Surface Reconstruction and Stereoscopic Video Rendering from Laser Scan Generated Point Cloud Data

Hans Martin Langø
Morten Tylden

Master of Science in Computer Science
Submission date: June 2007
Supervisor: Torbjørn Hallgren, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

With a basis of a laser scanned point cloud, this project will be an in-depth study on methods of post-processing of the data.

Different methods/algorithms will be reviewed and analysed, through both own implementations and proprietary applications, and the most promising ones will be combined to create the final data model.

The main purpose of the project is to create a surface model that is both lightweight enough to be run on low-end systems, while still keeping a high level of detail where applicable. This model will be created for demonstration purposes, which will be achieved through a stereoscopic movie.

Assignment given: 21. January 2007
Supervisor: Torbjørn Hallgren, IDI

Abstract

This paper contains studies about the process of creating three-dimensional objects from point clouds.

The main goal of this master thesis was to process a point cloud of the Nidaros Cathedral, mainly as a pilot project to create a standard procedure for future projects with similar goals.

The main challenges were two-fold; both processing the data and creating stereoscopic videos presenting it.

The approach to solving the problems include the study of earlier work on similar subjects, learning algorithms and tools, and finding the best procedures through trial and error.

This resulted in a visually pleasing model of the cathedral, as well as a stereoscopic video demonstrating it from all angles.

The conclusion of the thesis is a pilot project demonstrating the different operations needed to overcome the challenges encountered during the work. The focus have been on presenting the procedures in such a way that they might be used in future projects of similar nature.

Preface

This master thesis is the result of the diploma work of Hans Martin Langø and Morten Tylden, in association with the *Department of Computer and Information Science (IDI)* at the *Norwegian University of Science and Technology (NTNU)* spring 2007. The master thesis consists of this report, several digital models, and stereoscopic videos presenting the models.

The project was defined in collaboration with Torbjørn Hallgren at IDI, and is a continuation of the project *3D-Scanning of the Nidaros Cathedral*, carried out during the autumn of 2006 by the same team. The goal was to continue working with the data acquired there and improve on it visually.

We would like to direct special thanks to

- Torbjørn Hallgren, the teaching supervisor and our guide throughout the whole project.
- Knut Ragnar Holm, for help with both logistics and scanning/measuring.
- Kristin S. Bjørlykke, for providing us with access and support during the scanning.
- Henrik Mogens Gundersen, his expertise and help with visualization proved invaluable.

Trondheim 18.06.2007

Hans Martin Langø   Morten Tylden

Contents

List of Figures

Chapter 1

Introduction

This project is based on earlier work done on the '3D-Scanning of the Nidaros Cathedral' project [26]. In this project some of the exterior parts of the Nidaros Cathedral were scanned by a laser scanner, and the resulting point cloud was demonstrated through a series of short videos. Most of the work and research carried out by the former project served as a preparation for the pilot project detailed in this report. For more extensive documentation, we refer to the *3D-Scanning of the Nidaros Cathedral* project documentation [26].

1.1 Problem Statement

With a basis of a laser scanned point cloud, this project will be an in-depth study on methods of post-processing of the data.

Different methods/algorithms will be reviewed and analysed, through both own implementations and proprietary applications, and the most promising ones will be combined to create the final data model.

The main purpose of the project is to create a surface model that is both lightweight enough to be run on low-end systems, while still keeping a high level of detail where applicable. This model will be created for demonstration purposes, which will be achieved through a stereoscopic movie.

1.2 Objectives

In order to reach the goals described above, certain objectives had to be met. They are all smaller parts of the whole project, and most of them could be met individually.

These objectives can be summarised into five points:

Removal of noise and inaccurate data. A noisy model results in a poor quality of model demonstrations. Inaccurate measurements can create problems both for the restauration team and future work on the model.

Merging the data sets. Different point clouds stored as independent data sets must be merged in a manner that makes for a smooth transition in the overlapping areas.

Creation of 'watertight' surfaces. This includes the creation of polygons based on point clouds, as well as the filling of holes in the model due to lack of scanner coverage. This is important both for calculations and processing of the data, as well as for presentations.

Making the model 'light-weight'. Creating a lighter model reduces the hardware requirements for further processing and usage of the data. Some applications have problems reading large data files, so reducing the models decreases the amount of complications.

Demonstrations of the cathedral. The finished model should be demonstrated visually, preferably through a stereoscopic movie that shows the cathedral from all angles.

These objectives are based on the research from the '3D-Scanning of the Nidaros Cathedral' project [26], and have been further developed based on the possibilities and limitations of current tools (Chapter 2).

Chapter 3 describes how this project accomplishes these goals, by providing solutions based on current tools.

1.3 Scope

The scope of the project is based on the objectives defined earlier in this chapter.

The main focus will be on creating a model that is visually pleasing, and demonstrating it through a stereoscopic video. This implies using available software tools when possible, instead of manually doing the basic operations during the course of this project.

1.4 Structure

The report is split into chapters five main chapters and three appendix sections.

Chapter 1 is this chapter, the introduction containing the background motivation, objectives and scope of the project.

Chapter 2 describes the state of the art regarding both related projects, methods and algorithms, software tools, and storage formats.

Chapter 3 explains the different procedures and their functions, regarding experimentation, implementation and processing of the data.

Chapter 4 shows the results of the work done during the scope of Chapter 3.

Chapter 5 contains the conclusion.

Chapter 6 details the discussion and outlines possible further work.

Appendix A shows the hardware used during the project.

Appendix B lists all software used during the project, as well as its main functions.

Appendix C illustrates the data flow.

Chapter 2

State of the art

2.1 Overview

This chapter describes the state of the art regarding processing of scanned data and general reverse engineering of 3D objects.

Purpose

Many different tools have been developed and utilized for both reverse engineering and general geometric calculations. They are used in the different phases of the process and accomplish important tasks.

By analysing these tools in detail and how they have been used in other projects, one can achieve a greater understanding of:

- How they work.
- What results they produce.
- In which phases they are most useful.

Scope

This chapter will take a closer look on earlier projects of similar nature, and three different important parts of their data processing; the fundamental algorithms, the software tools using these algorithms, and the different storage formats used by different software tools.

Structure

This chapter is split into four sections, describing the different aspects defined in the scope. Each of them has their own summary at the end of each section, giving more specific details of their importance to this project.

- Chapter 2.2 features some of the projects that inspired this master's thesis.
- Chapter 2.3 analyses methods and algorithms used in these projects.
- Chapter 2.4 gives an overview of the different tools available, and their main features.
- Chapter 2.5 describes the different storage formats and what they offer.

2.2 Projects

Many other projects with similar goals have been carried out during the last decade, and their experiences were useful for planning the data processing. This section will not cover the projects in great detail, but rather feature some of the work that inspired this pilot project, such that interested parties may read more detailed accounts.

2.2.1 Virtual Heritage

The Virtual Heritage Project is a collection of projects where the common goal is the digital and physical preservation of cultural heritage [32]. The projects have been gathering, processing, and presenting the data since the repository was made available in 1997 [33].

The tools used in these projects include everything from laser scanners, photogrammetic equipment, high-resolution digital cameras, to 'manual' digital reconstruction based on ruins and imitations of the original monuments. Many different kinds of software tools have been utilized during this work, providing this pilot project a reference list of their possible applications [32] [26].

All data are distributed unrestricted through the online repositories, free to be further processed or used by other interested parties [32]. Both this master's thesis and the other projects described here are at least partially affiliated with the Virtual Heritage Project, and have similar goals regarding the preservation of history for future generations.

2.2.2 The Stanford 3D Scanning Repository

The Stanford 3D Scanning Repository is the result of a series of projects performed at Stanford University, where the goal was to scan and process a series of sculptures. The repository contains everything from raw range data to complete reconstructions of the objects [40].

The most well-known of these projects is the Digital Michelangelo Project [28], that scanned several historical sculptures to store them for future generations as part of the Virtual Heritage project [32]. Another known project is the Stanford Bunny, which is often referenced by other projects as an example of multiple scan modeling [41].

The Stanford projects utilize commonly available software tools, and store all their data in the PLY, VRML and 3DS formats [40] (see Chapter 2.5 for details).

2.2.3 Sverresborg Tour in 3D

Several projects have been working on reconstructing the Sverresborg fortress the way it was in the thirteenth century, mostly as work for the local museum or NTNU [3] [26]. Based on these projects, the goal was to create a virtual world that included both the castle, the people living there, and the environment, both visually and through sound.

The Sverresborg Tour project utilized earlier reconstruction projects to get a complete model of the fortress and the surrounding area. Based on accounts from the local museum, historically accurate models of men, women and children were constructed and animated to simulate daily life in the area. A narrator was added to explain what the guided tour showed [3].

The software tools used in this project were mainly Performer and 3D Studio Max [4] (see Chapter 2.4.5 for details). With these tools, a tour through the fortress was rendered as a stereoscopic movie, which the museum presented to its visitors [3].

2.2.4 Summary

These three projects served both as an inspiration and as a source of information regarding which methods to use for the processing phases (Chapter 3). Some of the tools used during these projects were used in this thesis too (see Chapter 2.4 for details).

2.3 Methods and Algorithms

This section describes three of the most important algorithms used during reverse engineering projects (Chapter 2.2.

They are implemented in different geometric tools [44] [53], and are available both through open source libraries [14] and as part of software packages [17] [24].

2.3.1 Delaunay Triangulation

Delaunay Triangulation is a way of triangulating a set of points **P**, so that the triangulation DT(**P**) has no point **P** inside the circumcircle of any of the triangles in DT(**P**) [44].

In other words, a circle drawn through each of the three points comprising a Delaunay triangle will have no other points inside this circle. If a fourth point is inserted inside the circle, new triangulations have to be calculated for all of the four points, which in this case results in two or three triangles instead of only one.

An example is shown in Figure 2.1, where a set of random points have been triangulated using Delaunay Triangulation. Each triangle in this picture has an 'imaginary circle' going through three points, where there are no other points inside.

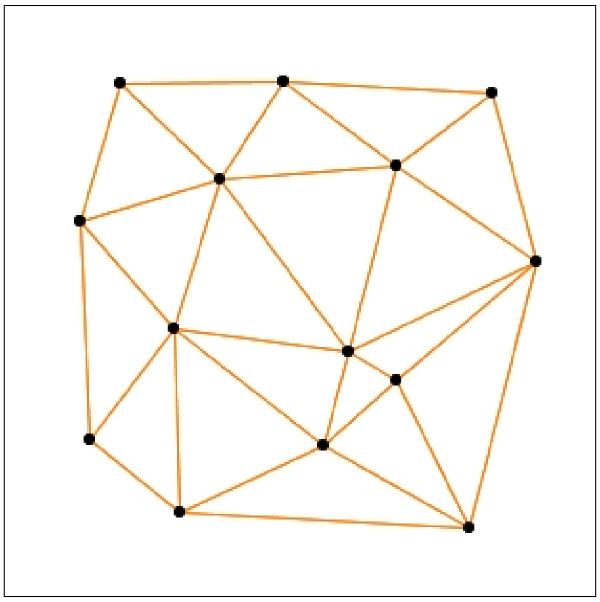

Figure 2.1: *Delaunay triangulation of a random set of points.*

Delaunay Triangulation works not only in 2D, but in 3D and any higher number of dimensions [14]. The difference in three dimensions from two dimensions, is that tetrahedas and spheres replace triangles and circles as the basis for triangulations.

Because of its simplicity and efficiency it is very commonly used in geometric calculations, often in combination with Voronoi Diagrams (Chapter 2.3.2) and other algorithms. A more detailed graphical explanation of how the circles and triangles are bound together has been made by Paul Chew[10].

2.3.2 Voronoi Diagrams

A Voronoi Diagram is a *special kind of decomposition of a metric space determined by distances to a specified discrete set of objects in the space, e.g., by a discrete set of points*[53].

In layman terms, the Voronoi Diagram divides the space in cells around the points, where the boundaries between the cells are defined as having the same distance to the points on both sides.

Figure 2.1 shows the Voronoi Diagram based on the same set of random points as the one featured for the Delaunay Triangulation (Chapter 2.3.1).

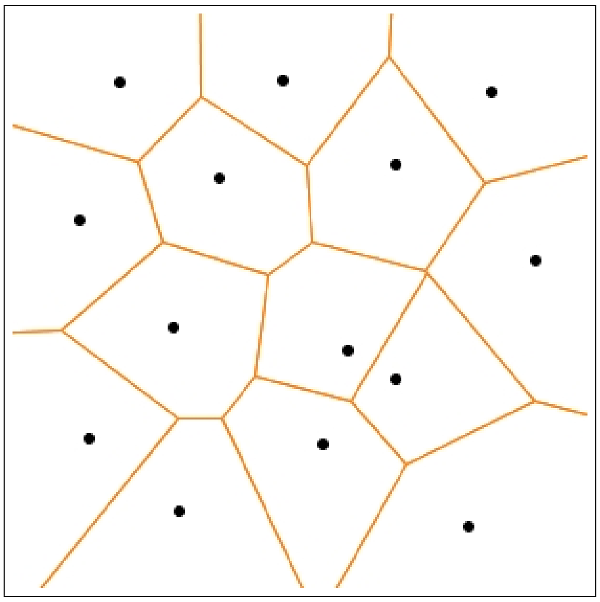

Figure 2.2: *Voronoi Diagram of a random set of points.*

Figure 2.3 shows a comparison of the Delaunay Triangulation and the Voronoi Diagram, where they both operate on the same set of random points.

Just like as with Delaunay Triangulation, a graphical comparison and explanation of the Voronoi diagram has been made by Paul Chew[10]

2.3.3 Edge Collapse

The Edge Collapse algorithm is a *surface simplification technique*, used for finding the edges in the model that contributes the least to the visual impression of the model [22].

It works by iteratively finding and collapsing the chosen edges, deleting the vertex and the closest elements, and stretching the remaining elements [5].

Figure 2.4 shows how the edge collapse algorithm works by comparing vertexes before and after the transformation.

Note that when the vertex collapse, two triangles are removed, which is what usually happens.

2.3.4 Summary

The algorithms described in this chapter are an integral part of the reverse engineering process, being the foundation this master's thesis' data process-

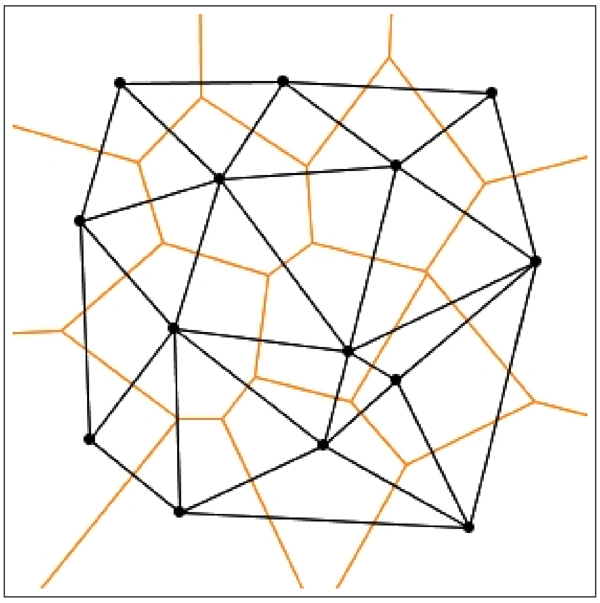

Figure 2.3: *Delaunay Triangulation and Voronoi Diagram of a random set of points.*

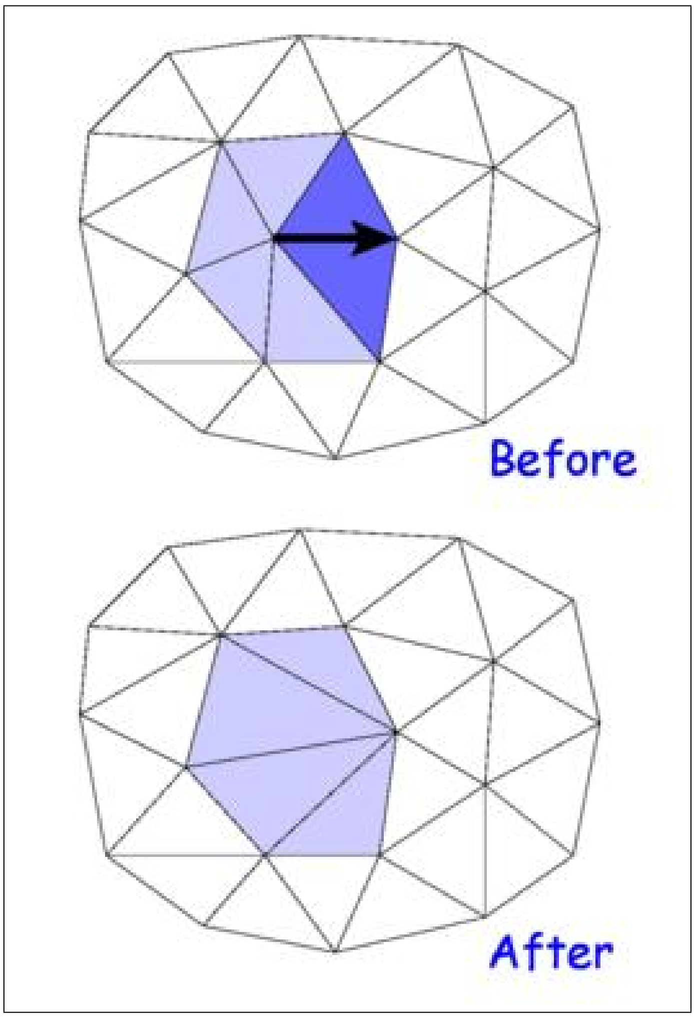

Figure 2.4: *A vertex grid before and after applying Edge Collapse.[5]*

ing is based upon. Some of them were implemented through code generated during this project, while the rest were accessible through external (modifiable) applications (see Chapter 2.4).

