876

This project tried to extract the 3D information from those stereo pairs via photogrammetry. Feeding the left and right image into Agisoft Photoscan Pro, respectively, was expected to give a nice 3D representation of the historic facade. Unfortunately this did not work well:



(a) Historic point cloud front view



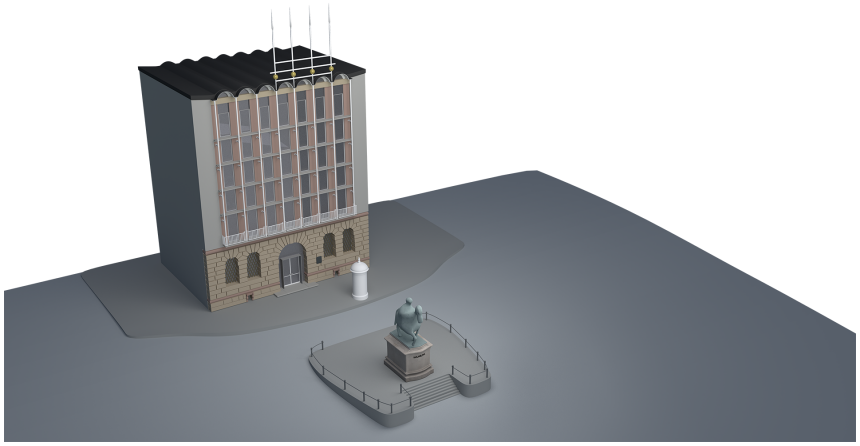
(b) Historic point cloud right side view

Figure 4.4: Automatic reconstruction results from historic stereo pairs

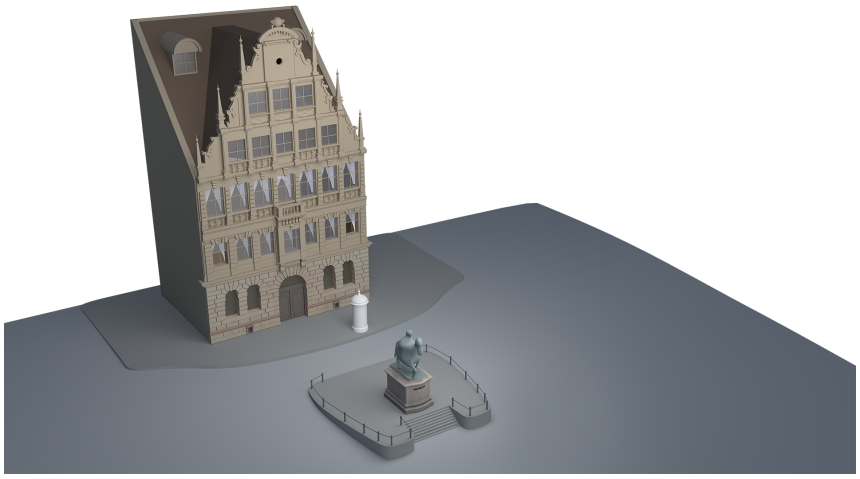
The result is quite noisy, but might be suitable for a presentation if viewed carefully with only slight camera movements. However, the results are not satisfying in our view. Hence, they are not suited for the historic 3D reconstruction in this research.

4.3 The finished models

The manual modeling process took 30 hours and 20 minutes to create the following results:



(a) Modern Pellerhaus



(b) Historic Pellerhaus

Figure 4.5: The finished 3D models of the Pellerhaus

Rendering took 52 minutes for the modern building model and 54 minutes for the historic model. The render engine Cycles was used at a resolution of 7680 by 4320 pixels and 200 samples.

After refining the models, they can be animated to create educational videos about the Pellerhaus history. The historic Pellerhaus can be simulated to fracture and collapse during World War II leaving only debris and major parts of the ground floor. The debris can disappear slowly and the new facade will be created gradually. This may be supported visually by a timeline in the video. Finally, the finished video can be played out in stereoscopic 3D or the models can be 3D printed.

Timelapse videos have been created while modeling both buildings and will be published shortly after the research on the project website.

Chapter 5

Conclusion and Future Work

This research presented a method to process point cloud files, mesh them via panoramic projection, and convert them to textured 3D panorama meshes. Furthermore, it examined whether this method creates useful meshes for artists, by testing it in a real world use case with the goal to reconstruct the historic Pellerhaus Nürnberg. This chapter discusses results and gives an outlook of future work.

5.1 Conclusion

To conclude, we would like to summarize the work, mention its limitations, and present the contributions made by this research.

