Hierarchic representation of 3D surfaces

A versatile multi resolution data model for digital surface representations

Jochen Steiner

Hierarchic representation of 3D surfaces

A versatile multi resolution data model for digital surface representations

Master's Thesis

at

Graz University of Technology

submitted by

Jochen Steiner

Institute for Computer Graphics and Vision(ICG),
Graz University of Technology
A-8010 Graz, Austria

01 Feb 2014

Advisor: o.Univ.-Prof. Dr. Horst Bischof

Hierarchische Repräsentation von 3D Oberflächen

Eine vielseitige mehrfach aufgelöste Datenstruktur zur Repräsentation digitaler Oberflächenmodelle

Diplomarbeit

an der

Technischen Universität Graz

vorgelegt von

Jochen Steiner

Institut für Maschinelles Sehen und Darstellen (ICG),
Technische Universität Graz
A-8010 Graz

1. Februar 2014

Diese Arbeit ist in englischer Sprache verfasst.

Begutachter: o.Univ.-Prof. Dr. Horst Bischof

Abstract

Digital models of 3D surfaces like a planet's terrain or the surface of buildings are needed in many applications, especially in the field of remote sensing. Due to progress in methods of attaining this information and users increasing their need for higher detailed models, the complexity of 3D surface representations has risen as well as the amount of data, including the incorporation of various levels of detail. In addition, digital elevation models commonly used in remote sensing cannot further serve the users' needs. Overhangs of rocky surfaces cannot be represented efficiently in DEMs. Large surface models like representations of whole tunnels can easily exceed the amount of several gigabytes. Surface models exhibiting different characteristics and resolution may be needed in the same scope and work. The results of different scanning or reconstruction technologies may require the proper combination of various resolutions. Close up photogrammetric reconstructions meet airborne laser scanner macro reconstructions. In our project, we address issues as regards reconstructing 3D surfaces concerning different fields of activity in remote sensing by providing a general method as to how to process, manage and store surface data with different levels of detail and resolution efficiently to cope with modern scanning technologies used for terrain and object reconstructions in remote sensing, as well as the rising requirements in representation and processing those surface models.

Kurzfassung

Digitale Modelle von 3D Oberflächen wie zum Beispiel das Gelände eines Planeten oder die Oberfläche eines Gebäudes werden in vielen verschiedenen Applikationen insbesondere im Bereich der Fernerkundung benötigt. Aufgrund des Fortschritts in den Methoden solche Oberflächeninformationen zu erhalten und des steigenden Bedürfnisses nach immer höhere detaillierteren Modellen stieg auch die Komplexität der 3D Oberflächenrepräsentationen genauso wie die Menge der Daten was die Einarbeitung von verschiedenen Detail Stufen eines Modelles inkludiert. Außerdem erfüllen *Digital Elevation Models* (DEMs), die in der Fernerkundung häufig verwendet werden, nicht länger die Bedürfnisse der Beutzer. Überhänge von felsigen Oberflächen können nicht effizient von DEMs repräsentiert werden. Riesige Oberflächenmodelle wie die Darstellungen von ganzen Verkehrstunnels können leicht die Datenmenge von einigen Gigabyte erreichen. Modelle mit unterschiedlichen Eigenschaften und Auflösungen können in der selben Anwendung im selben Kontext zur Anwendung kommen und die Resultate verschiedener Aufnahmetechnologien könnten eine passende Methode zur Kombinierung verschiedener Auflösungen benötigen. Als Beispiel sei erwähnt eine photogrammetrische Nahaufnahmen Rekonstruktion und ein Laserscan von einem Flugzeuggetragenen Scanner. In unserem Projekt sprechen wir Probleme bei der Rekonstruktion von 3D-Oberflächen in Bezug auf verschiedene Tätigkeitsfelder in der Fernerkundung an, indem wir eine Methode zur effizienten Verarbeitung, Verwaltung und Speicherung von Oberflächendaten mit verschiedenen Auflösungsstufen und Detailgraden bereitstellen um modernen Aufnahmetechnologien gerecht zu werden, wie sie für Gelände und Objekt Rekonstruktionen in der Fernerkundung verwendet werden und um die steigenden Anforderungen in Repräsentation und Verarbeitung solcher Modelle beherrschen zu können.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

Contents

List of Figures

List of Tables

Acknowledgements

I am indebted to my colleagues at the DITIGAL Institute of JOANNEUM RESEARCH who have provided invaluable help and feedback during the course of my work. I especially wish to thank my advisor, Horst Bischof, for his immediate attention to my questions and endless hours of toil in correcting draft versions of this thesis.

Special mention goes to my colleagues Bernhard Nauschnegg for helping me get back on track when I got on the wrong track and to Arnold Bauer and Gerhard Paar for their support of my work at all time .

Last but not least, without the support, motivation, understanding and tolerance of Kathi, my family, and my friends this thesis would not have been possible.

Jochen Steiner
Graz, Austria, January 2014

Credits

I would like to thank the following individuals and organisations for permission to use their material:

- The thesis was written using Keith Andrews' skeleton thesis [Andrews, 2006].
- The produced experimental results were rendered and visualized using a rendering tool by the team of VRVis [VRVis, 2013]. Special thanks go to T. Ortner and R. Tobler for the development of the visualization method and the basic data representation scheme.

Chapter 1

Introduction

1.1 Scope

This thesis aims to show a general purpose method for representing 3D object and terrain shapes addressing the needs of modern remote sensing applications and several limitations of existing methods. It will give an overview of existing methods and their limitations regarding the requirements of certain applications. Those may work with large terrain representations or detailed surfaces of buildings like long tunnels. Highly detailed models of smaller objects of special interest come along with coarse terrain representations of large scale. Models of different resolution and field of view of the same terrain can occur. A versatile multiple resolution model representation could handle both data sets in one model. This thesis will suggest a solution to address these needs efficiently.