2.4 Software Tools

This section gives a brief overview of the different software tools used in this project, as well as a short description of their most important features.

2.4.1 RiSCAN PRO

RiSCAN PRO is the software bundled with the scanner that provided the data for this project, and it has both field work and post-processing utilities [26]. It is developed and maintained by RIEGL Laser Measurement Systems[36], and the current version is 1.4.1 (May 2007).

RiSCAN PRO's main focus is the gathering of data, not the processing of it afterwards [35]. However, since it has all the raw scan data and additional unique information that are not exported to other tools, some of its features were used.

Post-processing features offered by RiSCAN PRO include: [35]

- Mesh generation from point clouds (Delaunay Triangulation, see Chapter 2.3.1).
- Attributing colour information to every data set.
- Generation of images for mesh texturing.
- Point cloud decimation.
- Object construction from point clouds.
- Global data registration based on external measurements.

RISCAN PRO also supports a wide variety of formats and standards for exporting the data [37], depending on the required properties. This makes transferring the data to other processing software easier.

2.4.2 The Computational Geometry Algorithms Library

The Computational Geometry Algorithms Library (CGAL) is a collection of geometric algorithms that have been implemented as a C++ library. It is open source and supported by most common platforms (Windows, Mac, Linux, Solaris, and Irix)[14].

According to the official site [14], CGAL offers:

- Triangulations (2D constrained triangulations and Delaunay triangulations in 2D and 3D).
- Voronoi diagrams (for 2D and 3D points, 2D additively weighted Voronoi diagrams, and segmented Voronoi diagrams).
- Boolean operations on polygons and polyhedra.
- Arrangements of curves.
- Mesh algorithms (2D Delaunay mesh generation and 3D surface mesh generation, surface mesh subdivision and parameterization).
- Alpha shapes.
- Convex hull algorithms (in 2D, 3D and dD).
- Operations on polygons (straight skeleton and offset polygon).
- Search structures (kd trees for nearest neighbor search, range and segment trees).
- Interpolation (natural neighbor interpolation and placement of streamlines).
- Optimization algorithms (smallest enclosing sphere of points or spheres, smallest enclosing ellipsoid of points, principal component analysis).
- Kinetic data structures.

All of these algorithms are modifiable, allowing utilization of them regardless of the storage format (see Chapter 2.5).

The main reason for using this tool is that it is open source, making it easily modified to suit the needs of the project. Its use has also been documented by many similar projects [14].

2.4.3 Rational Reducer

Rational Reducer (RR) is an application created by Systems in Motion (SIM) [23], that *reduces the number of polygons in 3D models, while preserving visual quality*[24].

The main use of Rational Reducer is to reduce the complexity of 3D models and scenes, resulting in:

- Better performance for real-time rendering.
- Smaller files, making model transfer faster and data storage more efficient.

Rational Reducer is used by companies such as NASA, the US Army, Toyota, Airbus, Statoil and Norsk Hydro [23]. The reason for its popular use is that the automatic reduction of models achieved through RR only takes a fraction of the time it takes to do manual reduction [22].

Rational reducer works by doing iterations of the Edge Collapse algorithm (Chapter 2.3.3) until only a predetermined number of triangles remain. This algorithm can be modified through four different parameters: [22].

- *Edge length* determines how much effort is put into reducing small geometry details first.
- *Curvature* decides the degree in which areas with high curvature are conserved.
- *Sharpness* controls how the feature edges (sharp edges) in the model are kept.
- *Material* defines the preservation of boundary lines between different materials and textures.

2.4.4 Geomagic Studio

Geomagic Studio is a processing tool developed by Geomagic, designed for general post-processing and reverse engineering of scanned data. It was first released in 1999, with the current version being v9.2 (May 2007)[17].

Geomagic Studio is mainly used by engineers and designers working in the fields of *Aerospace, Automotive, Consumer Products, Turbines and Power Generation, Heavy Industry and Medical Devices*[17]. The main reason for its widely spread use is its versatility; it is compatible with most modern scanners, cameras and computers.

Geomagic Studio is actually a set of different tools merged into one package, and offers among other things: [17]

- Scan Registration Tools (transformation matrixes, merging and alignment of scans, etc).
- Point Processing (data reduction, noise elimination, etc).
- Polygon Creation and Repair.
- Polygon Editing and hole filling.

- NURBS Surface Creation.
- Analysis (distance measurements, gravity calculations and curvature calculations).
- Color Support (texture maps and per-vertex colour).

This project uses Geomagic Studio because it is the most complete data processing tool available, and performs a wide array of geometric tasks. See Chapter 3 for more details regarding its use.

2.4.5 3D Studio Max

3D Studio Max is a 3D animation program developed for the Windows platform by Autodesk[4], but was originally created for the DOS platform by the Yost Group[43]. After being purchased by Autodesk the product has gone through many iterations through the years, with the current version being v9.0 (released August 2006)[43].

3D Studio Max is widely used by professionals to create content for video games, TV commercials and architectural design. In addition it is utilized to create special effects in movies [43].

Some of the most important features of 3D Studio Max includes: [4] [43]

- Advanced shaders (ambient occlusion and subsurface scattering).
- Dynamic simulation.
- Particle systems.
- Normal map creation and rendering.
- Global illumination algorithms (like Radiosity).

3D Studio Max utilizes several different storage formats (Chapter 2.5), including VRML, 3DS and PLY. This makes both textured surfaces and coloured polygons from other applications easy to import, despite varying support for the different file formats.

The main reason for using 3D Studio Max in this project is to render a stereoscopic movie (Chapter 4.2) to, present the model in three dimensional space.

2.4.6 AccuTrans 3D

AccuTrans 3D is a software application that provides translation of 3D models between different file formats. It is developed and maintained by Micro-

Mouse Productions, and supports a wide variety of operating systems and file formats [34].

AccuTrans 3D was originally made (in 1993) to convert between only a few file formats, but due to its popularity and ease of use, it has grown into a multi-purpose tool [34].

When translating between formats, AccuTrans 3D retains the following properties: [34]

- Positional and rotational information.
- Colour.
- Index of refraction.
- Reflection.
- Specularity.
- Phong shading.
- Textures.
- UV coordinates.

AccuTrans 3D is the software used in this project for translation between the different file formats, as well as for merging of data sets (Chapter 3.10).

2.4.7 Windows Media Encoder

Windows Media Encoder is a software application developed by Microsoft [30]. Its main purpose is to convert or encode both audio and video content to the Windows Media Video (WMV) format [54] (Chapter 2.5.4).

WME has many features, but the most important ones regarding this project concerns the properties for its Dual Stream encoding: [1]

- Support for dual core CPUs.
- Efficient compression of videos with high complexity.
- Supports usage by 2D software.

Windows Media Encoder is used in this project during the creation of the stereoscopic videos [1]. See Chapter 4.2 for the results.

2.4.8 Summary

The software tools described in this section were used during the implementation and processing phase (Chapter 3). Most of them accomplish one important task, but Geomagic Studio (Chapter 2.4.4) has a wider range of tools and was used to achieve several goals.

The reason for using so many different tools is that no single tool was able to carry out all the tasks in a satisfactory manner.

2.5 Formats and Storing

There are many ways to format and store the data, depending on its purpose, and the tools described in Chapter 2.4 have different needs and preferences.

This section explains the different standards, and presents a summary of which formats were used and why they were utilized.

2.5.1 VRML

Virtual Reality Modeling Language (VRML) is a hierarchic text based file format for defining and presenting 3D computer-generated graphics, 3D sound and hypermedia links. It was designed by the Web3D Consortium particularly for the World Wide Web, and have been one the most popular file formats since its specification in 1994 [52] [29] [6] [13].

VRML has been developed and improved in several iterations through the years, with VRML 1.0 (1994) and VRML 2.0 (1997) being the main versions [52]. VRML 2.0 has several improvements over VRML 1.0, however, VRML 1.0 is still used because of its simplicity and ease of use [29].

VRML offers several features for polygon surfaces, including *surface colour*, *image-mapped textures*, *shininess* and *transparency* [52]. Interaction with objects defined in VRML can be achieved through animation, sound and lightning (primarily in VRML 2.0), which most often appear as scripted events [52].

The Web3D Consortium made a successor to VRML 2.0 called X3D, which is backward compatible with almost all VRML files [12]. However, since most tools used in this project (see Chapter 2.4) were compatible with VRML and not with X3D, VRML was used.

2.5.2 PLY

Polygon File Format (PLY) is a computer file format designed to store three dimensional data from 3D scanners [48]. PLY has two sub-formats, depending on the needs of the user. The first one is an ASCII representation, with the advantage of being easily used and customised. The second version is binary storage, yielding more compact storage and faster saving and loading [8].

PLY describes models as a collection of faces, vertices and similar elements, and each PLY file contains only one object. Other information can be stored along with the object, including: *colour*, *surface normals*, *texture coordinates*, *transparency*, *range data confidence*, and different properties for the front and back of a polygon [8].

2.5.3 3DS

3DS is one of the standard exchange formats used by 3D Studio Max (Chapter 2.4.5), and was developed by Autodesk [4][22]. It is supported by a wide variety of other applications, and is one of the most popular formats for storing animated data in both the movie and the gaming industry [43] [19].

The 3DS format contains: [15]

- Mesh data.
- Material attributes.
- Bitmap references.
- Smoothing group data.
- Viewport configurations.
- Cameras and lighting information.
- Object animation data.

The 3DS file format is divided into blocks of data called ‘chunks’. Each of these chunks contains an ID and length description, and they store the shapes, lighting, and viewing information that together represent the three-dimensional scene [15].

The main problem with 3DS is that it is an outdated format, and among other things lacks support for meshes with more than 65536 polygons[19]. This leads to problems when converting large models between different formats.

2.5.4 WMV

Windows Media Video is a generic name for a set of video codecs developed by Microsoft [54] [30]. Although they were originally made as codecs for low-bitrate streaming tools, it is now an open proprietary standard used for playing back audio/video both from streaming media and local media [54].

The latest version (WMV9) has several advanced features, and the most important reasons for using it in this project include: [54] [50]

- Bit rates up to 135 Mbit/s.
- Compatibility with a wide range of hardware and software (HD-DVD, Blu-ray, XBOX 360, etc).
- Free implementation and easy usage.
- Supports DRM protected content.

2.5.5 Summary

These are just a small sample of the formats available, and they all have different advantages and drawbacks regarding storage possibilities, efficiency, properties, and compatibility with other formats. In order to reduce complexity, only these formats were used.

VRML was used during the most stages of the processing, since its popularity has ensured it to be supported by a wide range of tools. The main reason VRML was chosen is because it supports both point clouds and polygon mesh. However, there were some compatibility issues regarding different versions of VRML, so both VRML 1.0 and VRML 2.0 were utilized.

PLY was used for the final model, since it is easier to import large data sets into 3D Studio Max with the PLY format than the VRML format.

WMV was chosen as the media for storing the stereoscopic video, both due to its innate support for dual output, and because its compression algorithm preserves the video quality fairly well.

Chapter 3

Implementation and Processing

3.1 Overview

This chapter describes both the implementation done during the course of this project, as well as the general processing of data with the help of the tools described in Chapter 2.4.

Purpose The main purpose of this chapter is to give an understanding of what happens during each process. This will be achieved through both textual explanation and illustrations from the software tools as an example of how the objectives can be met.

This is neither meant to be the only or best solution, nor to provide the 'best' order of sequence that will work for all similar projects. Instead it is meant to be a pilot project investigating different ways of overcoming some of the common obstacles related to reverse engineering.

Scope The scope encompasses all the use of tools, their adjustments and modifications, implementations, and the various experiments performed during processing.

Structure The chapter is split into sections that each depicts an important phase of the total process:

- Chapter 3.2 details the overall research and planning of the processing operations.

- Chapter 3.3 describes how the raw data was prepared.
- Chapter 3.4 explains noise reduction.
- Chapter 3.5 shows the different methods of triangulation.
- Chapter 3.6 depicts the polygon reduction.
- Chapter 3.7 illustrates crossover removal.
- Chapter 3.8 tells how gaps in the models were filled.
- Chapter 3.9 is about straightening edges.
- Chapter 3.10 displays different ways of merging data sets.
- Chapter 3.11 describes the use of textures.
- Chapter 3.12 details the creation of a stereoscopic movie.
- Chapter 3.13 discusses perspective correction related to 3D rendering.
- Chapter 3.14 is a short summary of the processing phases.

3.2 Research and Planning

There are many steps required to complete the work of improving the model and creating a stereoscopic video. In order to not stray too far from the objective underway, careful planning is needed.

The sequence of tools and operations chosen is built on research [20] [2] [31] [38] [16], earlier work done by the project team [26], and trial and error during the implementation phase. Although different variations of algorithms, tools and parameters were tried on all the steps described below, the operations presented in this chapter were the ones that resulted in the best visual results. The sequence they are described in reflects the order in which the operations were executed in the final plan.

3.3 Data Preparation

Raw scan data usually consists of several point clouds with accompanying colour images. In this case it was a combination of data from the Scanning of the Nidaros Cathedral project [26], and data acquired during the scope of this thesis.

Before the main work begins, the raw data has to be processed and prepared properly. RiSCAN PRO (Chapter 2.4.1) was selected for this part due to prior experience with the tool, but almost any modern post-processing tool could accomplish the same results. A quick explanation of this work will be detailed here, but for a more in-depth description, see the project *Scanning of the Nidaros Cathedral* [26].

3.3.1 Model Trimming

Raw scan data always contains a lot of unnecessary information, often including the area surrounding the object and overlapping scans. The first step should therefore be cutting away all unnecessary details, leaving only the parts needed for the surface model.

The result of this operation is a smaller object that requires far less resources, considering both processing power and storage capacity. This affects all future work on the models.

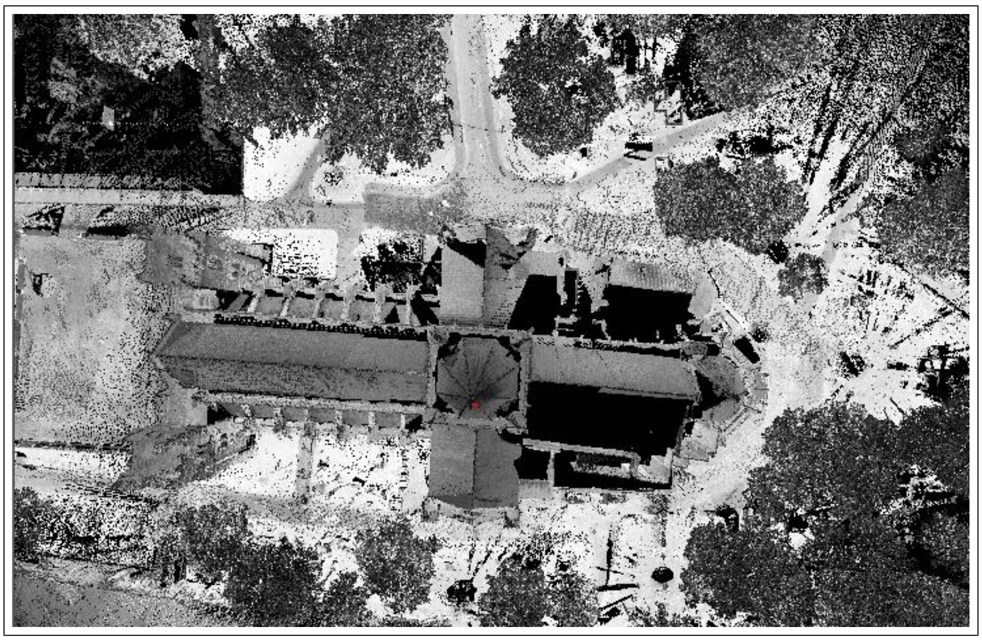

Figure 3.1: *Raw and unprocessed point clouds.*

Figure 3.1 and Figure 3.2 gives an overview of the amount of data the point clouds consists of before and after the trimming, respectively.

3.3.2 Colouring Point Clouds

Another step that is advantageous to do before the main processing, is to colour the point cloud. This can be done either in the form of colour per-polygon/vertex basis, or through textures covering larger areas.

Colours are usually supplied through a digital camera synchronized with the laser scanner. The process of coordinating the data manually can be both error-prone and very time-consuming, so an automatic merging of the data is desirable. An example of the differences between the point clouds when colours are applied can be seen in Figure 3.1 and Figure 3.2.

Another challenge is balancing the colour, contrast and brightness of the images. Because both the intensity and direction of the light change during the course of a day, the camera settings have to change to match the different conditions. This will inevitably lead to colour differences on overlapping parts during the merging of point clouds [26]. To compensate for this, all images should be adjusted to match the properties of a preset standard [39]. An example is shown in Figure 3.3, where a point cloud is compared before and after performing the colour correction.

3.3.3 Alignment

Alignment of the points clouds means creating a coordinate system common for all data sets. It can theoretically be done at any phase of the process, but by doing it early it is possible to discard redundant merging information and reduce the complexity of data storage.