This project was begun with the goal of finding a way for Blender artists to work with laser scanner point clouds in Blender. Blender cannot display colored 3D point clouds in its viewport. Therefore, this required finding a novel method to make point clouds available to artists who wish to work with such data in Blender. We propose a conversion from laser scanner point clouds to textured polygon meshes, because those geometric shapes can be viewed and rendered in Blender. Due to the characteristics of laser scanner devices, environments are sampled in a sphere. Projecting the point samples onto a two-dimensional grid simplifies the process to connect those points and determine texture coordinates. The 3D information can be retrieved by applying the inverse transform to this grid. The 3D mesh is then exported to a commonly used file format to exchange geometric information between applications.

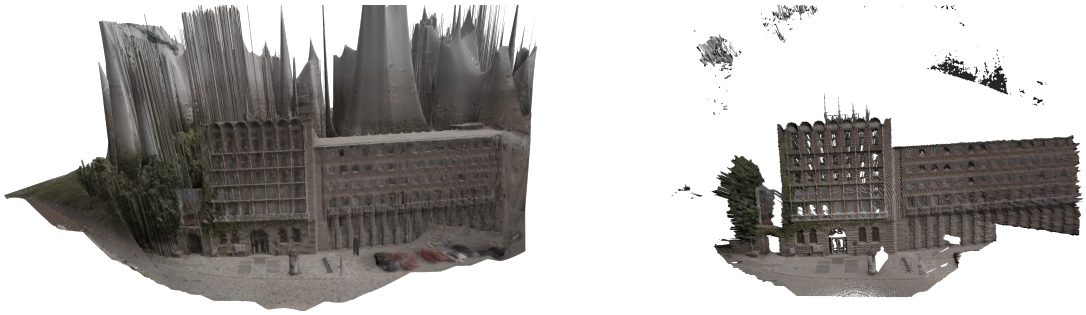
5.1.1 Mesh generation

The generated mesh was evaluated by using it in an artistic project. The goal of this project was to create the modern and historic facades of the Pellerhaus Nürnberg based on the generated 3D panoramas. The goals of this project have been met. First, creating 3D panoramas from point clouds solves the issue that Blender cannot display point clouds and therefore enables artists to work with LiDAR data. Second, the generated meshes have been helpful in producing 3D models of the Pellerhaus Nürnberg.

The limitations of this project are mostly related to the quality of the generated mesh. The generated mesh is good enough to provide artists a three-dimensional reference of a particular environment, but it is not perfect. A perfect mesh would have perfectly flat surfaces, for instance. This can only be achieved with almost no quantization of the image files, e.g. by using floating-point values for the pixel data or by using the original point data instead of reprojecting the panorama images. This is currently not implemented in the PC2B prototype due to its memory management, though PC2B can be optimized to improve the generated meshes. For the purpose of this research, the results are satisfying.

5.1.2 Handling non-LiDAR data

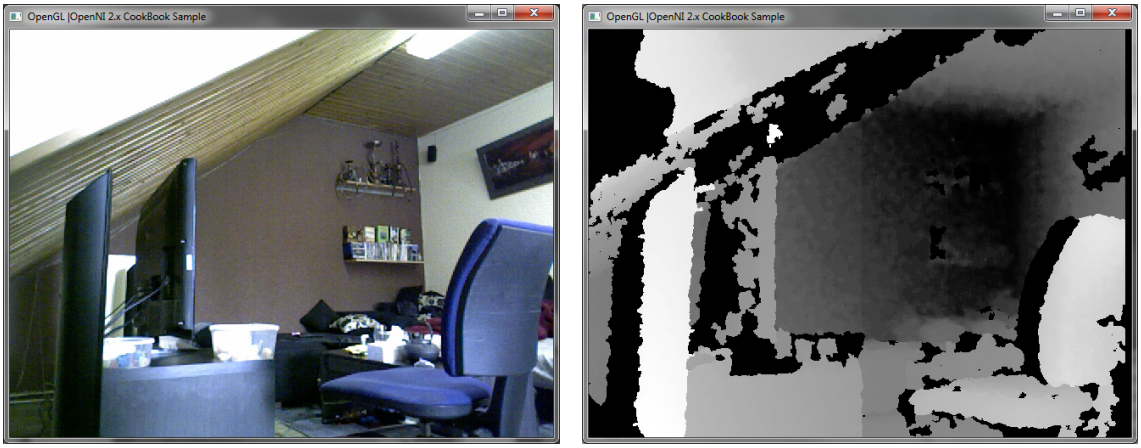
Although this research was aimed at investigating point clouds generated by laser scanners, we did experiment with other point cloud generators as well. There are additional ways to produce point clouds, such as Photogrammetry or Depth Cameras. The more points they generate, the better. We have tested our algorithm on both of those point cloud types. We found that our algorithm works well with non-LiDAR data. Photogrammetry point clouds are meshed as follows:



(a) Agisoft Photoscan Pro meshing (b) PC2B meshing

Figure 5.1: Comparison of meshing in Photoscan vs. PC2B

Additionally, we examined the output of a Depth Camera, namely the Asus Xtion Pro Live. Mrs. Eichhorn, a fellow student finishing her bachelor's degree in October 2015, teamed up with us to study ways that our present research can be used in combination with her's. The title of her research paper reads "Selbstständige Navigation eines autonomen Roboters mittels 3D-Tiefensensor" (which translates into English as "Independently navigating an autonomous robot employing a 3D depth sensor", see Eichhorn 2015 [Eic15]).



(a) Color image (b) Depth image

Figure 5.2: Depth Camera images

While we thought about ways to make use of 3D panoramas to help the robot understand its environment, she was asked to provide a point cloud file to see how PC2B would process it. She wrote a custom tool to save the output of a depth sensor to an .xyz file which is compatible with the PC2B converter software. This is the result of this data:

(a) PC2B depth map panorama (Note: flipped) (b) PC2B meshing

Figure 5.3: Meshing of Depth Cameras with PC2B

The results demonstrate that it is possible to generate 3D panoramas from depth sensors as well. The test data has 77,361 points and is processed in 3297 milliseconds for a quadruple resolution set in the PC2B converter. We plan to continue exploring the use of PC2B for her purpose.

5.2 Future Work

In this section we would like to present further ideas of how PC2B and the 3D models can be used in other fields of application.

5.2.1 Improving memory management and performance

Naturally, bug fixes, optimizations, and a more sophisticated software architecture are important to move from a prototype status to a mature product. We learned a great deal about the weaknesses and strengths of the underlying algorithms and gained the knowledge to decide what parts need to be optimized. The source code of this software tool will remain open, so anyone who would like to learn from it or contribute improvements will be welcome to do so.

5.2.2 Multi-threaded PC2B Server

Currently PC2B can only process one point cloud file at a time. The software could be extended to process multiple files at once or delegate them to other computers in a network, just like a renderfarm distributes 3D scenes to available clients for rendering.

5.2.3 Integration in the Blender core

As this research was executed to find ways to improve Blender, the integration of PC2B algorithms in the Blender core is desirable. Before starting the development of new source code related to this project, more research is needed. First, PC2B needs constant testing and tweaking to avoid adding any bugs or showstoppers into the Blender core. Next, the Blender community needs to be surveyed as to whether this feature would be appreciated or not. Only then is any effort reasonable to try integrating PC2B into Blender. The findings of this research will be shared with the Blender community shortly after publication.

5.2.4 3D printing, lenticulars and stereoscopic movies

The 3D models of the Pellerhaus can be used in various ways. Creating a 3D print from the historic model was already planned from the very beginning of this research. The historic model consists of many small parts, so a suitable material and size needs to be determined first. Different materials have different wall-thickness requirements. While plastic might have a minimum wall-thickness of 0.7 mm, steel could require 3 mm (see Shapeways Inc. 2015 [Sha15]). In addition, the models could be used to create 3D postcards by using lenticular printing (see Wikipedia 2015 [Wik15j]). This technique works by overlaying an interlaced image with a plastic sheet made out of many tiny lenses. Changing the view position results in a change of the image. With the right tools it is possible to create various effects, such as the illusion of depth. A similar principle is used for 3D television that does not require 3D glasses. Creating stereoscopic movies might be another use for the Pellerhaus 3D models. Watching a demonstration of the Pellerhaus history with the perception of depth could result in a more immersive experience for the viewer.

5.2.5 Augmented and Virtual Reality

Audiences could experience the Pellerhaus even better with interactive applications, such as games, virtual walkthroughs with virtual reality headsets, or the combination of real environments with digital objects. The latter could be realized by creating a smartphone application where users can explore the old Pellerhaus facade by pointing the phone's camera at the modern facade. The historic facade would then be overlayed and the user could view it from different angles (see Timetraveler augmented Ltd. 2014 [Tim14]).