1.2 Motivation

Many different kinds of computer applications are in need of representing, processing and displaying digital models of 3D objects. As the types of applications vary, the requirements for 3D surface representations do as well. For instance, in CAD applications, the amount of data to be processed may be a lot more manageable than the accuracy requirements. Several computer games present the user with large 3D worlds which must be maintained in memory, but the required accuracy is no where near that of a CAD injection molding tool which should be accurate down to parts of a millimeter. Techniques have been invented to display such large 3D terrains to computer gamers or store the information needed to construct precise tools. The field of remote sensing also faces the problem of processing very large 3D models which can vary significantly in detail or resolution but the assignments are different. Available surface data can alternate from large scale to highly detailed close-up models.

In the past years a lot of progress was made in the methods of attaining shape information. The users' need for higher detailed models increased. Therefore the complexity and the details of digital shapes or surfaces rose as well as the amount of data needed for 3D surface representations. In the field of remote sensing commonly used *digital elevation models* (DEMs) address neither the complexity nor the management of a large amount of data. Simple overhangs of rocky surfaces cannot be represented efficiently in DEM models. Modern surface representations must be able to hold arbitrarily oriented surfaces and extensive object models. Being a relevant application in the Austrian construction industry, reconstructing tunnels with the scanning software and hardware currently used does not uncommonly result in resolutions up to one 3D point per mm^2, which is why representations of long tunnels can easily exceed the amount of several hundred gigabytes. Reducing the resolution and model complexity of certain parts of the currently processed or viewed model without changing the data structure itself could reduce the quantity of data processed and stored in the memory dramatically and preserve the

model's information for further processing. An important issue is adding new surface information to an existing model and representing different detail variations of the same object or shape. Close-up reconstructions of surfaces which are part of a large scale terrain or object can be a worthwhile extension to a previous reconstruction or scan of that large surface. The new reconstruction would expand this model with highly detailed surface data of regions of specific interest within the object. Resulting models would show different resolutions in specific areas of their surface. In this thesis we intend to develop an efficient data structure which satisfies these needs. The idea of our versatile multi-resolution surface structure is to enable surfaces in 3D space with arbitrary combinations of resolution levels. Our structure should provide a level-of-detail pyramid and a breakdown into so called patches for efficient memory management.

1.3 Requirements

In remote sensing, one important issue is the reconstruction of existing objects or terrains into digital surface models. Applications in this field face several challenges. This thesis intends to address the needs of the applications emerging from these challenges with a novel surface representation system.

In remote sensing, a large variety of object shapes is of interest. Limitations in the composition of the digitally represented surfaces should be avoided. A representation of arbitrarily oriented and positioned surfaces or surface parts must be possible. As an example, in remote sensing, the use of DEM representations for terrain surfaces has been very common up to now. DEM representations and their characteristics will be defined and discussed in more detail later, in Section 2.2.2. One very important characteristic is the limitation of not being able to represent arbitrary surfaces. Objects or object parts can not be oriented or positioned in any order because multiple heights can not be described by DEMs. A surface represented by a DEM is projected to a plane spanned between two coordinate axes. When assuming for coordinate triples x, y, z the projection plane lies between the x, y axes, which means for a specific tuple of coordinates x, y there is only one coordinate z possible. For example, overhangs on rocky terrains or cave structures cannot be represented by DEM models. Constraints in this vein or similar ones invoked by the type of data representation restricting the types of shape modelled must be omitted. Nor may the dimension of the surface be limited by the representation. As landscapes of interest may exhibit a wide circumference, their digital models may show the same characteristic.

In the process of generating or reconstructing 3D objects, the resolution of the input data can change by using different hardware or software. Furthermore, different object parts are of different interest. Some regions may be analyzed in more detail than others and therefore different resolutions or detail levels are required. A novel and versatile surface structure needs to allow an efficient representation of various resolutions or grades of detail for differing regions within the model. The transitions between these regions must not, however, result in holes in the surface. The models must exhibit a watertight connection between parts with different resolutions.

Existing surface models must also be able to be updated or expanded by adding new data with different resolutions to the surface or parts of it. When different hardware or software technologies allow different detailed and scaled object reconstructions, several models of the same surface can exist, extending or complementing each other. As an example, a certain region of an existing large-scale surface model of a building or a natural terrain which had been previously reconstructed could be of special interest to the researchers. A stereo microscope 3D reconstruction of that part can add essential information to the existing model. It would also lead to a high resolution difference between areas within one and the same model.

Different applications are in need of overview representations of the processed surfaces. These represent a coarser version of the same shape and can co-exist along with the original surface version. This shall allow an efficient way of performing algorithms or calculations in a lower resolution of the same surface. A reduction of computation time of several algorithms without having to change the data struc-

ture itself or losing information on the original surface data can be achieved in that way. The cost for gaining time is the loss of accuracy. In some cases, this may be affordable and the benefit of faster calculations justifies this loss. As an example, the registration of two surfaces results in a transformation to transform both surfaces into the same coordinate system. Performing those on a lower level of both surfaces can be used to find the transformation parameters in a much shorter time before transforming the original data into the same system. To satisfy the need for computational efficiency, our data structure must feature several overview representations for each model in different resolutions to provide the ideal level for different calculations.

Ortner et al. [2010] presented an approach to visualize infinite surfaces with arbitrary levels-of-detail. The basis for these visualization techniques is the so-called *Ordered Point Cloud* file structure. 3D and texture data to be rendered and visualized is stored in a set of point arrays and raster image files which are linked up within xml files. We intend to take advantage of Ortner's representation scheme to present our models and results. For this reason this work must be compatible with this file structure to make use of it. It must be able to write the essential parts of the file structure which are at least necessary for successful visualization and transfer our model data to it.