The alignment can be based on pre-defined common tie-points between the scans, GPS navigation during scanning, or a combination of manual and

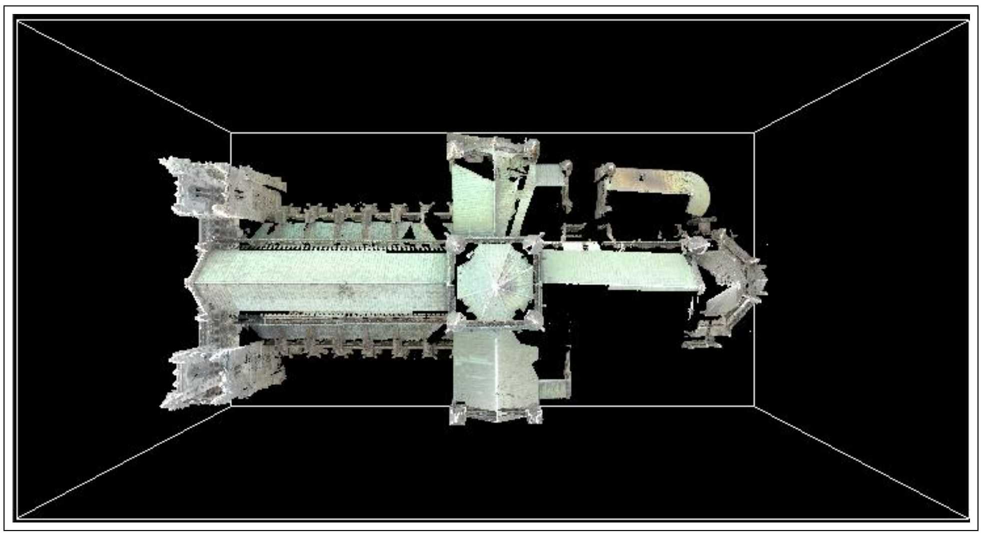

Figure 3.2: *The same point cloud as shown in Figure 3.1, trimmed and coloured.*

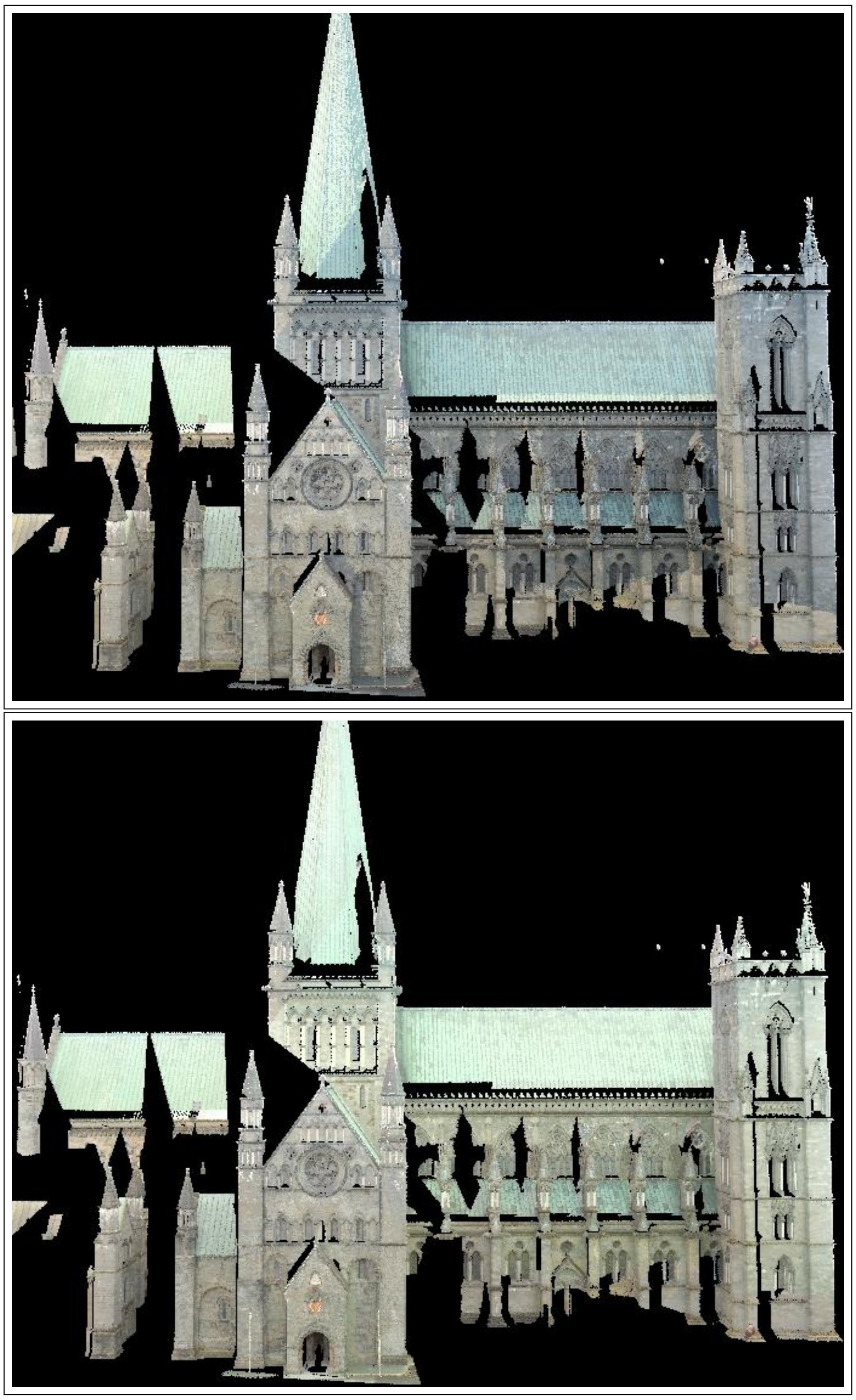

Figure 3.3: *The difference between raw colours (top) and balanced colours (bottom) on a point cloud.*

automatic merging done on the models after the data gathering [26].

This project used a set of pre-defined tie-points present on all point clouds, because in this case the laser measurements of the points gave more accurate results than any of the other methods could have [26].

3.3.4 Export

Exporting the point cloud indicates choosing which properties to retain (colour, surface normals, etc) and which storage formats to choose. The choices done during the first export of the raw point cloud will have repercussions for all future processing. This is partially because the different storage formats can lose information when translating between them, so the decision should be done with the intent to change formats as few times as possible (see Chapter 2.5).

The decision concerning the export of models is relevant every time a new tool is importing data. This is because the data is never exported in the same way, despite the work done regarding establishing standard storage formats.

In this case, both a textured and a non-textured version were exported to the VRML format, because it could be customized to store all relevant data (see Chapter 2.5.1). In addition, Geomagic Studio and Rational Reducer could both read VRML files without further conversions, reducing possible complexities.

3.4 Noise Reduction

Noise created by inaccuracies and scanning errors can both disrupt the visual quality of a model, and create problems and irregularities for other operations. Another (theoretical) result of reducing noise is making the edges sharper and the curvatures smoother [17].

There are several noise reduction techniques made for this purpose, most of which are already implemented in different processing and reverse engineering tools [27] [17].

One of the methods filter out points that vary too much from a plane created by its neighbouring points, which is implemented in Geomagic Studio.

Figure 3.4 shows the results of the outlier removal algorithm implemented in Geomagic Studio. The red dots are filtered out as noise, and constitutes about 0,05% of the total data amount. As can be seen on the image, there are more errors the further away the object is.

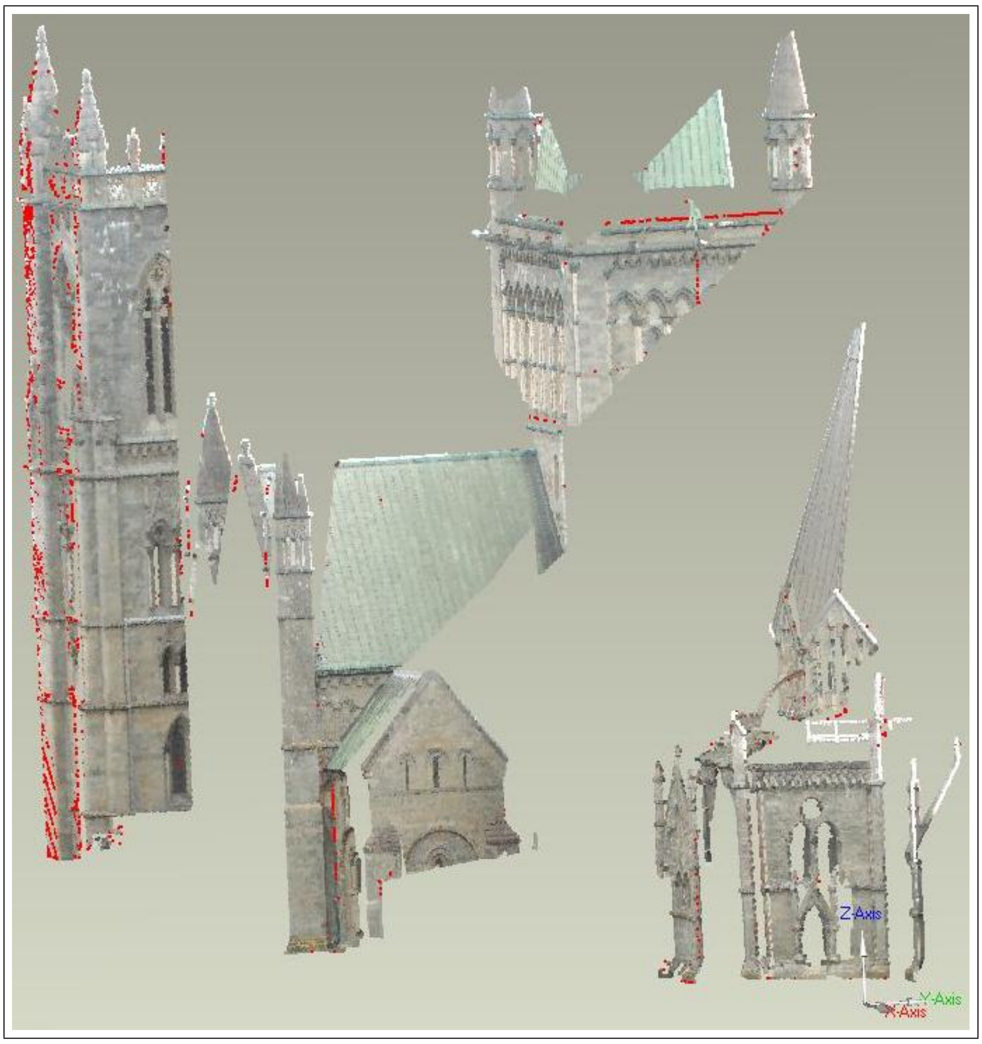

Figure 3.4: *Geomagic Studio's plane based noise reduction algorithm in progress.*

Another method moves the points around based on deviations, which are shown in Figure 3.5. By favouring prismatic shapes, sharp features such as edges on mechanical or angular shapes are better maintained. This method was used very conservatively, because sculptures and statues lost too many details. Figure 3.6 shows a comparison of the western wall before and after applying this method.

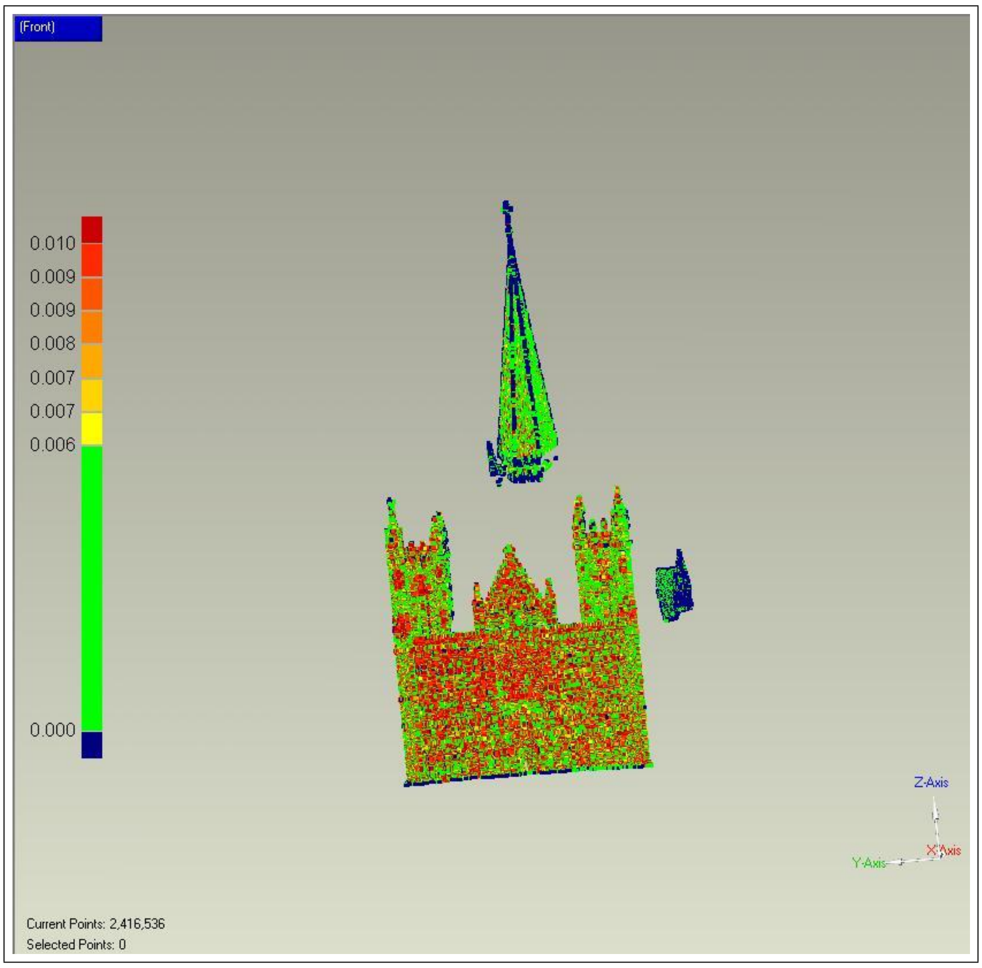

Figure 3.5: *Deviation chart from Geomagic Studio. The different colours represent grades of deviation.*

3.5 Triangulation

Triangulation is the creation of a network of surface polygons from point clouds. There are many ways of doing this operation, but based on research done in Chapter 2, Delaunay Triangulation (Chapter 2.3.1) was chosen for

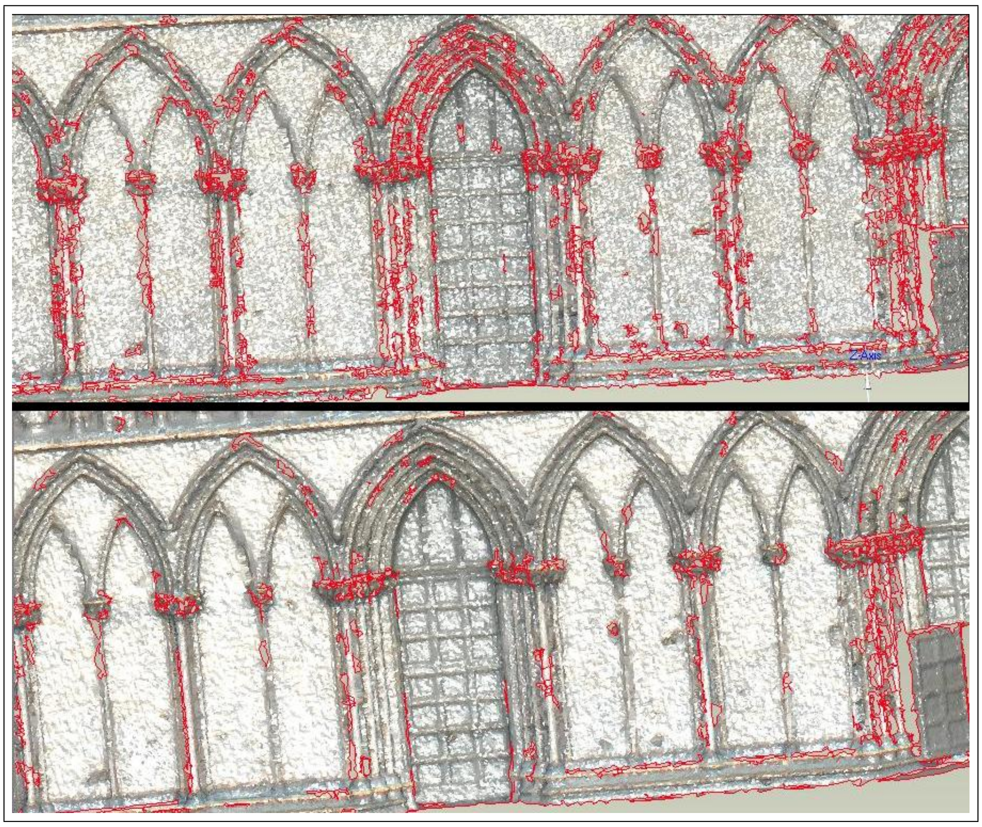

Figure 3.6: *Reduction based on prismatic shapes. The top image is before the algorithm is applied, the bottom image is after.*

this project.

Delaunay Triangulation is implemented in a wide variety of tools, but often returns different results based on the implementation and parameters. To reflect this, triangulations were done using three different tools, and the results compared afterwards.