Appendix A

Appendix

A.1 Pellerhaus Architect Biography

A.1.1 German

Körner, Hans-Michael and Jahn, Bruno (2012, p.2128) [KJ12] wrote a detailed biography about the architects of the Pellerhaus Nürnberg:

> **Wolff, Jakob d.Ä., Baumeister, Bildhauer, * um 1546 Bamberg, † 4.4.1612 Nürnberg**
> W. wurde 1596 Stadtbaumeister in Nürnberg, wo er mit seinem Sohn Jakob -> W. d.J. die Fleischbrücke errichtete. 1601-05 beteiligte er sich am Neubau der Feste Marienberg in Würzburg und am Umbau des Echtertors. Sein Hauptwerk ist das Pellerhaus in Nürnberg (1602-07), einer der vornehmsten Privatbauten der deutschen Renaissance (im Zweiten Weltkrieg zerstört; die Reste des Arkadenhofs wurden in den modernen Bau einbezogen).
> LITERATUR: Wilhelm Schwemmer: J.W. der Ältere und der Jüngere. In: Fränkische Lebensbilder. Bd. 3. Hrsg. v. Gerhard Pfeiffer. Würzburg 1969, S. 194-213.
>
> **Wolff, Jakob d.J., Baumeister, *1572 Bamberg, † 24.2.1620 Nürnberg**
> W. war Schüler seines Vaters Jakob -> W. d.Ä., erhielt 1605 in Nürnberg die Stelle eines Stadtwerkmeisters, hielt sich mit Erlaubnis des Rats u.a. in Bayreuth, Frauenaurach und Schwabach auf und begann, beeinflußt von der niederländischen und italienischen Renaissance, 1616 mit dem Neubau des Rathauses in Nürnberg, der 1622 von seinem Bruder Hans vollendet wurde.

A.1.2 English

Körner, Hans-Michael and Jahn, Bruno (2012, p.2128) [KJ12] translated from German by the author:

Wolff, Jakob d.Ä., master builder, sculptor, *1546 Bamberg, † 4.4.1612 Nuremberg
Wolff became the city architect of Nuremberg in 1596, where he and his son W. d.J. built the Fleischbrücke. During 1601-05 he took part in the new build of the stronghold Marienberg in Würzburg and in the reconstruction of the Echtertor. His principal work is the Pellerhaus in Nuremberg (1602-07), one of the most noble private properties during the German Renaissance (destroyed in the Second World War; the remaining parts of the arcade court have been included in the modern building) [...]

Wolff, Jakob d.J., master builder, *1572 Bamberg, † 24.2.1620 Nuremberg
Wolff was the student of his father W. d.Ä., was given the job of a Stadtwerkmeister (Municipal Master of the Works) in 1605, had the permission from the council to stay in Bayreuth, Frauenaurach and Schwabach and started, influenced by the Dutch and Italian Renaissance, with the new build of the city hall in Nuremberg in 1616, which was finished by his brother Hans in 1622.

A.2 Software used

A.2.1 LaTeX

This paper was written in LaTeX. On Windows, TeXstudio in conjunction with MikTeX (both portable versions) have been used for visual creation of the document. The author decided to switch from the free Adobe InDesign CS 2.0 version to LaTeXin favor of it being cross-platform and the hope of making it easier to publish the thesis online in the future. Since the author has never worked with LaTeXbefore, various tutorials [Sha13; Vel15] on the internet have been a great help to get started with the writing process.

A.2.2 FARO SCENE LT

The FARO Focus^{3D} laser scanner creates proprietary files of every scan. Those raw laser scanner point cloud files need to be preprocessed in the FARO SCENE software tool. FARO offers two versions of their software. The usual FARO SCENE software needs to be purchased. We used the free version which is called FARO SCENE LT, which only allows exporting the point clouds in grayscale. More information about FARO SCENE LT can be found in [Far15].

A.2.3 Blender 3D

Blender is a free and open source 3D creation suite. It is used as the primary 3D application in this research, mainly to inspect the generated meshes from PC2B and to create the 3D models of the Pellerhaus. More information about Blender can be found in [Ble15b].

A.2.4 Meshlab

Meshlab is an open source tool which can import and export various file formats. We used it to convert binary test files to ASCII files if they need to get opened in our custom software. More information about Meshlab can be found in [Pao14].

A.2.5 Visual SfM

Visual SfM is a free photogrammetry tool. We used it to generate 3D models from multiple images for free. More information about Visual SfM can be found in [Cha15].

A.2.6 CMP-MVS

CMP-MVS can create dense point clouds with photogrammetry algorithms based on camera parameters and scene descriptions. Those files can be generated in Visual SfM and passed on to CMP-MVS to create higher quality models. More information about CMP-MVS can be found in [Jan12].