The envisaged thesis will address issues in reconstructing 3D surfaces regarding different fields of activity in remote sensing by providing a general purpose method how to efficiently process, manage and store surface data with multiple resolutions. We will analyze methods of surface reconstruction and representation for similar purposes and generate a trade-off between our intended method and existing ones with respect to the following aspects:

- illustrating various levels of detail in large models
- the possibility of processing very large amounts of data
- representing arbitrary surface structures
- flexibility in adding surface data to existing models in different resolutions

The expected results will be usable for managing 3D surface models with levels of detail differing locally. Very large 3D surface models which additionally require a very high level of detail such as digital representations of tunnel surfaces for monitoring and maintaining will be manageable.

1.3.1 Target object shapes and applications

The main object of this thesis is to serve applications in remote sensing and close-range photogrammetry, particularly for high-resolution surface representations in the field of construction. In the field of remote sensing real-world objects, both natural and man-made, are surveyed. The methods and technologies of remote sensing are outlined and discussed in Section 2.2. Mainly surface representations of different terrains, construction surfaces or landscapes are of high interest. In this thesis, surface models representing terrain are the first target shapes and the focus is on satisfying the needs of applications which mainly work on those surfaces. Our experiments and analyses will be performed on such terrain surface representations as commonly needed in remote sensing. The experimental results achieved with models of that kind are shown in Chapter 4. Chapter 5 will will describe the future prospects of this thesis. As a part of that, an overview of the potential application, in which this method can be used and which can take advantage of it, is given in Section 5.6 before this thesis will be concluded in Chapter 6.

Chapter 2

State of the Art

In computer graphics, a huge variety of different object shapes and types are created, processed and visualized, which is why many differing methods of object representations have been developed to describe their surface characteristics as accurately as possible. Foley et al. and Hearn et at. provide a rough overview of the different possibilities of representing three-dimensional objects in computer graphics [Foley et al., 1996] [Foley et al., 2009] [Hearn and Baker, 1996], [Hearn et al., 2010].

2.1 Three-dimensional object representations

2.1.1 Surface representations

As the most commonly used surface description for 3D objects, Hearn and Baker [1996] refer to polygon meshes as ”standard graphics objects”. Polygon meshes are composed of a set of vertices, edges and polygons. The advantage of polygon meshes is the possibility to represent arbitrary forms of objects in various levels of detail. But this representation technique can be very memory hungry and the representation of curved surfaces can only be approximated. Curved line or surface representations, which are based on mathematical functions, are the better way of describing any curved shape. They are used to describe smooth, curved surfaces like spheres or surfaces which can be approximated by geometric objects. As examples quadric surfaces [Hearn and Baker, 1996], [Mortenson, 1997] and spline [Hearn and Baker, 1996], [de Boor, 2001] representations should be mentioned. Quadric surfaces are described with second degree equations *(quadratics)*. Spheres or ellipsoids would be examples of such surfaces. A spline curve is a piecewise defined polynomial function satisfying specific continuity conditions at the connecting points. A spline surface can be defined by two orthogonal spline curves. The advantages of these surface representations are their efficiency in memory usage and their precision in defining curved and geometric shapes. In contrast defining arbitrary, large surfaces can become very difficult, particularly if they need to remain watertight. Reconstructed surfaces consist of a number of discrete points describing the shape of the surface. A curved surface function is a continuous approximation of this set of discrete points. The more points involved and the more arbitrary a shape gets, the more complex the describing function grows. A piecewise functional shape description involves the problem of creating a watertight surface at the borders between the curved surface patches. In reference to the field of remote sensing, we will elaborate on to the subject of surface representations later, in Section 2.2.

2.1.2 Volume representations

For solid modeling or the representation of a volume instead of a surface, the inner and outer regions of an object must be defined. That means the volume of 3D space is separated into an inner and an outer volume. One form of solid modeling is boundary representation. Boundary representations (B-reps)

[Foley et al., 1996] are sets of surfaces like spline or polygon patches which satisfy the condition of defined inner and outer regions in order to form solid objects. Another way to describe closed shapes are space-partitioning or spatial-partitioning representations [Foley et al., 1996] or finite element models [Hearn et al., 2010]. Objects are assembled by a portion of adjacent, non-overlapping and primitive solids. The simplest way of space-partitioning is cell decomposition [Foley et al., 1996]. An object is a union of non-overlapping primitives, which do not have to be of the same type. Constructing the shape by arranging identical cells in a regular grid is the special but well known case of cell decomposition. Typically, these cells are cubes. Cubic cells in space partitioning are commonly known as voxels - volumetric elements. This method is also called spatial-occupancy enumeration. To address storage efficiency, octrees as a hierarchical variant of spatial-occupancy enumeration was developed by a number of researchers independently from the late 1970s to the early 1980s [Samet, 1984]. They were derived from the 2D quadtree and form a 3 dimensional hierarchical voxel composed object. Quadtrees were also introduced independently by several researchers from the late 1960s to he tearly 1970s [Samet, 1984]. As a plane in 2D is recursively divided into quadrants, the 3D space is divided into octants in each dimension. Each of them may be partially full, full or empty. Partially full octants are again divided until all resulting octants are homogeneous. Binary Space-Partitioning trees (BSP) represent another form of the hierarchical division of space. The use of BSPs for space partitioning was introduced by Thibault and Naylor [1987]. While octrees divide 3D space by three planes normal to each other into eight equal portions, BSP trees divide space into pairs of subspace by one arbitrarily oriented and positioned plane. This can address the distribution of object parts in space better and therefore reduce the tree depth and search time. Other well-known methods of solid modeling are sweeping representations, where a 2D shape swept through space defines a 3D object, and constructive solid geometry, which creates new objects by applying Boolean operations or transformations to other objects. A resulting object is defined by a tree of operations in the inner nodes and simple primitives at the leaves.