RiSCAN PRO RiSCAN PRO (Chapter 2.4.1) supports Delaunay triangulation on a per-scan basis. In other words, each point cloud can be triangulated independently of the others, but several data sets can not be triangulated simultaneously. This created limitations for the triangulation where areas are overlapping, but the hardware requirements were lower since the operation was limited by the size of a single point cloud.

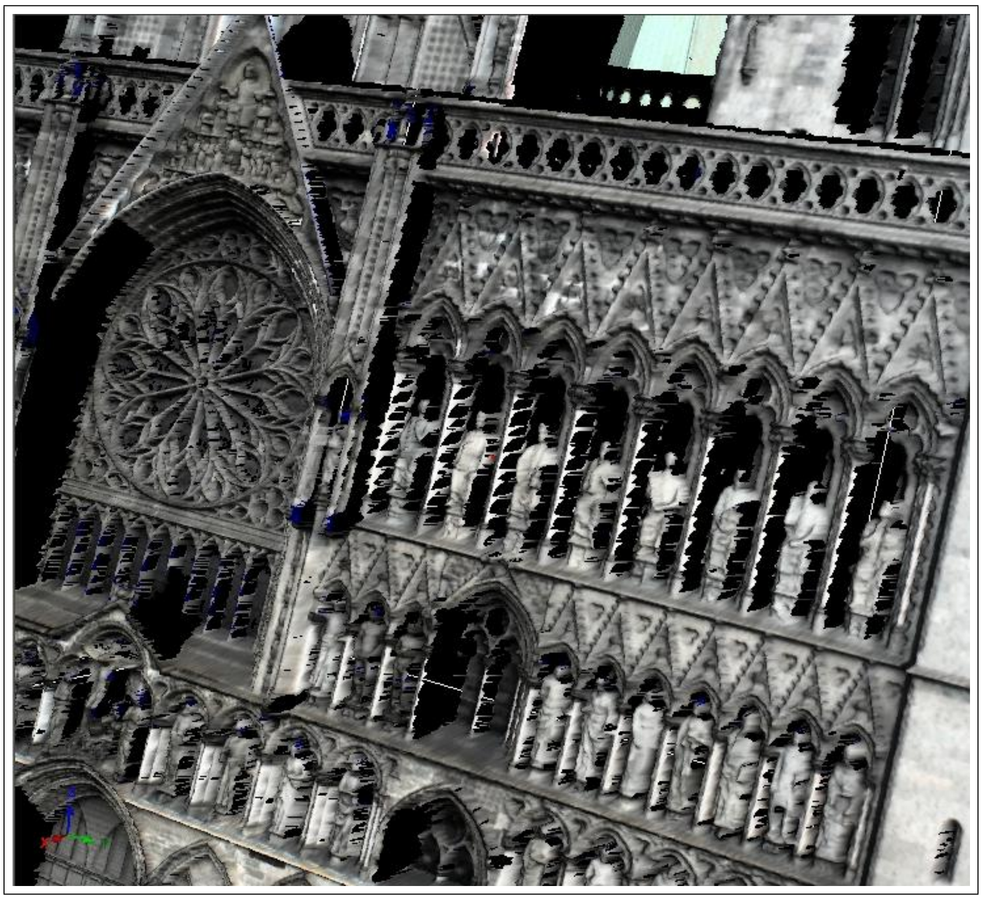

Figure 3.7: *Point cloud triangulated by RiSCAN PRO.*

Figure 3.7 shows the result of a Delaunay triangulation done through RiSCAN PRO.

Geomagic Geomagic Studio (Chapter 2.4.4) can triangulate point clouds regardless of whether it is a single point cloud or a combination of several point clouds.

The possibility of simultaneous triangulation of point clouds makes the operation much more flexible, and overlapping areas became a much smaller concern. On the other hand, the hardware requirements increases proportional to the size of the combined point clouds, creating limitations on how much data can be processed at once.

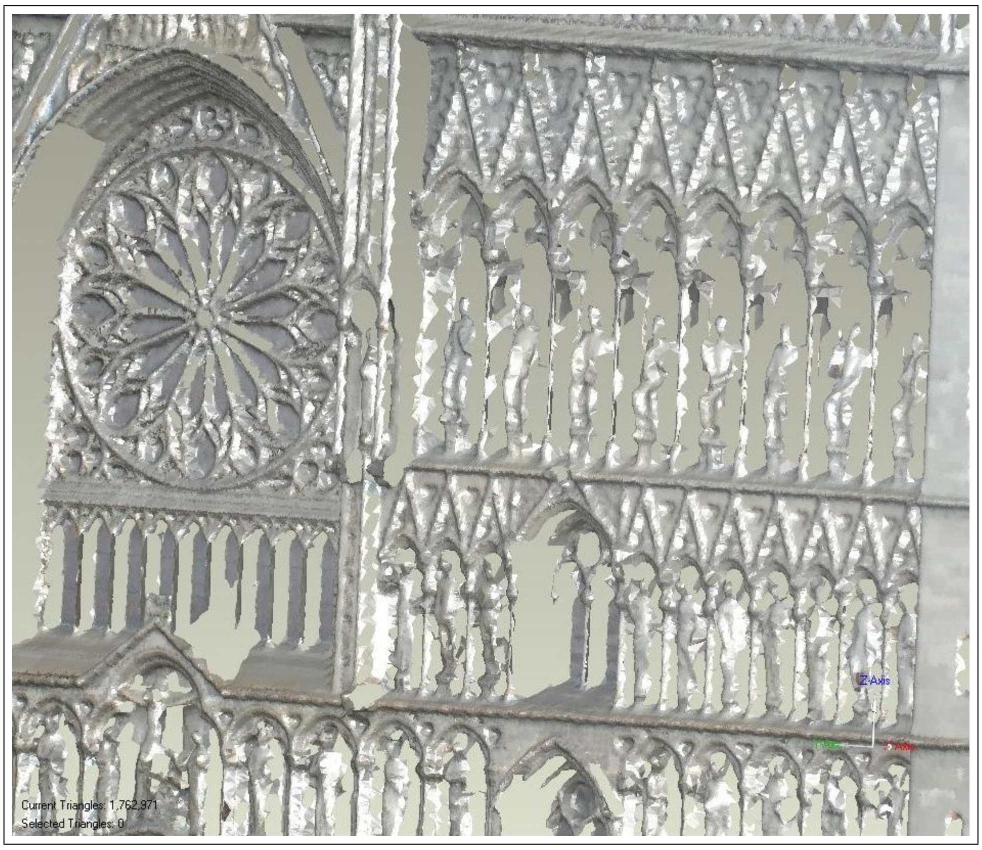

Figure 3.8: *Point cloud triangulated by Geomagic Studio.*

Figure 3.8 illustrates a Delaunay triangulation of a point cloud with Geomagic Studio.

CGAL CGAL (Chapter 2.4.2) has a highly customizable implementation of Delaunay Triangulation [14], allowing it to be adapted to the project's needs.

An adjusted version of the standard template was used to test and compare the different results the triangulation returned.

An example of Delaunay triangulation done with the use of CGAL is shown in Figure 3.9

Comparison The triangulations produced quite different results, and to avoid complexities during later processing, only one method should be used.

A comparison of the triangulations from RiSCAN (Figure 3.7) and Geomagic (Figure 3.7) shows that the former produced abnormally long triangles from certain angles. This was caused by assumptions done by RiSCAN regarding the geometric composition of the point cloud.

The triangulation done with the help of CGAL (Figure 3.9) could theoretically obtain better quality than both the one from RiSCAN (Figure 3.7) and Geomagic (Figure 3.7), but achieving this would consume too much time with implementations.

The decision fell on using Geomagic Studio to perform the triangulation for the reasons mentioned above. Furthermore, the choice was made to triangulate each point cloud separately in most cases. This allowed far more control over the different parts during later stages of processing.

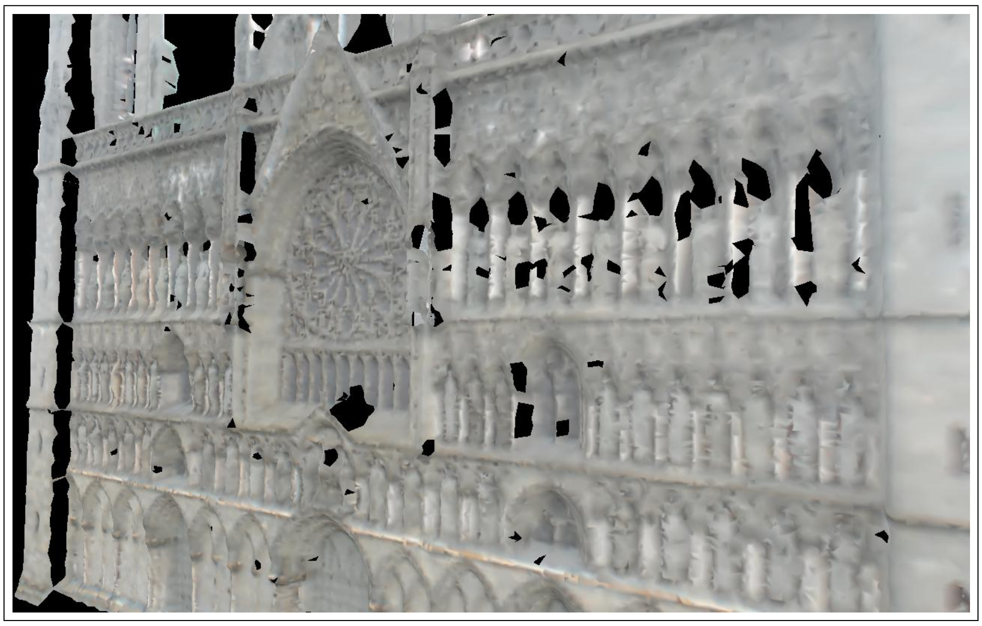

Figure 3.9: *Point cloud triangulated with the use of CGAL.*

3.6 Polygon Reduction

Polygon Reduction is an operation that reduces the number of polygons while *keeping as much of the visual quality as possible.*

There are several ways of doing this operation [27], but the Edge Collapse algorithm was chosen for this project (see Chapter 2.3.3 for details). Rational Reducer (Chapter 2.4.3) was used due to its implementation of Edge Collapse, but experiments showed that it needed certain adjustments to reach the best visual results:

- *Edge Length* was modified to ignore the relative size differences of the resulting new edges and surfaces. This resulted in large surfaces being created where few details were necessary (non-decorated walls), while still keeping the details of more suitable segments (statues, wall decorations, etc).
- *Curvature* was turned off completely due to a loss of details near sharp edges. Although the models looked smoother when this parameter was increased, future merging of the models would become more complicated.
- *Sharpness* was adjusted to maintain smaller details better.
- *Material* differences were ignored, because the lack of defined materials created unwanted results.

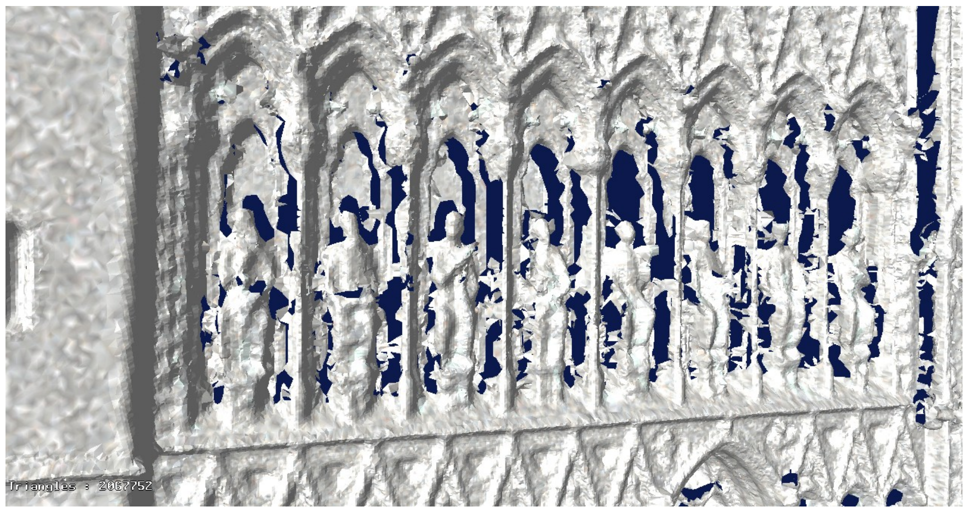

Figure 3.10: *The original (unreduced) data as shown in Rational Reducer.*

Figure 3.10 and 3.11 shows a comparison of a model before and after its reduction. No colours or textured are applied here, to better show the differences.

3.7 Crossover Removal

Intersecting triangles are generated during both the triangulation and merging operations. These triangles create complications for both processing operations and export to some formats (Chapter 2.5).

Figure 3.12 shows an example of intersecting vertices, where the intersecting polygons are outlined in green and the conflicting areas marked as orange.

One way of resolving this problems is to use an algorithm that automatically moves the offending vertices and thereby creates a flat mesh. The disadvantage of this solution is that some tangles are too complex and time-consuming to solve automatically. This project used Geomagic Studio's crossover removal algorithms to fix some of the areas.

Another solution is to manually remove the vertices in the offending area, and use a hole filling algorithm to repair the area by reconstructing the missing segments. This was done for all conflicts that were too difficult to do automatically.

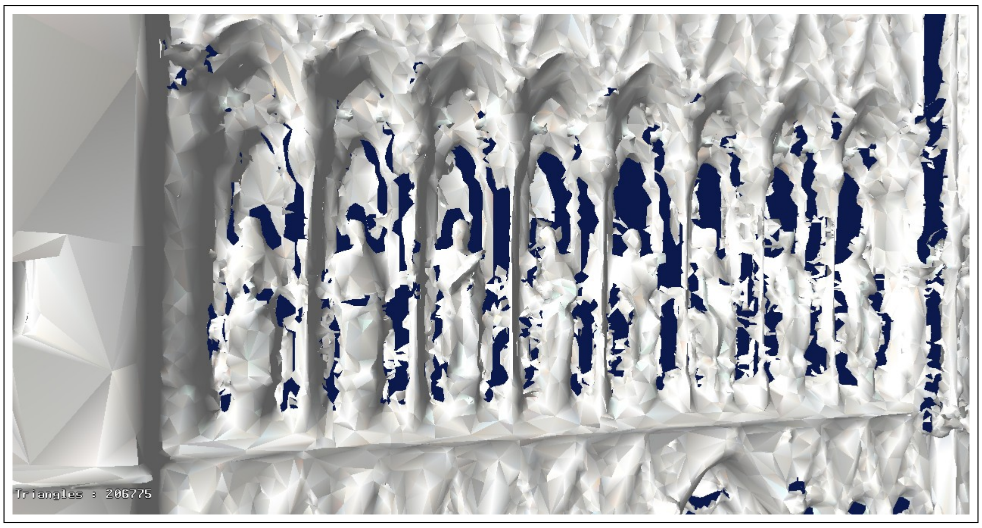

Figure 3.11: *A 90% reduction of the data shown in Figure 3.10.*

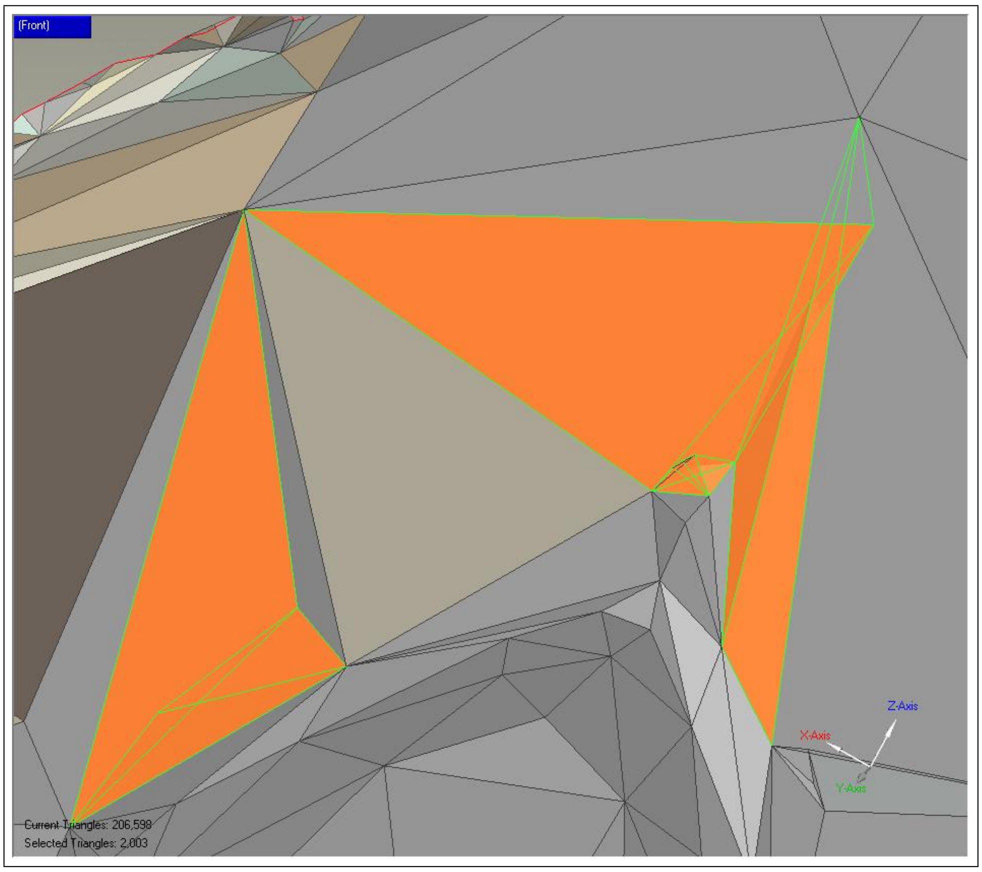

Figure 3.12: *Intersecting vertices tangled together.*

3.8 Hole Filling

It is theoretically possible to gather data from enough positions to cover all surfaces, but it is not practically feasible on large objects due to the time constraints of most projects [26]. Therefore there will always be holes in surface models based on scanning.