A.2.7 AgiSoft PhotoScan Professional

A commercial alternative to Visual SfM to create 3D models from images via photogrammetry. The AgiSoft PhotoScan Professional Edition can be purchased for 3,264.25 Euro. We have used a 30-day test period in this research. More information about PhotoScan can be found in [Agi14].

A.3 Programming frameworks and libraries

A.3.1 Qt 5.4

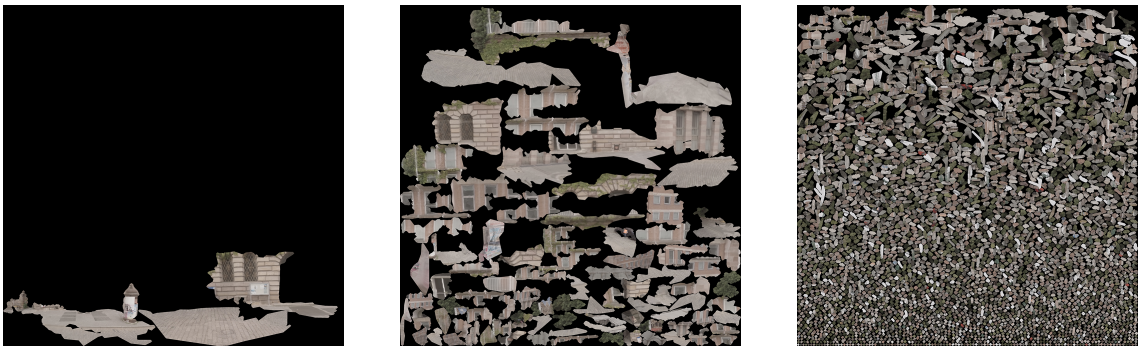
Qt (pronounced "cute") is an open source C++ framework that offers a wide range of features. It provides cross-platform support for user interfaces, image manipulation, graphics drawing and much more. It was the perfect framework for this research due to its mature feature set. More information about Qt can be found in [The15].

A.3.2 OpenGL

The Open Graphics Library (OpenGL) is an abstraction layer for accessing graphics hardware on a high level on almost any operating system and programming language. The current version supports the programmable function pipeline, where vertex and fragment shaders provide a rich set of ways to manipulate the final pixel on the output device, e.g. computer screens. We used OpenGL in this project to draw millions of points and the final mesh generated from them. More information about OpenGL can be found in [Khr15].

A.4 Delaunay Tetrahedralization Texture Maps

These texture maps are generated in Visual SfM with the Delaunay Tetrahedralization algorithm.



(a) Big UV islands (b) Medium UV islands (c) Small UV islands

Figure A.1: Texture maps generated with Delaunay Tetrahedralization

Bibliography

[Wik14] Wikipedia. *Parallax barrier — Wikipedia, The Free Encyclopedia*. 2014. URL: https://en.wikipedia.org/w/index.php?title=Parallax_barrier&oldid=612869363 (Retrieved 06/19/2015).

[Ocu15] Oculus VR, LLC. *Oculus - Oculus VR*. 2015. URL: https://www.oculus.com/en-us/ (Retrieved 06/19/2015).

[Wik15a] Wikipedia. *Nazi party rally grounds — Wikipedia, The Free Encyclopedia*. 2015. URL: https://en.wikipedia.org/w/index.php?title=Nazi_party_rally_grounds&oldid=664601942 (Retrieved 06/19/2015).

[Sta15] Stadtteilforum Langwasser. *Langwasser! Geschichte eines neuen Stadtteils*. 2015. URL: http://stadtteilforum.org/langwasser/langwasser-geschichte/tafel-6.html.

[Goo15] Google Inc. *Google Search Trends: Blender, 3ds Max and Maya*. 2015. URL: https://www.google.com/trends/explore#q=/m/0l1qq,%20/m/01tgyn,%20/m/0svhm&cmpt=q&tz=Etc/GMT-2 (Retrieved 06/19/2015).

[Ble14a] Adam Kalisz (BlenderEi). *Point Cloud support in Blender: What's already there and what does the community need*. 2014. URL: http://blenderartists.org/forum/showthread.php?349851-Point-Cloud-support-in-Blender-What-s-already-there-and-what-does-the-community-need (Retrieved 01/24/2015).

[Wik15b] Wikipedia. *Speed of light — Wikipedia, The Free Encyclopedia*. 2015. URL: https://en.wikipedia.org/w/index.php?title=Speed_of_light&oldid=667601477 (Retrieved 06/20/2015).

[Sch14] Rayk Schroeder. "Was kann eigentlich dieser LIDAR-Scanner? (German) [What is this LIDAR-Scanner capable of?]" In: *Digital Production* 07 / 14 (2014), pp. 8–9.