2.1.3 Usability in remote sensing

In the field of remote sensing, closed surfaces or solid objects are not commonly used. Surface representations instead of volumetric representations are more beneficial. In many cases, only parts of real-life objects are reconstructed, digitalized and analyzed. When reconstructing a planetary terrain, only a part of the planet's surface from a specific point and direction of view is commonly surveyed. For example, recording the inner surface of a transport tunnel for maintenance produces only the digital model of one side of an object. To construct a solid model, every part of the object's surface bordering the inner volume from the outer volume must be known. When speaking of transportation tunnels, the inner surface can be captured but the outer surface facing the rocks of the mountain can not be recorded. Very often in remote sensing, objects can not be recorded from every point of view to create a closed surface clearly defining inner and outer volume. Furthermore, for typical remote sensing applications, solid modeling is not necessary.

Curved surfaces can be very accurate for geometric objects or shapes and are efficient in memory usage. Based on mathematical functions, they are designed to represent surfaces described by them or which can be approximated efficiently by geometric objects. However, this is not that helpful with arbitrary shapes as present in nature. Rocks on a planetary terrain do not follow mathematical concepts. Therefore, curved surfaces are not suitable for our field of use, either.

We will focus on polygon mesh surfaces throughout our project because they are most suitable for arbitrary surfaces which do not represent closed objects.

2.2 Surfaces in remote sensing

There are several ways to gain surface information in remote sensing. Most remote sensing applications result in surface data of a certain part of a terrain or building which is intended to be analyzed in its digital form. Because geological formations and other arbitrary natural objects are not constructed in their way of appearance, the resulting models must be capable of showing arbitrary shapes. This can be achieved by using polygon meshes to approximate the recorded surfaces. To save memory, instead of 3D meshes, 2.5D heightmaps are often used in remote sensing applications, such as for example in [Gallup et al., 2010]. Li et al. [2010] mainly speak of regular gridded surface models and triangulated meshes as well when describing terrain modeling techniques.

2.2.1 Reconstruction methods and technologies

This section summarizes common 3D data acquisition methods used in remote sensing. Li et al. [2010] give an overview methods commonly used in remote sensing. They identify the different methods of gathering the surface data of terrains and objects in this field from pictures, by radar, by laser scanning or by GPS surveys. Attaining 3D data from pictures is called photogrammetry.

While photogrammetry definitely means obtaining 3D information from pictures, definitions differ in detail. While Li et al. [2010] strictly separate laser scanning and photogrammetry, according to Kraus [2007], photogrammetry means the reconstruction of an object's position and orientation, not only from either photochemical or photoelectric images, but from laser scanner images as well, as long as the focus is on the geometric characteristics of the object. The basic idea of photogrammetry from photoelectric or photochemical images is a stereo reconstruction from a pair of overlapping images. By using image matching techniques, pixel correspondences can be calculated. For each pixel pair, an intersection equation can be solved by knowing the cameras' position and orientation. By contrast, laser images consist of distance information image elements directly.

Radar-based remote sensing applications are based on the recorded echos of radar systems. Similar to photogrammetry, a stereo reconstruction can be performed using two images of the intensity of the radar echo of the same terrain. Additionally the phase shift between two radar images of the same area can be used to retrieve topological information. These systems are independent of weather conditions or day light intensity.

TLS (terrestrial laser scanning) and ALS (airborne laser scanning) produce 3D point clouds by measuring the time between sending and receiving a laser signal and calculating the distance of the hit object to the sensor. While TLS systems are fixed, ALS systems use GPS systems for determining their own position. ALS-based data acquisition methods have gained in importance to record large scale terrains. An early approach of recording large wooden areas by ALS systems was proposed by [Kraus and Pfeifer, 1998].

All these technologies have different advantages and fields of use. Regarding accuracy, costs and speed they produce different results [Li et al., 2010]. This is also illustrated in Table 2.1. Furthermore, the efficiency of each technology varies according to the size of the recorded terrain. For instance, it is not useful to use ALS for the reconstruction of an area of a few m^2. What they have in common is that objects or terrains are recorded or scanned from one specific viewpoint and the result represents the object from that one perspective. The result is mostly a surface structure and not solid objects [Li et al., 2010] [Kraus, 2000].

Nevertheless, there is a huge variety of further methods to attain 3D information about real-life objects. For example, there approaches using sonar, tactile sensors or structured light [Allen and Michelman, 1989] [Elfes, 1989] [Zhang et al., 2002]. Moreover, 3D sensing became a field of interest for the entertainment industry in addition to its professional use [Shotton et al., 2013]. Several gaming consoles introduced motion sensing input devices along with connected games. Because we focused on the needs

Acquisition Method	Accuracy	Speed	Cost	Application domain
Photogrammetry	Medium to high (cm-m)	Fast	Relatively low	Medium to large areas
Space Photogrammetry	Low to medium (m)	Very fast	Low	Large areas
Radargrammetry	Very low (10 m)	Very fast	Low	Large areas
ALS	High (cm)	Fast	High	Medium to large areas

Table 2.1: The table compares the different data acquisition methods for terrains in remote sensing according to [Li et al., 2010].

of remote sensing with this thesis and a complete evaluation of methods acquiring 3D information is out of the scope of this work, this summary remains incomplete.

2.2.2 Surface mesh representation techniques

A very simple and efficient option to depict associated point data is a heightfield or heightmap, also called *digital elevation model* (DEM), which was mentioned previously in Chapter 1. As heightfield is a 2D scalar field with each 2D point linked to a scalar value which describes the height. The height can also be described in function form as $z = f(x, y)$ [Foley et al., 2009]. Regular grids are very common despite the fact that irregular distributions are possible, so raster image file formats are able to store these heightfields. The association between the 2D points is implicitly stored by the pixel grid and the pixel values store the heights of each point. That means the surface is projected onto a plane which is the 2D scalar field. Each vertex of the surface is described by a certain point in that field and a height value describing the length of an orthogonal vector to that plane or field. This is a 2.5D data format due to the fact that for each point (x, y), only one height z is possible. Arbitrary 3D structures can not be stored in a simple heightfield. Not every structure with multiple heights like overhangs or caves can be represented. As one solution to this deficit, multiple heightfields were introduced [Gallup et al., 2010]. These allow a certain number of heights per 2D point by providing several layers of height maps for one model.