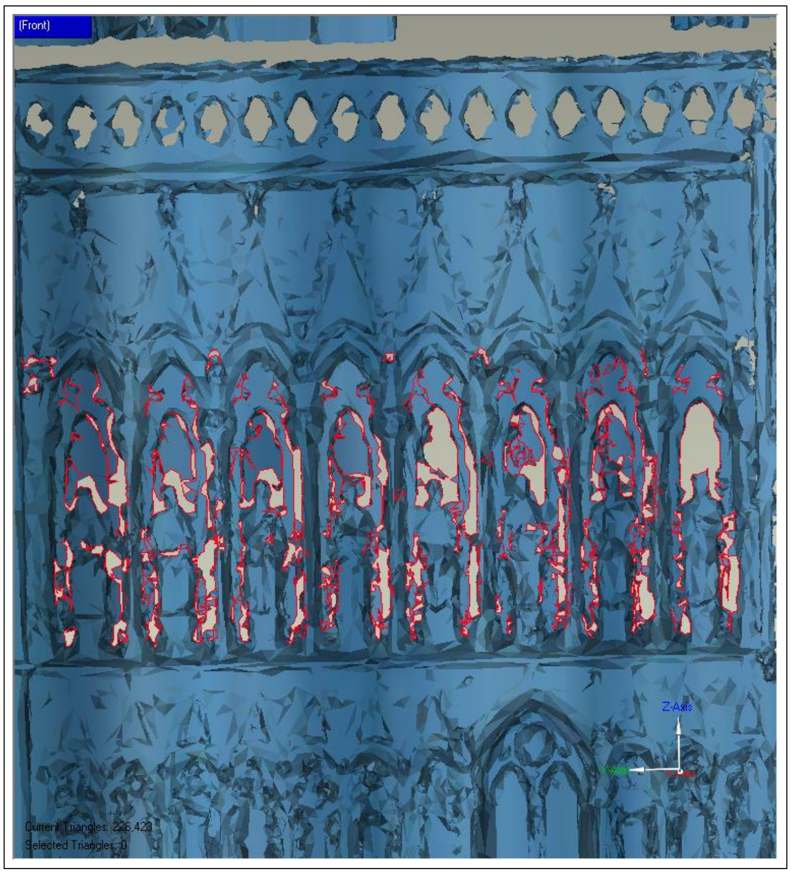

Figure 3.13: *Surface model of sculptures with holes.*

An example of holes in the model is shown in Figure 3.13, where the walls have several gaps, especially near the sculptures.

There are multiple ways of filling such holes, both automatic and manual. The automatic methods are much faster than the manual ones, but the visual quality of the result can vary a lot [17]. No matter which method is chosen to fill the holes, all gaps should be automatically identified by modern reverse-engineering tools [26].

The automatic filling algorithm chosen for this project tries to follow the curvature of the surroundings where possible. If the gaps are too wide, they will be replaced with a flat wall. In both cases it smooth out the area filled and the surrounding edges to create a continuous surface mesh [17].

Manual filling requires a lot more work regarding choosing which surfaces to extend, the curvature of both the filled part and surrounding area, colours, etc.

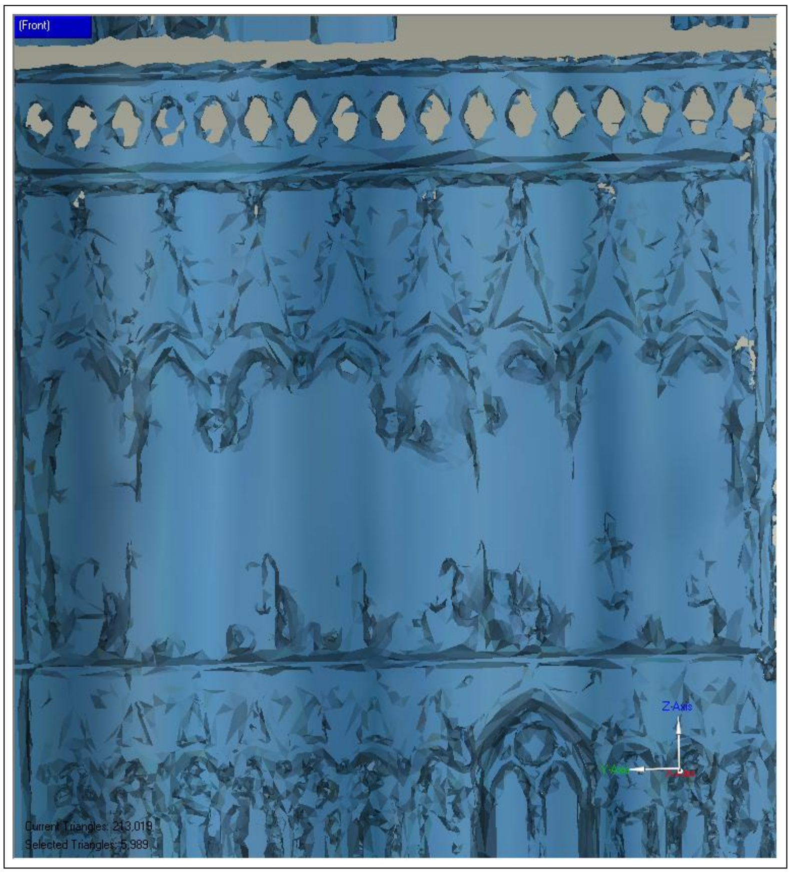

Figure 3.14: *Automatic hole filling of sculptures with Geomagic Studio. Compare to Figure 3.13 and Figure 3.15.*

Figure 3.14 and Figure 3.15 shows the different results achieved through automatic and manual hole filling, respectively. This example illustrates that for highly detailed areas with sharp features, the manual method is superior to the automatic. This severe limitation of the automatic algorithm is caused by a lack of data, something that is hard to replace without human knowledge of how the features should look [17].

An illustration of automatic hole filling working better can be seen in Figure 3.16. The left part of the image shows the original wall segment, while the right part has been filled by the automatic algorithm. On areas where

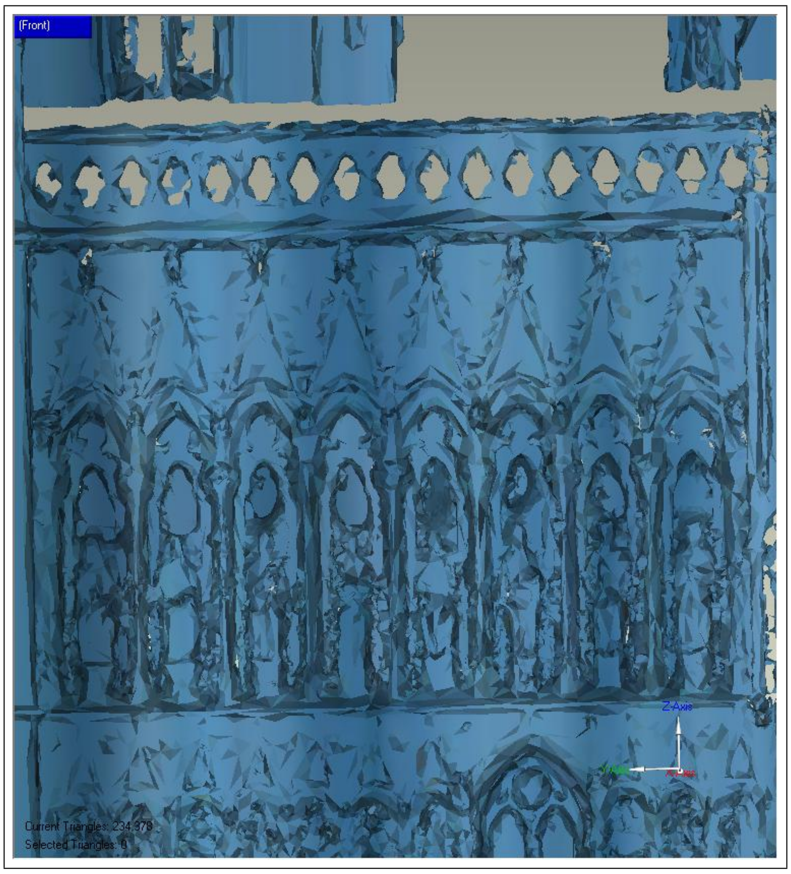

Figure 3.15: *Manual hole filling of sculptures. Compare to Figure 3.13 and Figure 3.14.*

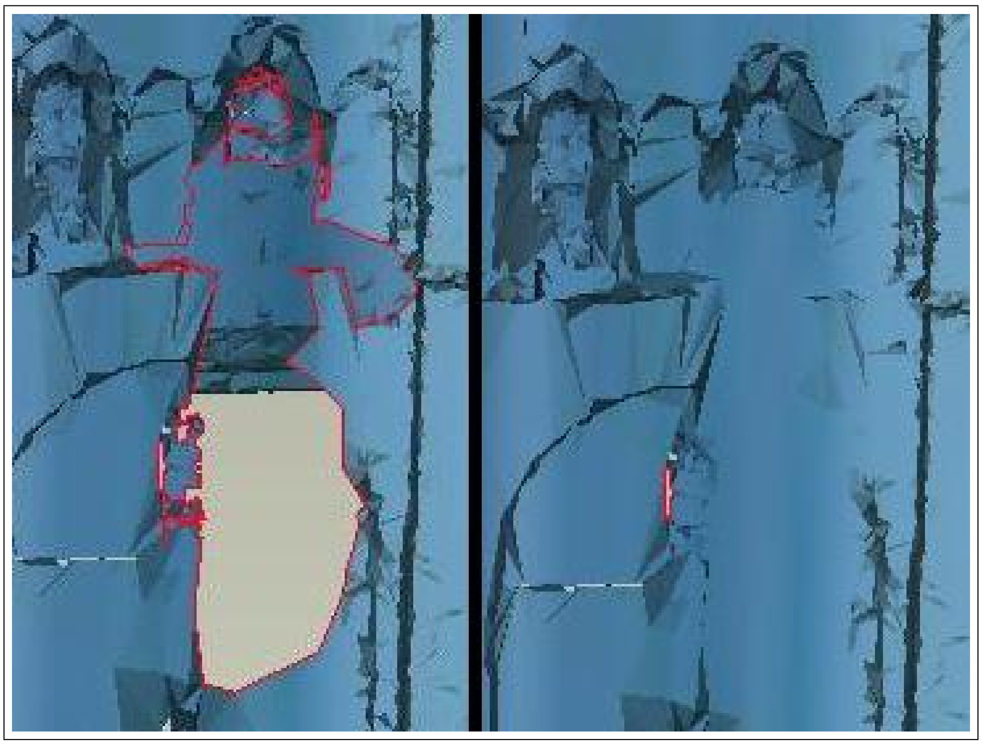

Figure 3.16: *Automatic filling of the hole on a wall.*

keeping small details is of less importance, the time saved by doing the hole filling automatically can be considerable [26].

3.9 Edge Straightening

After being through Noise Reduction, Triangulation and Hole Filling, the edges on the models are often deformed, which impairs the visual quality. Therefore the edges should be straightened after major operations that drastically change the composition of the model.

The simplest way of achieving this is by creating a straight line between the end points, and moving the nearby edge-points to the line.

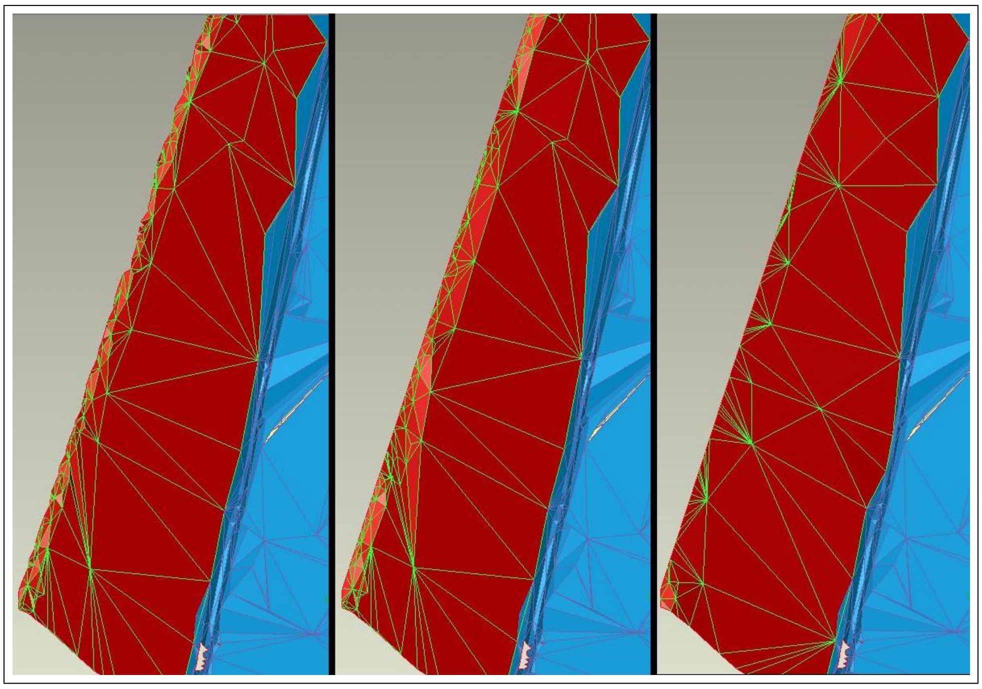

Figure 3.17: *Comparison of the original edge (left), a straightened edge (center), and a reduced version of the straightened edge (right).*

Figure 3.17 illustrates the effect gained by straightening the edges. The image in the middle demonstrates that the straightened edge is simpler than the curved lines on the left image. The right image shows that reducing the area afterwards actually improves the visual results.

3.10 Merging

The merging of data is combining input from two or more data sets to create a single output. An assumption is made beforehand that the data sets have a common coordinate system (see Chapter 3.3.3).

The merging process can be done with any sort of Boolean operation, including *Union*, *Combine*, *Fragment*, and *Subtract* [18]. It can be done using both point clouds and polygons, and during any phase of the reverse engineering process. The merging is often done either as early as possible during such projects, or as late as possible, depending on the data and the desired results.

The main reason for doing it early is that overlapping points can be removed through noise reduction. This is mostly done when the data sets have relatively large overlapping parts [26].

The main reason for doing it late is because limitations on hardware and software can create problems when there is too much data, and everything can slow down or fail to work entirely. This method is often chosen for data sets with few overlapping areas [26].

Figure 3.18 shows a comparison of merging before and after triangulating the point cloud. Problematic areas (intersecting polygons, see Chapter 3.7) are shown as red areas, whereas the blue areas mark sections without problems. In this case the point clouds overlapped very much, resulting in merging the data sets before triangulating produced the best result.

Figure 3.19 shows another comparison of merging before and after mesh creation. In this case, the data sets have very few overlapping areas. Red areas mark problematic sections with intersecting polygons, of which there are some (but few enough to handle) in the case of merging after triangulating. However, closer study of the left image reveals that the model becomes very rough and bumpy, which is not visually pleasing. Using a stronger noise reduction setting than the one used for the rest of the project could have solved this, but that would impair the visual quality in other ways. This demonstrates that for this structure, waiting with the merging produced better results.

The merging of this project was done using two different tools: Geomagic Studio and AccuTrans 3D.

Geomagic Studio was used during the merging of point clouds early in the process. Large amounts of data created problems when merging models; both long processing times and erroneous results. For this reason very, few data sets were merged early.