[FAR13] FARO Technologies Inc. *FARO Focus^{3D} Features, Benefits and Technical Specifications*. 2013. URL: http://www2.faro.com/site/resources/share/944 (Retrieved 06/20/2015).

[Opt15] Opti-cal Survey Equipment Ltd. *Opti-cal Survey Equipment - FARO Focus 3D X 330 Laser Scanner*. 2015. URL: http://surveyequipment.com/faro-focus-3d-x-330-laser-scanner/ (Retrieved 06/22/2015).

[Vic14] Victoria Woollaston. *The 'Prometheus' scanner that can 3D can capture a crime scene in minutes*. 2014. URL: http://www.dailymail.co.uk/sciencetech/article-2562329/Prometheus-style-scanner-maps-crime-scenes-3D-rotated-tagged-without-damaging-evidence.html (Retrieved 06/20/2015).

[Pul15] PulsedLight. *PulsedLight | PulsedLight*. 2015. URL: http://pulsedlight3d.com/ (Retrieved 06/22/2015).

[VEG15] VEGA Grieshaber KG. *Radarsensor zur kontinuierlichen Füllstandmessung von Schüttgut – VEGAPULS 69*. 2015. URL: https://www.vega.com/en/home_ch/Home_International/Products/Product%20cataloque/Level/Radar/102559 (Retrieved 06/22/2015).

[Sen15] Sensors Magazine. *Choosing an Ultrasonic Sensor for Proximity or Distance Measurement Part 1: Acoustic Considerations | Sensors*. 2015. URL: http://www.sensorsmag.com/sensors/acoustic-ultrasound/choosing-ultrasonic-sensor-proximity-or-distance-measurement-825 (Retrieved 06/22/2015).

[Huy13] Huy Dinh. *VGU.EXCELSIOR - Ultrasonic 3D Scanner - Vietnam MCU Contest 2013*. 2013. URL: https://www.youtube.com/watch?v=saWWhEYQxTg (Retrieved 06/22/2015).

[Sol12] Jan Erik Solem. *Programming Computer Vision with Python*. Sebastopol, CA 95472: O'Reilly, 2012.

[Ble15a] Blender Foundation. *Re-branding Blender - blender.org - Home of the Blender project - Free and Open 3D Creation Software*. 2015. URL: https://www.blender.org/press/re-branding-blender/ (Retrieved 06/24/2015).

[Bal+14] Caterina Balletti et al. "Calibration of Action Cameras for Photogrammetric Purposes". In: *Sensors* 14.9 (2014), p. 17471. ISSN: 1424-8220. DOI: 10.3390/s140917471. URL: http://www.mdpi.com/1424-8220/14/9/17471.

[Wik15c] Wikipedia. *Triangulation — Wikipedia, The Free Encyclopedia*. 2015. URL: https://en.wikipedia.org/w/index.php?title=Triangulation&oldid=663507003 (Retrieved 06/25/2015).

[New+11] Richard A. Newcombe et al. "KinectFusion: Real-time Dense Surface Mapping and Tracking". In: *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*. ISMAR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 127–136. ISBN: 978-1-4577-2183-0. DOI: 10.1109/ISMAR.2011.6092378. URL: http://dx.doi.org/10.1109/ISMAR.2011.6092378.

[Ric14] Ricardo Zamora. *Google Maps 3D*. 2014. URL: https://www.youtube.com/watch?v=5iolZU8LwPU (Retrieved 06/22/2015).

[F414] F4. *F4map Demo - Interactive 3D map*. 2014. URL: http://demo.f4map.com/#lat=49.4559869&lon=11.0762814&zoom=18 (Retrieved 06/22/2015).

[Ope15] OpenStreetMap Wiki. *Indoor Mapping – OpenStreetMap Wiki*. 2015. URL: http://wiki.openstreetmap.org/wiki/Indoor_Mapping (Retrieved 06/22/2015).

[She10] Shen, Shaojie and Michael, Nathan and Kumar, Vijay. *Autonomous Aerial Navigation in Confined Indoor Environments*. 2010. URL: https://www.youtube.com/watch?v=IMSozUpFFkU (Retrieved 06/22/2015).

[Geo14] Georgina Voss. *DevArt: Google's powerful new move to arts patronage | Science | The Guardian*. 2014. URL: http://www.theguardian.com/science/political-science/2014/feb/28/devart-googles-powerful-new-move-to-arts-patronage (Retrieved 06/25/2015).