A more arbitrary, memory hungry and complex method of 3D point association into a mesh are polygon meshes. They can be represented by several data representations. A popular example is the *VRML Standard* which was introduced in [Carson et al., 1999]. Vertices, edges and polygons must be linked together explicitly. Very common polygon meshes are triangle meshes, often called TIN (triangulated irregular network).

Surface meshes are still very domination in computer graphics [Foley et al., 2009]. Several mesh or point cloud representation schemes have been developed. In the past few years, the need for accurate 3D sensing hardware increased, as 3D scan or model precision became more and more of an issue. As an example for a modern library, the *Point Cloud Library* (PCL) should be mentioned, which is an advanced approach to the wide area of 3D perception and point cloud processing addressing the rising needs of modern applications [Rusu and Cousins, 2011].

2.3 Simplification and level of details algorithms

With the rising complexity of polygonal meshes, the requirements in computing capability and memory need have risen as well for visualizing and processing models. Many applications need high detailed models for realistic presentation or accurate computation, so models are often created or reconstructed with a high resolution. But not every part of the model is needed in the highest possible resolution and

not every part of an application requires highly detailed data. Many algorithms have been presented to address these issues.

Hoppe [1996] introduced an edge collapsing algorithm to simplify polygon meshes. Hoppe's progressive meshes transform a mesh into a simplified version of itself by means of a sequence of edge collapse operations. This algorithm was later refined to a real-time algorithm depending on the viewing parameters of the users camera [Hoppe, 1997]. Garland and Heckbert [1997] presented an algorithm based on vertex pair contractions and error quadrics. The error quadrics at each vertex approximate the costs for each contraction. Lindstrom and Turk [1998] also used edge collapse operations. They especially addressed the high memory demand for large models. Gigabytes of 3D data made in-core algorithms inapplicable for many models. In [Lindstrom, 2000] an out-of-core algorithm using several ideas from [Garland and Heckbert, 1997] and [Lindstrom and Turk, 1998] was introduced. Rising computation capabilities and the increased use of GPU algorithms created several parallel computing GPU simplification algorithms in the last years [DeCoro and Tatarchuk, 2007] [Strugar, 2009] [Hu et al., 2009].

Overview representations with different resolutions for each model will be an essential part of our work. Besides, the order of the vertices in our grid is one essential criterion. Because the common simplification algorithms take care of model specific structures when reducing vertices but not of data structure specific ordering, additional challenges arise when applying common simplification principles to this work. Therefore, advanced simplification techniques were not part of our work but are considered to be future work.

2.4 Co-registration and merging 3D surfaces

As mentioned previously in Section 2.2.1, reconstruction in remote sensing results in surfaces depicting an object or a part of an object from a specific point of view. This can be a part of a landscape recorded by an airborne laserscanner. It can also be one rock face of a cave reconstructed by photogrammetry from photoelectric images. In many cases, there are different data sets which are related to each other and can be joined in the same surface model. One such example is the result of the next flyover containing the neighboring part of the previously recorded landscape or another rock face of the same cave reconstructed using other images from a different viewpoint.

2.4.1 Co-registration

Despite being part of the same object, models from different sensor sources are commonly recorded in their own local coordinate system and in order to be merged into one, they must be transformed into one common coordinate system. That is mostly called registration. An example of co-registration is illustrated in Figure 4.7b. The point correspondences found in the illustrated point cloud datasets are marked with red lines.

A very frequently used method is the *Iterative Closest Point (ICP)* algorithm. Many different variations of it have been developed over the past years. The early ones were [Besl and McKay, 1992], [Chen and Medioni, 1992] and [Zhang, 1994]. It basically searches for pairs of nearest neighbors in both point clouds, estimates a transformation which aligns them and then applies the transformation to one point cloud. This is iterated until the point clouds converge. Gruen and Akca [2005] proposed a new technique for matching 3D surfaces based on the Generalized Gauss-Markoff model of least squares. They estimate the transformation parameters for a 3D point cloud to match it to a template surface by minimizing the sum of squares of the Euclidean distances between the surfaces.

Registration algorithms are essential when merging point clouds which have not be created in the same coordinate system. This may happen due to calibration error or previously applied transformations. Before joining them int the same model, they have to be transformed into the same coordinate system