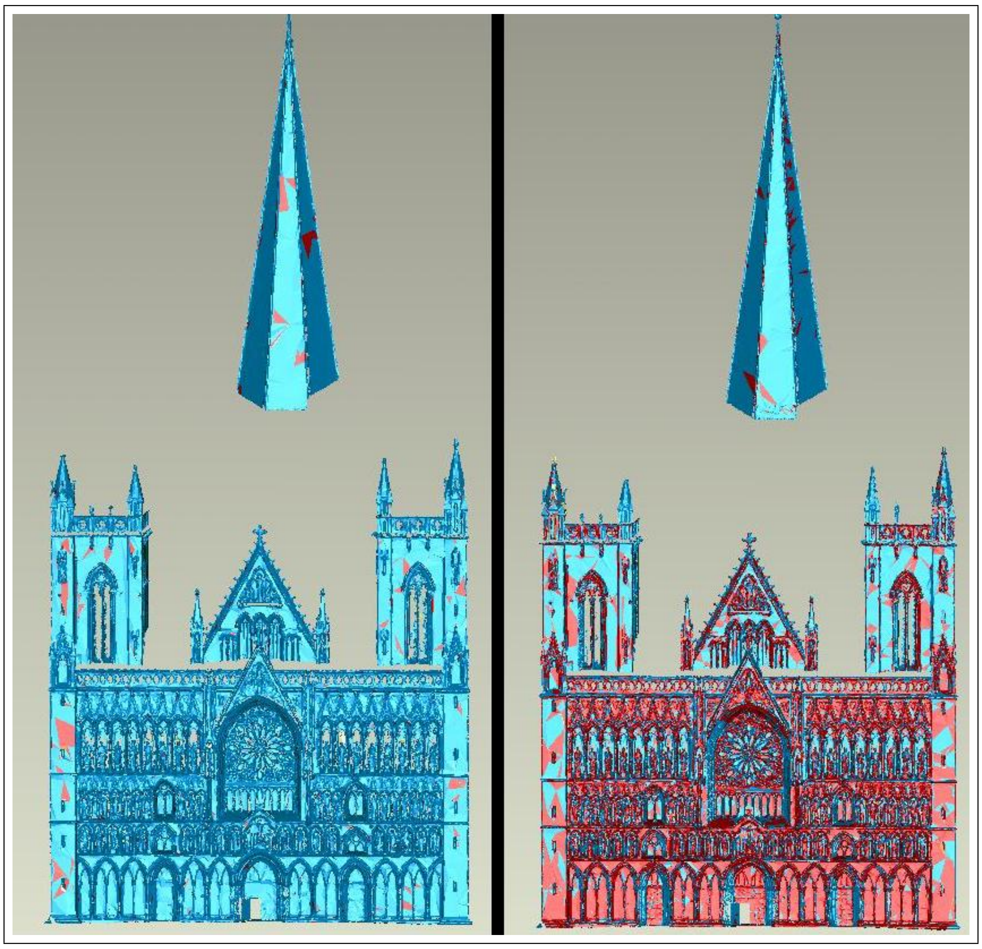

Figure 3.18: *Comparison of merging before triangulation (left) and after triangulation (right).*

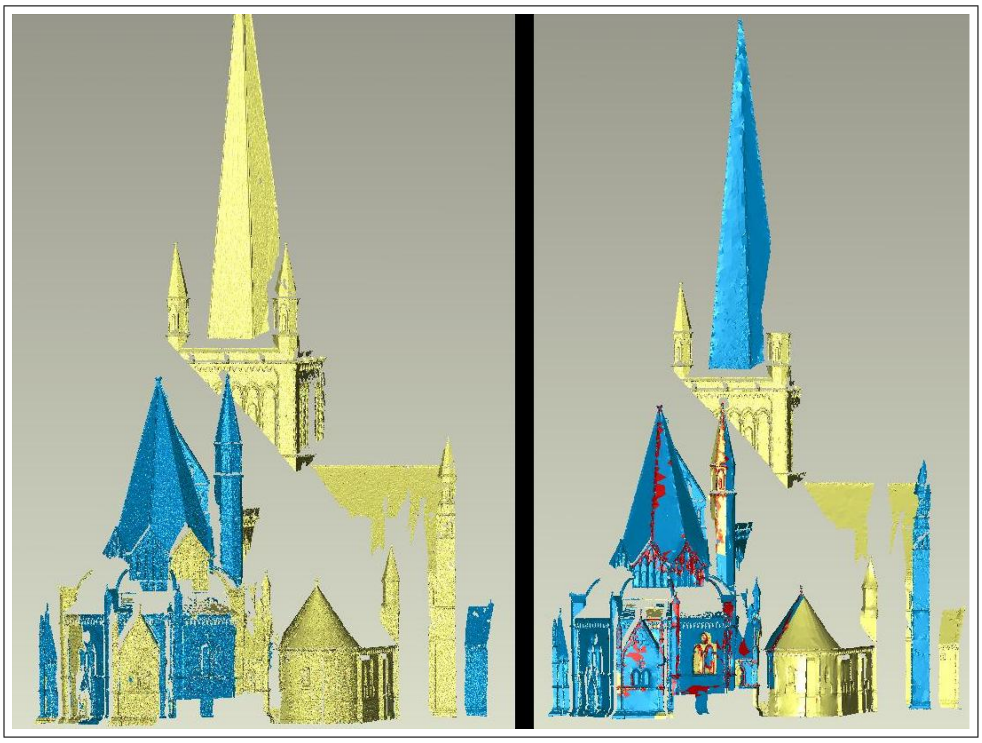

Figure 3.19: *Comparison of merging before triangulation (left) and after triangulation (right).*

AccuTrans 3D was responsible for merging the surface models at the later stages of the processing. This was because its light-weight implementation lessened the burden on the hardware, and more complex merging operations became possible.

3.11 Texturing

Colour information was gradually lost through transformations and format conversions. Polygon reduction, hole filling, decimation and other operations all took their toll on the finer details of the objects.

To counter the loss of visual quality following the deteriorations, textures can be added as an outer layer on the models. This technique is very common in video games, where highly detailed textures can make poorly modeled objects look much better [49]. In this case, textures were created during the Data Preparation phase (Chapter 3.3), which were still calibrated to the models coordinates.

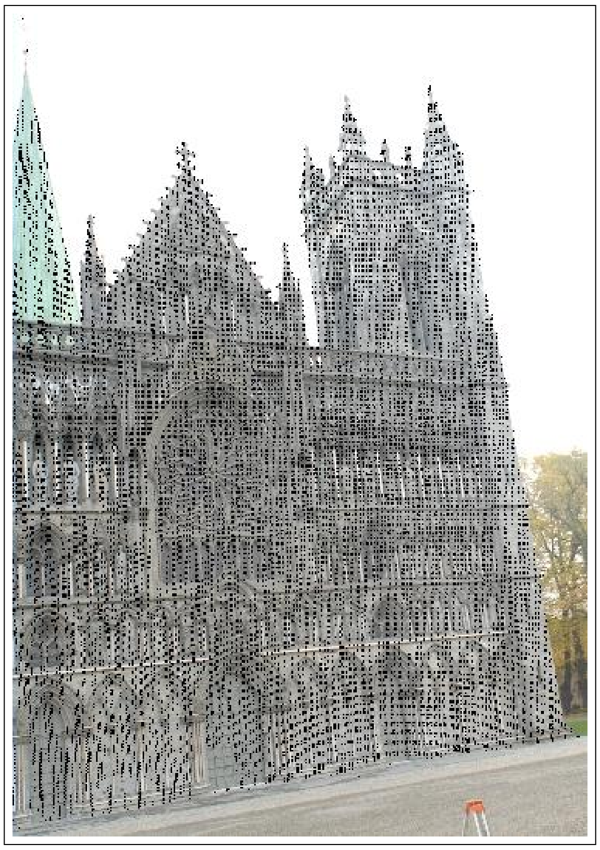

Figure 3.20: *Textures on top a point cloud.*

Figure 3.20 shows textures on top of a point cloud. Although the objects are triangulated, reduced, and modified in other ways, using the original textures is an easy and automated operation in most reverse engineering tools [17].

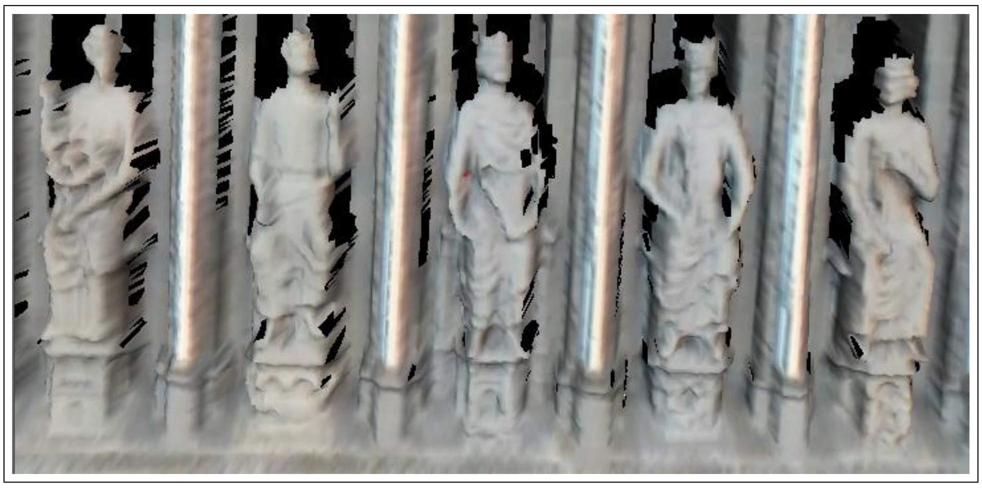

Figure 3.21: *Textures on top of a surface mesh.*

Figure 3.21 demonstrates how textures can be applied on a triangulated surface mesh. Although the sculptures in this image have lost a great deal of detail, the textures improves the visual quality and makes the real features more visible.

The main challenge with adding textures is that areas that are not covered by the original textures, such as filled holes and sections outside the camera range, will stand out from the rest of the model. This is discussed in greater detail in Chapter 6.1.2.

3.12 Rendering Stereoscopic Videos

A stereoscopic video creates an illusion of depth by having two videos with a slightly different viewpoint. If calibrated correctly, giving the eyes different views forces the visual centre of the brain to interpret the objects in three-dimensional space [1].

To get the best effect of depth, the focal point was set to the centre of the model. This results in positive parallax for objects further away and negative parallax for the closer objects, adding to the depth immersion [26] [25].

There are several ways of configurating the cameras, depending on the desired effects [1]. For this project, a toed-in camera configuration was chosen. In other words, both cameras are pointing towards the focal point. The distance between the cameras should emulate the distance between the pupils (65mm), so both the viewing distance and the screen width should be decided before rendering the videos [25] [55]. A camera distance of 1/30 of the focal distance proved to be a safe base for producing pleasant results [9]. Increasing this distance creates a higher feeling of immersion in 3D, until it gets unpleasant.

Several limitations can occur when rendering a stereoscopic video. The most challenging one was when some parts of the objects came 'closer' to the viewer than half the focal length. This puts extra strain on the eyes and creates unpleasantness when focusing on the objects. The easiest workaround is using bigger lenses for close-up shots, and thereby avoiding the effect.

For this project, 3D Studio Max (Chapter 2.4.5) was used to render one video for each eye. These video files were further compressed to the WMV format by Windows Media Encoder (see Chapter 2 for details). To merge them into one stereoscopic video file, Windows Media Stereo Muxer was utilized [1].

3.13 Perspective Correction

Tilting the camera during rendering of large objects causes vertical lines to converge with the height. This is caused by the tilting, since the lens is no longer parallel to the viewing-plane. Although the same thing happens in real life, the accepted norm of constructed perspective demands edges that are straight in reality to be straight in images and videos too [46]. There are two ways to correct this; either by avoiding the problem, or by adjusting the perspective [47].

The easiest way of avoiding the problem is for the camera to always be parallel to the viewing-plane. The disadvantage of this solution is that the camera either has to be far away to get the whole object in view, or only get parts of the object.

Adjusting the perspective is the 'correct' way of compensating for the perspective distortion. By moving the frustum upwards, the camera can stay parallel to the viewing-plane and still get the whole object. This technique is used in special perspective correction cameras that can move the lens, achieving an affect similar to moving the frustum [47]. Figure 3.22 illustrates that perspective correction (bottom) provides the best results.

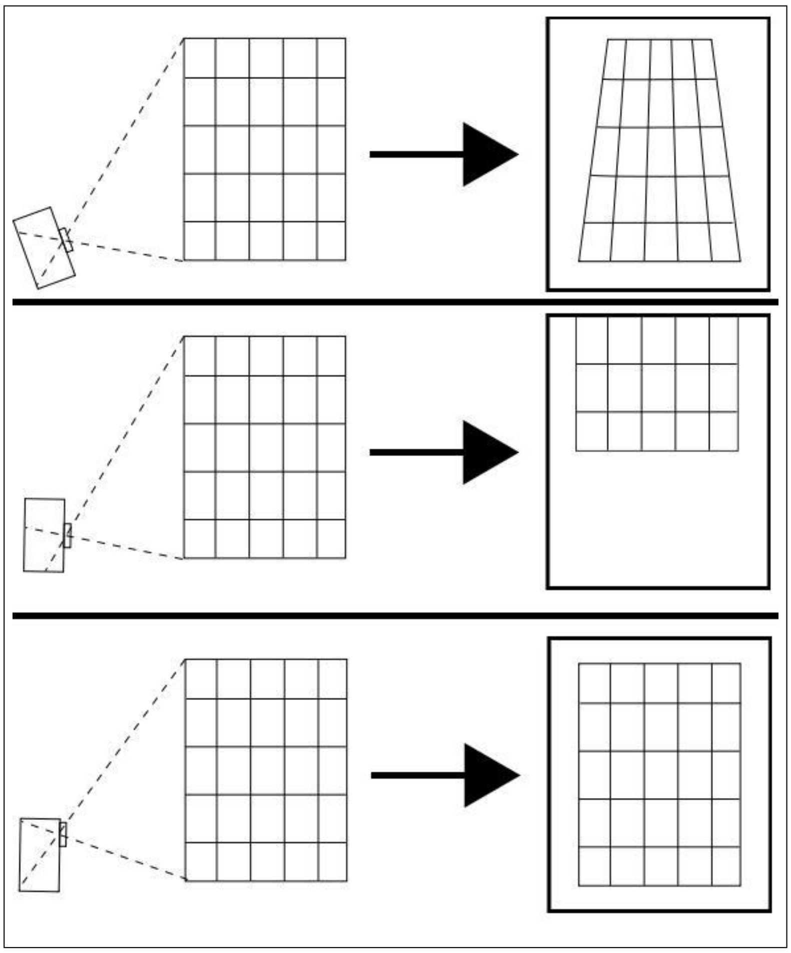

Figure 3.22: *Tilted camera (top), parallel camera (centre), and perspective correction (bottom).*

Figure 3.23 demonstrates the differences with and without perspective correction from the rendering of this project's movies.

3.14 Summary

All the operations described in this chapter show a way to overcome different challenges concerning reverse engineering. This is a pilot project, so some of the steps are detailed with several alternatives, showing the different results achieved through experimentation. For other phases only one path is chosen, based on both research and limitations imposed by software and hardware.

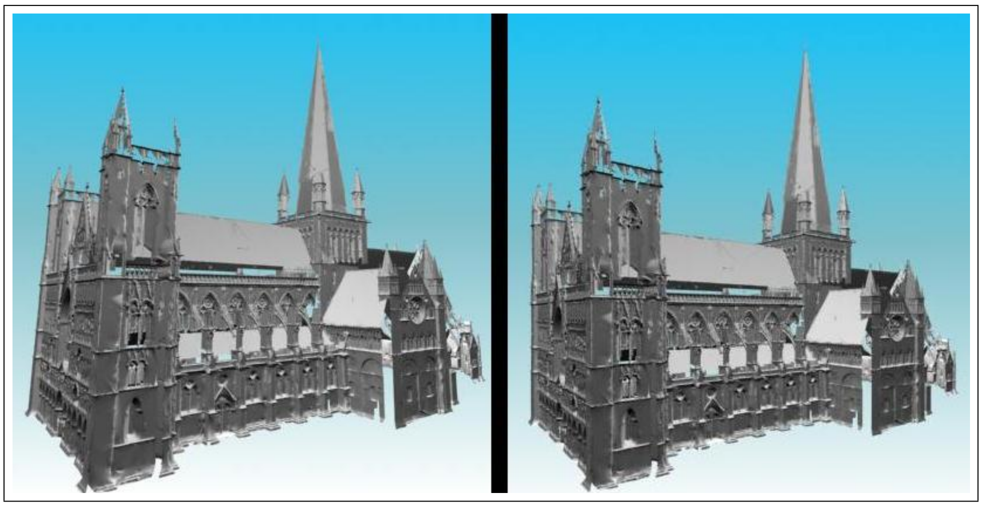

Figure 3.23: *Uncorrected perspective (left) compared to the same model with perspective correction (right).*

Chapter 4

Results

As a pilot project, the focus has been on investigating the different steps needed to process the data automatically, with the goal of creating a visually pleasing model in 3D. The model was rendered as a stereoscopic video to demonstrate it from all angles.

4.1 Model

The main result is a model of the Nidaros Cathedral, which have been processed through all the steps detailed in Chapter 3. The textures were not added to the model for several reasons, which are detailed in Chapter 6.

Although the scanner coverage was 360° around the cathedral at ground level, some parts of it are missing due to obstacles. These holes were too large to be filled during the data processing, and can still be seen on the model from certain angles.

The final model consists of about 2 million polygons (138MB), which is a huge reduction from the 29 million polygons (2080MB) exported after the Data Preparation phase (Chapter 3.3).

The model is accessible through the DVD accompanying this report. It is available in both the VRML and the PLY format, but can be translated to other formats through the tools used in Chapter 2.4. Even though the model lacks real colours, the original textures and colour information are still available.

Figures 4.1, 4.2, 4.3, 4.4, and 4.5 shows the model from five different angles.