[Wik15d] Wikipedia. *Greco-Roman world — Wikipedia, The Free Encyclopedia*. 2015. URL: https://en.wikipedia.org/w/index.php?title=Greco-Roman_world&oldid=654515334 (Retrieved 06/25/2015).

[Wik15e] Wikipedia. *Renaissance architecture — Wikipedia, The Free Encyclopedia*. 2015. URL: https://en.wikipedia.org/w/index.php?title=Renaissance_architecture&oldid=665879352 (Retrieved 06/25/2015).

[Wik15f] Wikipedia. *German Renaissance — Wikipedia, The Free Encyclopedia*. 2015. URL: https://en.wikipedia.org/w/index.php?title=German_Renaissance&oldid=645400210 (Retrieved 06/26/2015).

[Mäh00] Wolfgang Mährle. *Academia Norica*. Berlin: Franz Steiner Verlag, 2000, pp. 39–40. URL: http://www.steiner-verlag.de/titel/51429.html (Retrieved 03/11/2015).

[Alt13] Altstadtfreunde e.V. *Pellerhaus Magazin 02. Ein Magazin der Altstadtfreunde zum Wiederaufbau des Pellerhofes*. German. 2013.

[Alt12] Altstadtfreunde e.V. *Pellerhaus Magazin 01. Ein Magazin der Altstadtfreunde zum Wiederaufbau des Pellerhofes*. German. 2012.

[Alt] Altstadtfreunde e.V. *Wiederaufbau des Pellerhofes*. German.

[Alt14] Altstadtfreunde e.V. *Pellerhaus Magazin 03. Ein Magazin der Altstadtfreunde zum Wiederaufbau des Pellerhofes*. German. 2014.

[Wik15g] Wikipedia. *Panorama — Wikipedia, The Free Encyclopedia*. 2015. URL: https://en.wikipedia.org/w/index.php?title=Panorama&oldid=665030494 (Retrieved 06/27/2015).

[Wik15h] Wikipedia. *Circle-Vision 360 — Wikipedia, The Free Encyclopedia*. 2015. URL: https://en.wikipedia.org/w/index.php?title=Circle-Vision_360%C3%82%C2%B0&oldid=666618020 (Retrieved 06/27/2015).

[Kol15] Kolor Team. *Kolor | Image-stitching, virtual tour, 360-degree video experts*. 2015. URL: http://www.kolor.com/ (Retrieved 06/27/2015).

[CGM15] CGMeetup. *Google ATAP 'HELP' Behind The ScenesComputer Graphics and Digital Art Community for Artist: Job, Tutorial, Art, Concept Art, Portfolio*. 2015. URL: http://www.cgmeetup.net/home/google-atap-help-behind-the-scenes/ (Retrieved 06/27/2015).

[Car14] Carlos Alberto Furuti. *Map Projections: Mapping Definitions and Concepts*. 2014. URL: http://www.progonos.com/furuti/MapProj/Normal/CartDef/MapDef/mapDef.html (Retrieved 06/27/2015).

[Hou+15] HamidReza Houshiar et al. "A Study of Projections for Key Point Based Registration of Panoramic Terrestrial 3D Laser Scans". In: (2015), pp. 11–31. DOI: 10.1080/10095020.2015.1017913.

[She02] Jonathan Richard Shewchuk. "Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery". In: *In Eleventh International Meshing Roundtable*. 2002, pp. 193–204.

[Wik15i] Wikipedia. *Coordinate system — Wikipedia, The Free Encyclopedia*. 2015. URL: https://en.wikipedia.org/w/index.php?title=Coordinate_system&oldid=662627300 (Retrieved 06/29/2015).

[Gsc+11] Michael Gschwandtner et al. "BlenSor: Blender Sensor Simulation Toolbox Advances in Visual Computing". In: ed. by George Bebis et al. Vol. 6939. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2011. Chap. 20, pp. 199–208. ISBN: 978-3-642-24030-0. DOI: 10.1007/978-3-642-24031-7_20. URL: http://dx.doi.org/10.1007/978-3-642-24031-7%5C_20 (Retrieved 05/31/2015).

[Sta14] Stanford Computer Graphics Laboratory. *The Stanford 3D Scanning Repository*. 2014. URL: http://graphics.stanford.edu/data/3Dscanrep/ (Retrieved 06/28/2015).

[PH10] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. 2nd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010. ISBN: 0123750792, 9780123750792.

[Bak09] Jeroen Bakker. *The mystery of the blend*
The blender file-format explained. 2009. URL: http://www.atmind.nl/blender/mystery_ot_blend.html (Retrieved 03/10/2015).