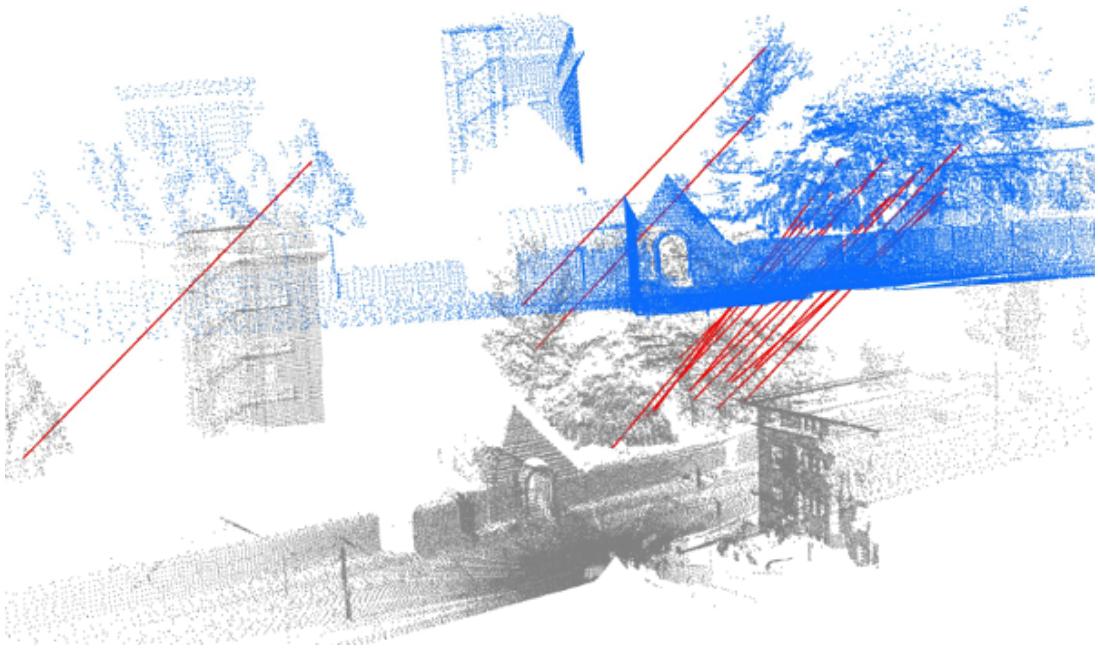


Figure 2.1: This figure shows an example of co-registration. The corresponding points between two point cloud datasets are found and illustrated. This image is taken from the *Point Cloud Library* documentation website [PCL, 2014a].

to depict the same surface. Co-registration is not a core part of this work. There have been numerous approaches to co-registration issues in the past and it was out of the scope of this project to include an extensive research on that field. Nevertheless, as an important step to complete surface integration of surfaces which are not aligned, it has to be considered to be an issue of future work how to integrate co-registration into our system.

2.4.2 Surface integration

To represent a combined model, separate registered surface models must be integrated into one structure. Basically the existing approaches can be divided into two kinds of methods, depending on the input type of the data. One kind integrates unstructured point data while the other works with structured data with knowledge about the object's shape either in surface or volumetric form [Johnson and Bing Kang, 1999].

An example of merging unstructured 3D data is the approach of [Matiukas and Miniotas, 2011]. Their work is focused on the registration process of unstructured laser scanner 3D point data. They divide their merging process into a pre-alignment method and a refinement step. According to their work, the merging process is finished after the refinement of the registration.

Soucy and Laurendeau [1992] proposed an approach integrating several registered range images from multiple viewpoints into a non-redundant surface triangulation. They construct a Venn diagramm of the range views determining the surface intersections of the different views' points. Each of the views describes a set of triangulations and a canonic subset of the Venn diagram. Overlapping parts are merged by a weighted average, thus eliminating redundancies and overlaps. The holes between the now non-redundant surfaces are then interpolated by a Delauny triangulation. In [Dorai et al., 1998] an approach to integrate registered data into a seamless surface was proposed. They used a weighted average mechanism on registered depth images to create their integration model. The integration model is calculated by averaging the overlapping depth pixels and weighting them by their distance to the nearest boundary.

In the proposal introduced by Johnson and Bing Kang [1999], a volumetric data structure is used

to sample the function. The surfaces are registered using the *ICP* algortihm and are accumulated in an occupancy grid which does not only store the likelihood of occupancy, but additionally the consensus surface normal in each voxel. It is stored as the weighted surface normals of nearby data points. The surface of the merged input is recovered by ridge detection in surface likelihood and the surface function is polygonized using the Marching Cubes algorithm [Lorensen and Cline, 1987]. The texture of the integrated surface is merged by weighting overlapping textures of the original data sets. The weighting depends on the original surface normal and the consensus surface normal.

A range image fusion pipeline for object reconstruction and modeling based on a kd-tree search was proposed by Li and Wee [2004]. It consists of range image registration taken from multiple views, data integration, smoothing and resampling the integration result. For the integration process the input data sets are indexed in a kd-tree [Friedman et al., 1977] for a fast nearest neighbor search. Overlaps found by searching the tree are eliminated in all neighboring views.

Surface merging also find assignment in the entertainment sector. An approach fusing several depth measurements into a global volumetric model was proposed by Newcombe et al. [2011]. They showed the reconstruction of dense surfaces in real-time by using a moving depth sensor from the Microsoft Kinect system.

Grzeszczuk and Pelizzari [1996] proposed a solution to integrate several overlapping surfaces at visualization time. It was designed to combine multiple overlapping surfaces in one view avoiding occlusion artefacts. Overlapping polygons are evaluated by their quality in real-time and polygons of less quality are deleted from the view.

Surface merging or integration is an essential part of this work as already stated in Section 1.3. Because the integration of aligned surfaces is not only intended to address visualization issues in this work, approaches combining surfaces at visualization time are of less interest for this thesis. Fused surface representations are meant to be processed automatically. Therefore, the surface fusion must be finished before further processing and not only at the time of visualization. A new surface integration algorithm will be described in this thesis along with a new surface representation method. Experimental results based on this method will be shown in Chapter 4 and will also demonstrate the capabilities of the new surface representation technique.

Chapter 3

Technical Realization of our Versatile Surface Model

This chapter describes the conceptual design decisions as well as the evolution of the concept, structural hierarchy, memory management, file system management, data access, data content and an approach to merge individual 3D models into multi-resolution models will be explained and justified.

With our *Versatile Surface Model* we propose a solution for surface representations with locally differing multiple resolutions supporting very large surface models.

3.1 The idea of arbitrary multi-resolution-surfaces

In remote sensing, many objects of interest are very large. Airborne laser scans of a whole region can exceed the capacity of several hard disks. Even highway tunnel scans may have several gigabytes. Common hardware can easily result in resolutions of one 3D point per mm^2. By estimating 20 m for one tunnel pipe's circumference and presuming $3 * 32$bit floating point, values per point a digital scan of, for example, the Plabutsch Tunnel in Austria would result in 2.2 TB of 3D data. To process such large amounts of data, we decided to partition a surface model into smaller portions of data which are connected to each other. When processing a model only the parts containing the data in the area of interest are loaded into the main memory.