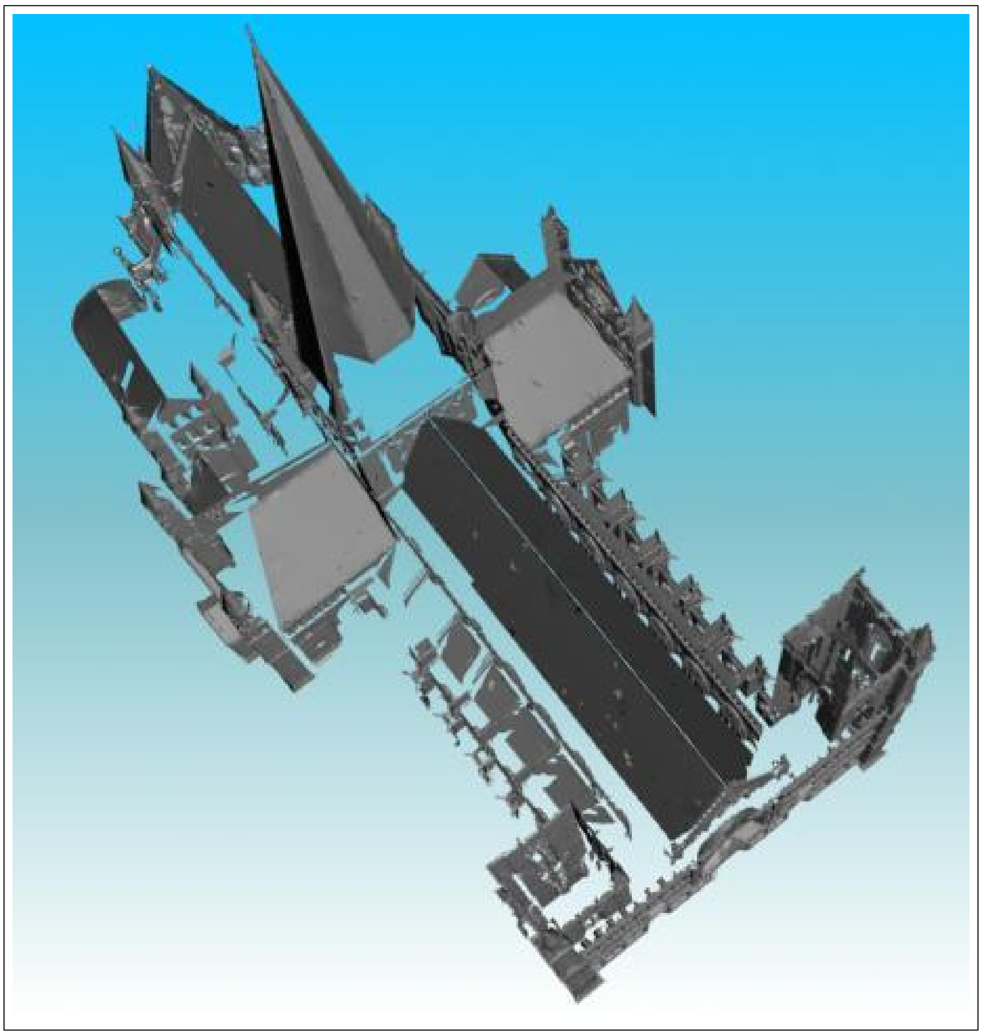

Figure 4.1: *The Nidaros Cathedral as seen from the air.*

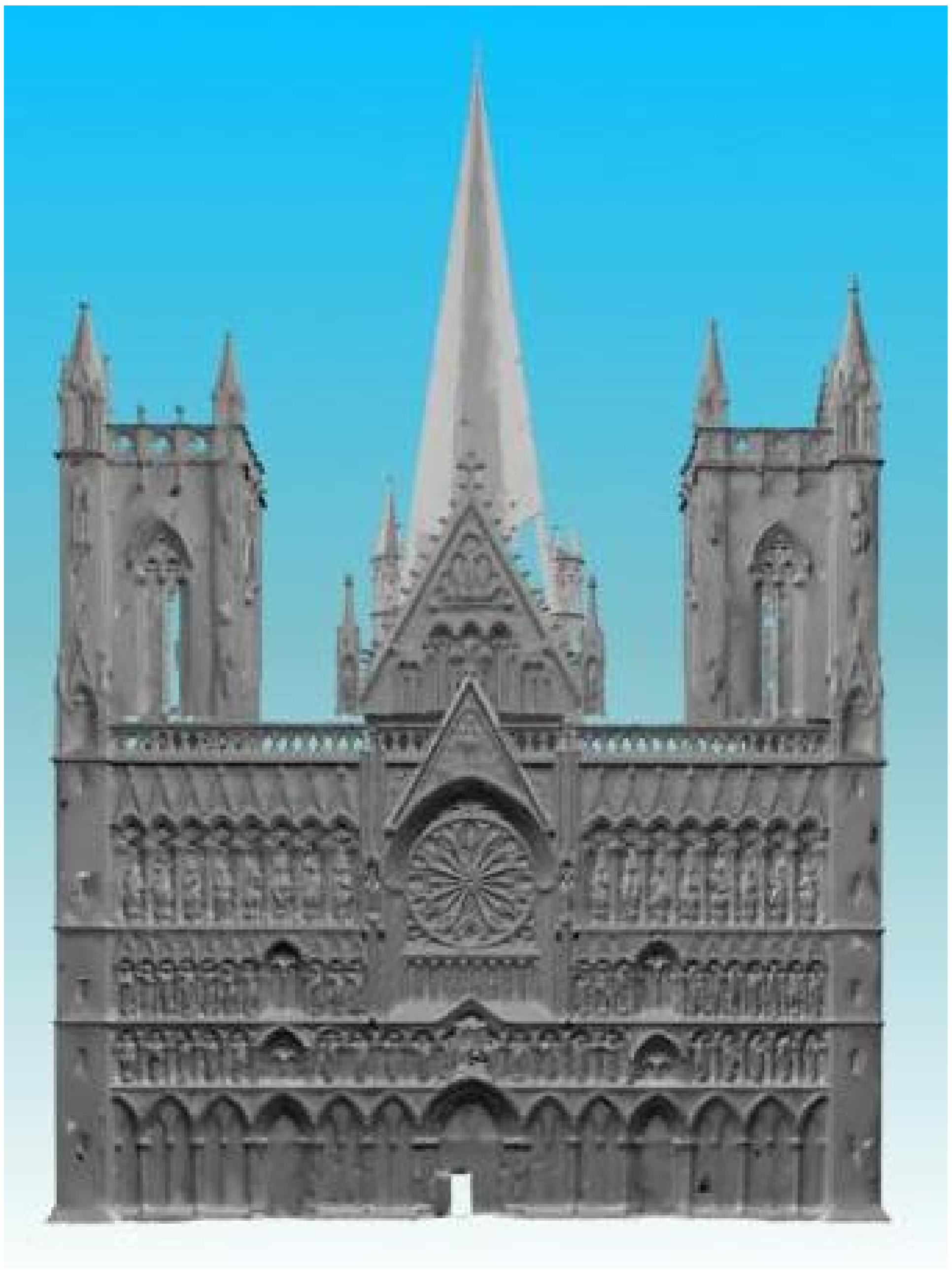

Figure 4.2: *The west front of the Nidaros Cathedral.*

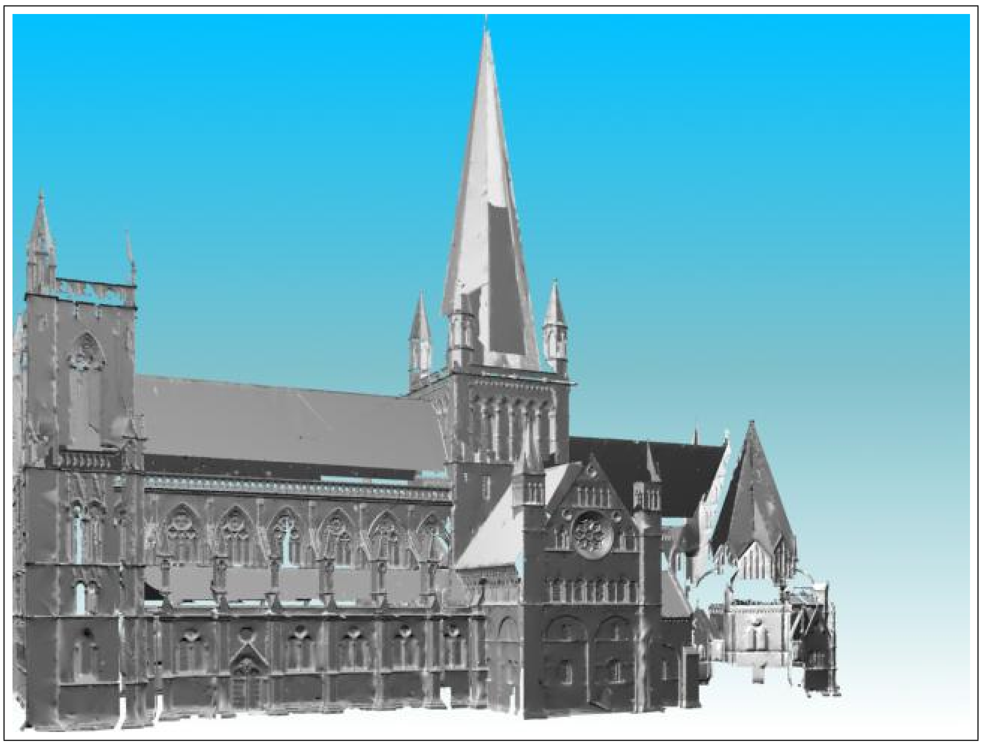

Figure 4.3: *The cathedral seen from the south.*

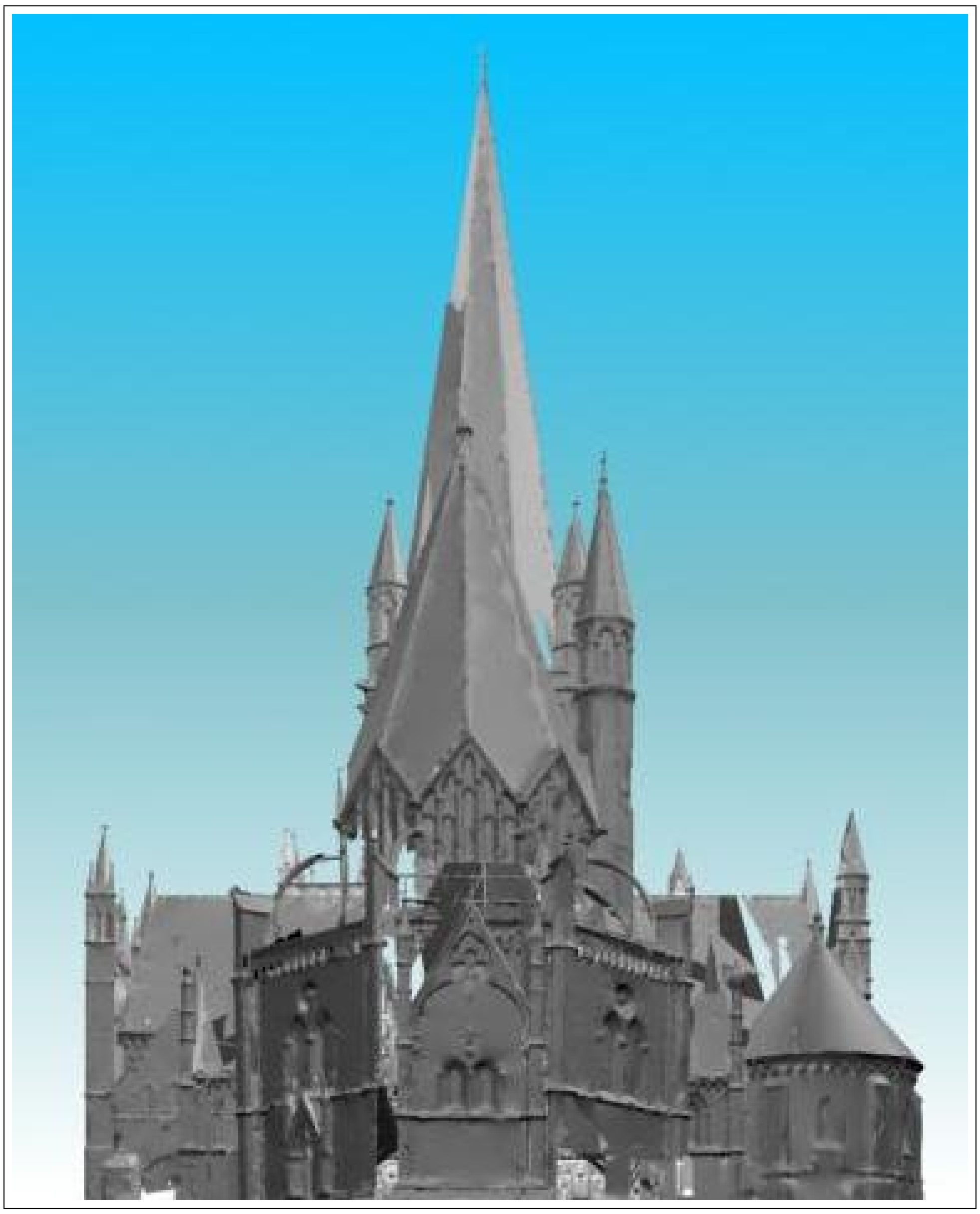

Figure 4.4: *A screen-shot of the eastern side.*

4.2 Stereoscopic Video

A stereoscopic video was rendered to demonstrate both the model and the possibilities of the available technology. The video consists of two parts; first a tour around the cathedral with a relatively long distance to the model, then a close-up of the western front.

Each minute of rendered video consisted of approximately 2GB of data (uncompressed) for each view. The compression in WMV reduced the size to less than 3% of this.

The final video configuration is:

Duration	2 minutes 25 seconds
Size	161 MB
Frames Per Second	30
Dimensions	640x480

Figures 4.1, 4.2, 4.3, 4.4, and 4.5 are screen-shots from the video, showing some of the angles featured during the tour around the cathedral. The video is available on the DVD accompanying this report.

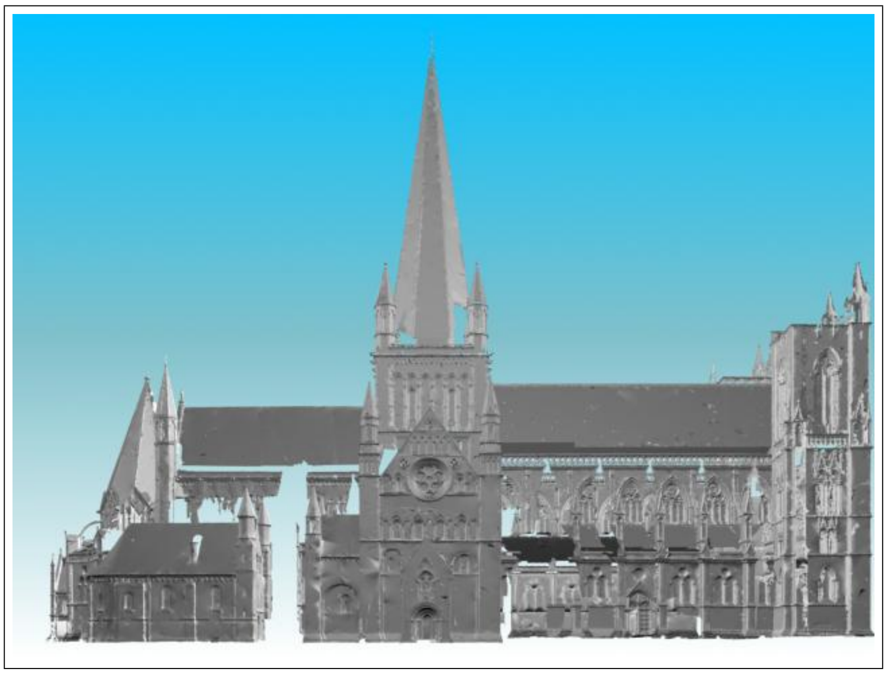

Figure 4.5: *The northern part of the cathedral.*

Chapter 5

Conclusion

The pilot project has completed most of the goals regarding both the model processing, video rendering, and gathering of useful experiences for other projects. Five objectives were established in Chapter 1.2, and they have all been more or less achieved.

Removal of noise and inaccurate data. Both noise in the model and inaccurate scanning data have been carefully cleaned up or compensated for.

Merging the data sets. All data sets have been merged into one complete model.

Creation of 'watertight' surfaces. The whole model is not watertight, but the different segments have accomplished this goal partially.

Making the model 'light-weight'. The final model is small enough to be used on normal computers without creating memory problems or slowing down.

Demonstrations of the cathedral. A stereoscopic video was rendered, showing the cathedral from all angles, including a close-up of the western front.

Chapter 6

Discussion and Further Work

6.1 Discussion

During both the research, experimentation and processing phases, several issues surfaced that seemed to be of importance for similar projects.

6.1.1 File Formats

One of the greatest and most problematic issues is the different file formats.

Too many file formats are claimed to be the standard for the different fields in computer modeling, and most of them are backed by powerful actors in the industry [17] [35] [24] [43]. Despite several attempts to create a unifying, common file format that satisfies most modeling needs, many of these big companies refuse to convert and insists on using their own format [8] [6] [15] [12]. This results in the various software tools supporting different formats.

The problem arises when models are to be transferred between these applications. Far from always can the same format be found among the respective export and import possibilities. The consequence of this is that third-party tools are needed to convert a model from one format to another [34], creating a bottleneck for the processing. This translation can cause problems when the formats do not have the same properties regarding what qualities can be stored [26]. Another problem is that some formats have long loading times due to inefficient structuring. 3D Studio Max was able to load model data from the PLY format, but was unable to read the exact same model when stored in the VRML format.

Experience shows that there is no easy way to solve this problem, because earlier attempts at creating a standard format that satisfies all needs, failed to get the necessary support from the big actors [12] [13]. A temporary solution that some of the tools have implemented is the possibility to import and export a great variety of different formats. Until a common standard is embraced by all big companies, this seems to be the best option.