[Ble14b] Adam Kalisz (BlenderEi). *[C++/Qt]: Export .blend with procedurally generated mesh?* 2014. URL: http://www.blenderartists.org/forum/showthread.php?362865-C-Qt-Export-blend-with-procedurally-generated-mesh (Retrieved 11/03/2015).

[Enz14] Ivan Enzhaev. *Qt C++ OpenGL GLSL (parts 1 to 8)*. 2014. URL: https://www.youtube.com/watch?v=jTHvFGjy5uA&list=PLHlHFavas54e0-05vZcZ0wNGhnpRyFmLc (Retrieved 03/10/2015).

[Owe15] Owen Blacker. *Freedom of Panorama is under attack — Vantage — Medium*. 2015. URL: https://medium.com/vantage/freedom-of-panorama-is-under-attack-6cc5353b4f65 (Retrieved 07/10/2015).

[Chr15] Chris Cheesman. *European Parliament rejects 'absurd' EU plan to axe Freedom of Panorama - Amateur Photographer*. 2015. URL: http://www.amateurphotographer.co.uk/latest/photo-news/european-parliament-rejects-absurd-eu-plan-to-axe-freedom-of-panorama-55708 (Retrieved 07/10/2015).

[Eic15] Anja Eichhorn. "Selbstständige Navigation eines autonomen Roboters mittels 3D-Tiefensensor". German. Oct. 16, 2015.

[Sha15] Shapeways Inc. *3D Printing Materials Comparison Sheet*. 2015. URL: http://www.shapeways.com/materials/material-options (Retrieved 07/01/2015).

[Wik15j] Wikipedia. *Lenticular printing — Wikipedia, The Free Encyclopedia.* 2015. URL: https://en.wikipedia.org/w/index.php?title=Lenticular_printing&oldid=657839036 (Retrieved 07/01/2015).

[Tim14] Timetraveler augmented Ltd. *Timetraveler The Berlin Wall App.* 2014. URL: https://www.youtube.com/watch?t=75&v=CY9f6UJZlmM (Retrieved 07/01/2015).

[KJ12] Hans-Michael Körner and Bruno Jahn. *Große Bayerische Biographische Enzyklopädie.* Berlin: De Gruyter, 2012, p. 2128. URL: http://www.degruyter.com/view/product/62595 (Retrieved 03/11/2015).

[Sha13] ShareLaTeX. *Youtube: How to Write a Thesis in LaTeX (parts 1 to 5).* 2013. URL: https://www.youtube.com/watch?v=gFAQd0ueIMc&index=1&list=PLCRFsOKSM7eNGNghvT6QdzsDYwSTZxqjC (Retrieved 01/23/2015).

[Vel15] Claudio Vellage. *LaTeXTutorial – Interactive Lessons, Code Examples – For Beginners:* 2015. URL: http://www.latex-tutorial.com/ (Retrieved 01/23/2015).

[Far15] Faro. *SCENE LT.* 2015. URL: http://www.faro.com/faro-3d-app-center/stand-alone-apps/scene-lt (Retrieved 07/01/2015).

[Ble15b] Blender Foundation. *blender.org - Home of the Blender project - Free and Open 3D Creation Software.* 2015. URL: http://blender.org (Retrieved 07/01/2015).

[Pao14] Paolo Cignoni. *MeshLab.* 2014. URL: http://meshlab.sourceforge.net/ (Retrieved 07/01/2015).

[Cha15] Changchang Wu. *VisualSFM : A Visual Structure from Motion System.* 2015. URL: http://ccwu.me/vsfm/ (Retrieved 07/01/2015).

[Jan12] Jancosek, Michal and Pajdla, Tomas. *CMP SfM Web Service.* 2012. URL: http://ptak.felk.cvut.cz/sfmservice/websfm.pl?menu=cmpmvs (Retrieved 07/01/2015).

[Agi14] Agisoft LLC. *Agisoft PhotoScan.* 2014. URL: http://www.agisoft.com (Retrieved 07/01/2015).

[The15] The Qt Company. *Qt | Cross-platform application and UI development framework.* 2015. URL: http://www.qt.io (Retrieved 07/01/2015).

[Khr15] Khronos Group. *OpenGL - The Industry Standard for High Performance Graphics.* 2015. URL: https://www.opengl.org (Retrieved 07/01/2015).

Source code and electronic material:

Additional material for this project, such as source code or this paper, are hosted in a public SVN repository online. It can be checked out via:

```
svn co svn://schorsch.efi.fh-nuernberg.de/pc2blender
```

The complete documentation, videos, tutorials, errata, contact information, and more is available on the official project website:

```
http://bachelor.kalisz.co
```