On the other hand, the region of interest for interesting objects can be very small while very highly detailed information about it is needed. For instance, reconstructions of rocky surfaces recorded with a planetary rover on a distant planet need to exhibit very high detail to allow the retrieval of specific geological information. This work allows handling both large scale models and highly detailed reconstructions. In some cases it may be interesting to show the context of some detailed models. Thus this work furthermore enables applications to combine large-scale surfaces and detail reconstructions into one model.

The initial idea to provide this functionality and moreover to address the needs stated in the requirements Section 1.3 was to arrange a set of plane patches arbitrarily in 3D space. Each of these patches can be positioned and oriented freely in space. These patches serve as a ground projection plane for the surface parts on it. A regular grid is attached onto each of them describing orthogonal distances on it similar to a height map or an elevation model (DEM) in Cartesian space. Every patch is allowed to have its own resolution and its own level of detail pyramid description. Neighboring patches need to be connected to form a closed surface. Each border point in neighboring patches must have a specific neighbor point. To support different grid sizes, one point may have several neighbors in the other patch. By varying the position, orientation and size of the grids of the patches, an arbitrary surface can be approximated. Nevertheless, the system takes advantage of the memory efficient and intuitive way

of storing the information in a gridded height-map. A cave could be modeled by arranging the surface patches approximately around the cave's volume.

To describe more complex surface geometries, the system shall also support not only planar patches but also different geometric systems. For example, cylindrical patches could describe objects of similar shape to tubes, handrails or wires. Different sensor geometrics may also result in surface representations requiring different coordinate systems for the surface patches. TLS systems can result in spherically oriented surfaces or line-wise scanned cylindrical surfaces. To accommodate these circumstances, the coordinate system of the patches of the model should reflect the geometric system used by the recording sensor. Therefore, different geometrics than only planar ones are valuable for both the geometric characteristics of the represented object and the recording sensors geometric characteristics.

This would avoid the limitation of a single DEM of not being able to model multiple heights. Wide, flat surfaces without much detail information could be described by a few patches which are large in scale and feature a low resolution while complex structures in need of exact details can be modeled by many patches of a small field of view, with each exhibiting a high grid resolution. Although such a solution is not as flexible as a mesh or point cloud based structure it is far more versatile than the commonly used DEM surface representation. It preserves the DEMs memory efficiency while avoiding its main limitations. Furthermore, because DEMs were widely used in remote sensing in the past years many existing algorithms and results are based on them. The use of this idea makes integration of this work into existing systems easier than the use of an approach based on a completely different surface representation technique.

3.1.1 Multiple resolutions

As already mentioned in Section 1.3 the system is required to illustrate multiple resolutions in one surface model. On the one hand it is needed to represent several overview levels of the object's shape to allow efficient processing where the full resolution is not necessary. This could be achieved by allowing each DEM patch its own level of detail pyramid. Regular gridded digital elevation models can be represented easily by raster images. These can be reduced in a very straight forward way to several overview levels with lower resolutions. For that reason, the use of DEMs is also a big advantage for computing individual overview levels of the surface.

On the other hand the representation system is required to enable surface updates with more detailed surface parts of certain regions in the model. This could affect any part of the surface data in the model and therefore any patch or set of patches. New surface details could be added on top of one or more existing patches. By allowing individual resolutions of the patches, the new surface can be integrated easily into new patches created in the exact sizes needed. Nevertheless, by having projected height maps on each patch, major challenges arise. The new data has to be integrated into existing surface parts which are already projected to a specific geometry on their respective patches. The regression plane of the new surface data may be normal to the plane on which the existing data has been projected to. For instance, the existing surface could be scanned by sensors mounted on a satellite or a plane resulting in a bird's eye view. The new detail data to be added to the model may be a scan of the terrain's surface showing the face of a rock wall which is normal to the terrains surface. The integration of such data into the existing plane patches would lead to an enormous loss of data. To conserve the data the most in the region where the data has to be inserted, the patches have to be divided into parts and new patches providing a fitting geometric projection have to be inserted. In that case, it could be very hard to connect these surface patches and sew the surface together along the borders of the patches.

Connecting and sewing together geometrically projected DEM patches of various different resolutions could be a very hard task, especially when inserting new surface details into an already existing surface model. For more complex structures, some geometric systems might not be useful. This represents another limitation of the DEM patch concept although the general restrictions of a DEM representation can be avoided. Constructing a versatile surface model by using a point cloud may be much more

suitable in this case.

3.1.2 A file system for visualization

To visualize the surface models and the results created by the surface representation system with the visualization method proposed by Ortner et al. [2010], it is required to serve their *Ordered Point Cloud* file structure. This file structure represents the surface as a set of quadratic patches. Each patch contains a 3D vertex array and a portion of the texture of the surface. The vertices are implicitly linked together to a quad mesh by their neighborhood relations. The surface exhibits several overview levels ordered into a level of detail pyramid. Each level contains a set of reduced patches showing the same surface in a lower resolution. This file system is described in more detail later, in Section 3.5. The system must be able to write these data files correctly to enable the visualization system to render and present a complete and accurate surface model to the user.

The usage of arbitrary height-map patches brings up further issues. Arbitrarily positioned and oriented DEM patches have to be converted to portions of a point cloud. What's more, each of the DEM patches can exhibit its own individual geometric reference system, an individual resolution and a level of detail pyramid. Transferring the data within these highly inhomogeneous patches to a file structure represented by a set of homogeneous patches runs the risk of loss of data or accuracy during conversion. For example, two directly neighboring DEM patches with highly different resolutions exhibited have to be connected to each other to form a closed surface. The border vertices in the patch with the lower resolution would have many direct neighboring vertices in the other patch. In the *Ordered Point Cloud (OPC)* file structure, the raster neighborhood defines the quad mesh. Each vertex has a fixed number of four neighbors in its raster neighborhood which are linked together by edges to a quad mesh. These are the left, right, upper and bottom neighbor. In the above-mentioned example, some vertices would feature a number of left, right, bottom or upper neighbors. Therefore, a straight-forward conversion of the surface parts at such border areas would not be possible. A conversion of a DEM patch based file system to the *OPC* file system would be lossy and complex in computation.