6.1.2 Textures

The photos the textures were based on were taken through a digital camera with a 20mm lens. The distance to the closest wall was in most cases 20-50 meters, indicating textures given on surfaces 100+ meters away from the camera.

The main goal of the textures was to improve the perceived amount of details on the sculptures. Due to limited time for data gathering, it was not practically feasible to get more detailed photos. The consequence is that the textures did not improve the visual quality as much as anticipated. Since the hardware limits were already being pushed as far as possible, the decision was taken to not include the textures on the final model.

6.1.3 Stereoscopic Video

Using a stereoscopic video to show a 3D model proved to be useful, because the immersion and feeling of depth improved the visual quality of the model. But there are some problems with stereoscopic videos, and they become more prominent if the configuration is a little off.

Perspective correction (Chapter 3.13) was not used in the final rendering due to time limitations. Tweaking it to give satisfying results would have been possible with more time and a better choice of tools.

Another problem is the strain put on the eyes when focusing on parts of the object far from the middle. Due to the toed-in camera configuration, the vertical parallax increases further away from the centre, an effect known as *keystone distortion* [55].

A valuable experience is that to avoid unpleasantness, the camera focus has to be carefully attended when turning corners, so both views turn simultaneously.

6.1.4 Hardware and Software Limitations

Limitations placed on projects from available hardware or software can in some cases be crippling. Although the hardware and software problems are not directly related, they are of similar nature and can affect each other.

One of the challenges encountered during the scope of this project was the lack of powerful hardware. This limited the amount of data that could be processed simultaneously and caused the advanced operations to be very time-consuming. On the other hand, there will always be hardware limitations. More powerful hardware would result in a more detailed model, but still face the same challenges regarding pushing the limits.

A similar issue is constraints implemented in the available software. There were upper limits on the amount of data the software could load, not subject to the available hardware. Inefficient implementations of algorithms caused operations to take considerably more time than necessary when the amount of data exceeded what the developers had anticipated. New versions of software seem to fix many of these problems, but there are still many challenges and room for a lot of improvement.

The biggest bottlenecks in this project were Rational Reducer and 3D Studio Max, being limited by both hardware and software (see Appendix C). The workaround was to reduce the models to limit the scope of the problem. They were still kept as large and detailed as possible, since the loading and processing times were of little significance.

6.2 Further Work

Although this thesis shows how a model could be processed and demonstrated through practical examples, there still is a lot more work to be done. This future work regards both improving the current results, and to better plan similar projects.

6.2.1 Textures

New images with higher resolution could be acquired, making texturing the entire model more sensible. This would also mend visual irregularities created by hole filling.

The new textures have to be added either through synchronizing with existing tie points, or through a merging algorithm such as Iterative Closest

Point [7]. With the use of modern tools (Chapter 2.4), this can be achieved more or less automatically.

6.2.2 Offaxis projection

The stereoscopic videos can be improved by using a parallel camera configuration. Offaxis projection can be used to avoid convergence that appears when the distance reaches infinity (see Chapter 3.13) [55].

Moving the frustum of the cameras closer together will result in the cameras having a shorter convergence distance. This is similar to increasing the angle of the cameras in the toe-in configuration. The benefit of moving the frustum this way is that both the camera and the viewing plane are parallel to the scene [51]. The consequence is that the keystone distortion and the depth plane curvature are avoided [42].

6.2.3 Level of Detail

Level of Detail (LoD) refers to decreasing the complexity and amount of details on 3D objects as they move further away from the viewer. Other conditions can influence the complexity too, namely object importance, speed, and position [45] [11].

The reduction of details can greatly increase performance on systems where the capacities of hardware or software have reached their limits. Decreasing the amount of polygons will reduce the visual quality of the objects, but if implemented correctly this loss will only have a minor effect on the appearance of the model due to distance or speed (CLoD)[11]. There are two main approaches to this, namely *Discrete LoD* (DLoD) and *Continuous LoD* [45].

The first method (DLoD) creates a finite number of different detail levels for the object based on the distance to the viewer. In other words, several different models of the same object are created. This requires more memory and storage space, but the calculations for rendering the correct model is very fast [11]. Sudden changes of the objects can also create visual 'popping', making the transitions between models rough [45].

The latter method (CLoD) considers a function for the polygon mesh, usually based on distance, which is evaluated to create a temporary model of the object. This function is constantly evaluated during rendering and requires much processing power, which usually comes at the cost of performance. The advantage of this method is a smooth transition through the different levels of complexity, independent of speed and distance to the objects

[45].

Both of these methods can be applied to projects like this to enhance the visual experience [11].

Appendices

Appendix A

Hardware

The hardware used for the thesis is split into two sections:

- Section A.1 lists the hardware used to acquire the data in the field.
- Section A.2 details the hardware used to process the data.

A.1 Data Gathering Hardware

The data gathering hardware consists of a laser scanner and a camera.

Technical data for LMS-Z420i laser scanner: [37]

Measurement range for natural targets, $\rho \geq 80\%$	up to 1000m
Measurement range for natural targets, $\rho \geq 10\%$	up to 350m
Minimum range	2m
Measurement rate @ low scanning rate	up to 11000 pts/sec
Measurement rate @ high scanning rate	up to 8000 pts/sec
Vertical scanning range	0° to 80°
Horizontal scanning range	0° to 360°
Temperature range (operation)	0°C to +40°C

Nikon D200 digital camera: [21]

Effective pixels	10.2 million
Image sensor	RGB CCD, 23.6 x 15.8 mm, 10.92 million total pixels
Dimensions	147 x 113 x 74mm
Shutter speed	30 - 1/8000 sec
Pre-calibrated lenses	20mm, 50mm, 85mm

A.2 Data Processing Hardware

Two computers were used for processing the data.

Main processing computer: [26]

Processor	Xeon(TM) CPU 3.20 GHZ, Dual Core
Memory	2048MB RAM
Graphics card	NVIDIA Quadro FX 3400, 256MB RAM
Operation system	Windows XP Pro, SP2

Secondary processing computer: [26]

Processor	Intel Pentium M processor 1.87GHz
Memory	2048MB RAM
Graphics card	ATI Mobility Radeon X700
Operation system	Windows Vista Business

Appendix B

Software

Several software tools were used during the project, and a complete listing follows here, with particulars regarding what they were used for.

Software	**Usage**
RiSCAN PRO	Data gathering and preparation
Geomagic Studio	Data processing
Rational Reducer	Data reduction
AccuTrans 3D	Merging and format conversion
3D Studio Max	Rendering stereoscopic video
Visual Studio 2005	Implementing own code
CGAL	Geometrical libraries for own implementation
Boost	Mathemathical libraries for own implementation
AccuTrans 3D	Merging and format conversion
AccuTrans 3D	Merging and format conversion
AccuTrans 3D	Merging and format conversion

Appendix C

Data Flow

The data flow decision is related to both Format Selection and Data Limitation (Chapter 6.1.4). The flow sequence should be organized to have as few complexities as possible regarding bottlenecks and software/hardware limits.

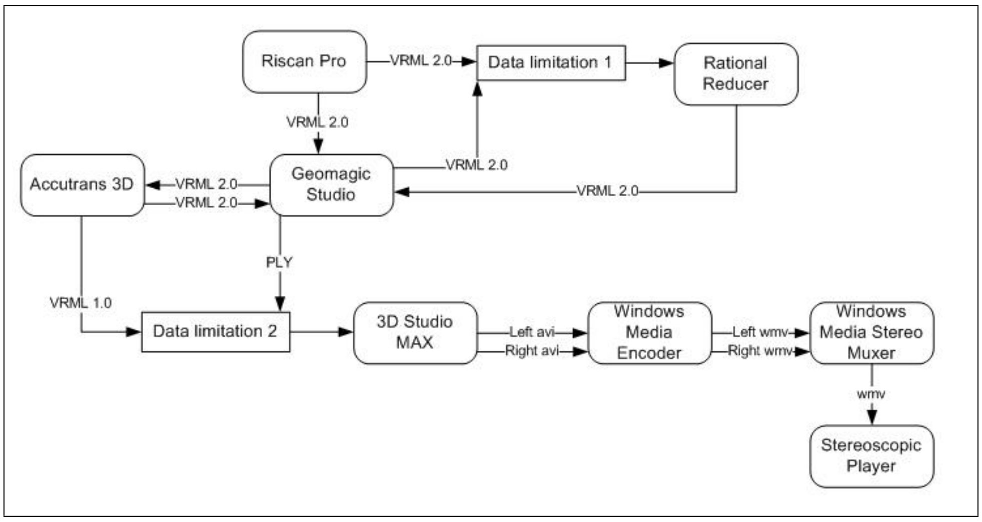

Figure C.1: *The flow of data.*

Figure C.1 shows the flow of data between the different tools, and which formats were used for the individual data transfers.

Bibliography

[1] 3DTV. Stereoscopic encoding guide for windows media. http://www.3dtv.at/Knowhow/EncodingWmv_en.aspx. Last visited: 2007-06-11.

[2] Peter K. Allen, Alejandro Troccoli, Benjamin Smith, Ioannis Stamos, and Stephen Murray. The Beauvais Cathedral Project (BCP). Technical report, Department of Computer Science and Department of Art History and Archaeology, Columbia University, and Department of Computer Science, Hunter College, 2002.

[3] Espen Almdahl. Guidet tur gjennom sverresborg. Master's thesis, The Norwegian University of Science and Technology, 2007.

[4] Autodesk. Autodesk 3ds max. http://www.autodesk.com/3dsmax. Last visited: 2007-05-08.

[5] Julie Dorsey Barbara Cutler and Leonard McMillan. Simplification and improvement of tetrahedral models for simulation. http://people.csail.mit.edu/bmcutler/PROJECTS/SGP04/talk/. Last visited: 2007-05-01.

[6] Gavin Bell, Anthony Parisi, and Mark Pesce. The virtual reality modeling language - version 1.0 specification. http://local.wasp.uwa.edu.au/p̃bourke/dataformats/vrml1/. Last visited: 2007-05-07.

[7] Paul J. Besl and Neil D. McKay. *A method for registration of 3-D shapes*, volume 14 of *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 239–256. 2 1992.

[8] Paul Bourke. Ply - Polygon File Format. http://local.wasp.uwa.edu.au/p̃bourke/dataformats/ply/. Last visited: 2007-05-07.

[9] Paul Bourke. A portable rear projection stereoscopic display. http://local.wasp.uwa.edu.au/p̃bourke/exhibition/vpac/theory.html. Last visited: 2007-06-17.

[10] Paul Chew. Voronoi diagram and delaunay triangulation. http://www.cs.cornell.edu/Info/People/chew/Delaunay.html. Last visited: 2007-04-25.

[11] James H. Clark. Hierarchical geometric models for visible surface algorithms. Technical report, Information Sciences, University of California at Santa Cruz, 1976.

[12] Web 3D Consortium. X3d international standards. http://www.web3d.org/x3d/specifications/. Last visited: 2007-05-14.

[13] Web3D Consortium. Vrml archives. http://www.web3d.org/x3d/vrml/index.html. Last visited: 2007-05-01.

[14] Sylvain Pion et al. Computational Geometry Algorithms Library. http://cgal.org/. Last visited: 2007-03-09.

[15] Fileinfo.net. 3ds file extension. http://www.fileinfo.net/extension/3ds. Last visited: 2007-05-08.

[16] G.Artese, V.Achilli, G.Salemi, and A.Trecroci. Automatic orientation and merging of laser scanner acquisitions through volumetric targets. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume 34, 2004. Part XXX.

[17] Inc. Geomagic. Geomagic studio. http://www.geomagic.com/en/products/studio/. Last visited: 2007-05-10.

[18] Ralph Grabowski. Boolean operations. http://www.design-drawing.com/visio/rgvt4.htm. Last visited: 2007-06-12.

[19] Okino Computer Graphics. 3d Studio .3ds Geometry Export Converter. http://www.okino.com/conv/exp_3ds.htm. Last visited: 2007-05-08.

[20] Karl Vincent Høiseth and Jan Arve Øverli. Computational Mechanics in Civil Engineering (CMC): 3-Dimensional Scanning and Structural Analysis. Technical report, Norwegian Marine Technology Research Institute, 9 2002. Case-study Nidaros Cathedral.

[21] Nikon Imaging. Nikon d200. http://nikonimaging.com/global/products/digitalcamera/slr/d200/index.htm. Last visited: 2007-06-15.

[22] Systems in Motion. *Rational Reducer Manual*. Included with the RR software.

[23] Systems in Motion. (SIM) advanced 3D visualization. http://www.sim.no/products/RR/. Last visited: 2007-04-12.

[24] Systems in Motion. (SIM) rational reducer. http://www.sim.no/. Last visited: 2007-04-12.

[25] Bahram Javidi and Fumio Okana. *Three-Dimensional Television, Video, and Display Technologies.* Springer-Verlag, first edition, 2002.

[26] Hans Martin Langø and Morten Tylden. 3d-scanning of the nidaros cathedral. Master's thesis, The Norwegian University of Science and Technology, 2006.

[27] K. H. Lee, H. Woo, and T. Suk. Data reduction methods for reverse engineering. *The International Journal of Advanced Manufacturing Technology*, Volume 17:735–743, 5 2001. Number 10.

[28] Marc Levoy. The digital michelangelo project. http://graphics.stanford.edu/projects/mich/. Last visited: 2007-06-14.

[29] VRML Site Magazine. Vrml site - 3D on the internet. http://www.vrmlsite.com/. Last visited: 2007-05-01.

[30] Microsoft. Windows media encoder. http://www.microsoft.com/windows/windowsmedia/forpros/encoder/default.mspx. Last visited: 2007-06-11.

[31] Daisuke Miyazaki, Takeshi Oishi, Taku Nishikawa, Ryusuke Sagawa, Ko Nishino, Takashi Tomomatsu, Yutaka Takase, and Katsushi Ikeuchi. The great buddha project: modeling cultural heritage through observation. pages 181–193, 2001.

[32] International Society on Virtual Systems and MultiMedia. Information portal for technology in cultural, natural and world heritage. http://www.virtualheritage.net/. Last visited: 2007-06-13.

[33] International Society on Virtual Systems and MultiMedia. Vsmm society. http://www.vsmm.org/. Last visited: 2007-06-13.

[34] MicroMouse Productions. Accutrans 3d - 3d object conversion software. http://www.micromouse.ca/. Last visited: 2007-05-22.

[35] RIEGL. Operation and preprocessing software riscan pro. http://www.sim.no/. Last visited: 2007-04-12.

[36] RIEGL. Riegl laser measurement systems. http://www.riegl.com/. Last visited: 2007-05-03.

[37] RIEGL. *Riegl Terrestrial Scanner z420i Manual*, 2007.

[38] Johannes Riegl, Nikolaus Studnicka, and Andreas Ullrich. Merging and processing of laser scan data and high-resolution digital images acquired with a hybrid 3d laser sensor. 2003.

[39] VicMan Software. Color correction wizard. http://www.vicman.net/colorcorrectionwizard/. Last visited: 2007-06-13.

[40] Stanford University. The stanford 3d scanning repository. http://graphics.stanford.edu/data/3Dscanrep/. Last visited: 2007-06-14.

[41] Stanford University. The stanford bunny. http://www-static.cc.gatech.edu/ turk/bunny/bunny.html. Last visited: 2007-06-14.

[42] Wolfgang Wieser. Creating stereoscopic left-right image pairs with povray. http://www.cip.physik.uni-muenchen.de/w̃wieser/render/stereo/create.html. Last visited: 2007-06-18.

[43] Wikipedia. 3ds max. http://en.wikipedia.org/wiki/3D_Studio_Max. Last visited: 2007-05-08.

[44] Wikipedia. Delaunay triangulation. http://en.wikipedia.org/wiki/Delaunay_triangulation. Last visited: 2007-02-15.

[45] Wikipedia. Level of detail. http://en.wikipedia.org/wiki/Level_of_detail. Last visited: 2007-06-12.

[46] Wikipedia. Perspective correction. http://en.wikipedia.org/wiki/Perspective_correction. Last visited: 2007-06-17.

[47] Wikipedia. Perspective correction lens. http://en.wikipedia.org/wiki/Perspective_correction_lens. Last visited: 2007-06-17.

[48] Wikipedia. Polygon file format. http://en.wikipedia.org/wiki/PLY_file_format. Last visited: 2007-05-7.

[49] Wikipedia. Texture mapping. http://en.wikipedia.org/wiki/Texture_mapping. Last visited: 2007-06-17.

[50] Wikipedia. Vc-1. http://en.wikipedia.org/wiki/VC-1. Last visited: 2007-06-11.

[51] Wikipedia. Viewing frustrum.
http://en.wikipedia.org/wiki/Viewing_frustum. Last visited:
2007-06-18.

[52] Wikipedia. Virtual reality modeling language.
http://en.wikipedia.org/wiki/VRML. Last visited: 2007-05-01.

[53] Wikipedia. Voronoi diagram.
http://en.wikipedia.org/wiki/Voronoi_diagram. Last visited:
2007-02-15.

[54] Wikipedia. Windows media video.
http://en.wikipedia.org/wiki/Windows_Media_Video. Last visited:
2007-06-11.

[55] Andrew Woods, Tom Docherty, and Rolf Koch. Image distortions in
stereoscopic video systems.
http://www.3d.curtin.edu.au/spie93pa.html. Last visited: 2007-06-17.