As already described, some surface models illustrating large objects or terrains may grow very large in hard disk space. The system must avoid unnecessary limitations also in size and scale. If models could grow so large that they reach the size of a whole common consumer hard disk there would also be a massive amount of data to convert to the *OPC* file structure. Performing a complex conversion mechanism from one hard disk sized surface model to another very likely hard disk sized surface model of a different format would require an overwhelming amount of computation time.

Especially for enabling a visualization support for Ortner's method, a surface representation based on height-map patches seems to be very ineffective. On the one hand, the quality of the visualization model could suffer noticeably. On the other hand, the complex conversion method necessary to transfer the models data into the target format would call for a ridiculously high computational effort. Especially when handling very large models the conversion time could get out of hand. A different basis for a versatile surface model could help handle these issues more effectively.

3.1.3 Hierarchically ordered point cloud patches

The previous sections, described which challenges arise when realizing the initial idea of assembling a surface model from a set of DEM patches. Facing these, we found parts of this initial idea not only challenging but also inefficient and unsuitable for our system. To be able to implement a system as defined by the requirements in 1.3, the concept had to be re-factored and refined. Dividing a surface into several connected parts is a good solution to work with large amounts of data. Each part of the surface can be processed incrementally if the whole surface is to large to fit into the memory. Additionally, different surface parts can show different properties, like resolution for example, to describe locally differing surface characteristics in a better way. Because of that, representing a surface by a set of patches and

being able to process them incrementally is suitable for the goals of this thesis. Nevertheless, to cope with multiple resolutions of the same surface and to be able to extend one model by adding or inserting detail data for certain regions of the surface, a more suitable data representation within the patches has to be found.

Point cloud meshes are a very flexible representation for 3D surfaces. Although sets of DEM patches allow much more freedom than a single DEM, surface patches containing point cloud based meshes provide additional flexibility. Extending the surface by new data is not restricted to any projection plane. It becomes easier to mix different sensor geometries and therefore easier to add new data to an existing model. In the *Ordered Point Cloud* of Ortner, several hierarchy levels of patches were used to provide overview levels of the surface. By making use of this idea new, data can be added at the correct hierarchy level suitable for the resolution of the new data. Creating a hierarchy pyramid for overview levels of lower detail can also be used to create sub-pyramids for parts of the surface featuring more information with high detail. By building a surface model from hierarchically ordered point cloud patches instead of DEM patches, the difficulties of adding new information and representing several detail levels of the surface from different sensor sources can be decreased.

Additionally, serving the point cloud based *Ordered Point Cloud* file structure becomes more efficient by the use of point clouds in our own system. But the data transfer of inhomogeneous patches which can vary in grid size totally arbitrarily to homogeneous ones of a fixed size is still an issue. This challenge is increased by a characteristic of Ortner's visualization process. Once transferred into the *Ordered Point Cloud*, the vertices must form a closed surface by being link into a quad mesh by their neighborhood relations. However, each patch's point cloud data is rendered separately depending on the field of view, the patch's hierarchy level and the camera position. Vertices in different patches, although neighboring in the surface, are not connected by edges. That could result in holes in the surface when rendered and visualized. To show a watertight surface mesh to the user, neighboring patches must feature overlapping border vertices. That means that when transferring the surface data to the file system, the overlapping borders between the OPC's patches must be treated specially to form a surface which can be rendered watertight. This can be complicated when allowing patch grid sizes to differ totally arbitrarily within one model. To avoid these difficulties we considered aligning the patch raster sizes of our system with the ones of the file structure. The data transfer of large models from intermediate results to the *Ordered Point Cloud* for rendering and visualization can still take a long time. Thus, the most efficient way to serve that file system seems to be to directly write or read the relevant files. By taking advantage of the idea of hierarchically ordered point cloud patches in a quadratic raster, there are lower conversion costs to access the file structure's data. When taking care of the file systems integrity and saving additional information in a way that does not affect the visualization method of Ortner we find this to be the most efficient way to serve all needs stated in 1.3.

3.2 The patch concept

The concept underlying the *Versatile Surface Model* takes advantage of the idea of hierarchically ordered point cloud patches similar to the proposed idea in the visualization approach of Ortner et al. [2010]. The surface represented by one model is divided into a number of patches. These patches are hierarchically ordered and interconnected. Ortner et al. used this approach to create several overview levels of the surface which are also divided into patches. In their work they introduced a pyramid of overview levels above the actual surface providing representations of the surface at lower resolutions. In this work the idea of hierarchic surface patch levels is extended to several highly detailed sub-pyramids instead of one overview pyramid of levels of detail. It is used to enable the model to manage locally differing grades of detail or resolutions within the surface by suggesting these sub-pyramids reside under the actual surface to provide representations of distinct regions of the surface in higher significant resolutions. We will explain that further in Section 3.2.2.

The respective part of the surface data within a patch is represented by a 3D point cloud which is rasterized into a quadratic grid. All patches of one model exhibit the same grid size. That is very supportive to create a watertight surface mesh and avoid holes at the transitions between patches. Creating overview patches for the level of detail pyramid also benefits from homogeneous patch grids. This will also be described in more detail in Section 3.2.2. The surface data within each of the patches can have its own individual resolution. The resolution of one patch is defined by the mean distances of the points within to its direct neighbours. As mentioned previously common resolutions of TLS systems are up to one point per mm^2. As another example, ALS system's resulting resolutions can be much lower, with about 3 to 4 points per m^2. Providing surface data of both sources in one model can result in large resolution differences, which is why independent surface resolutions within the patches are very useful.

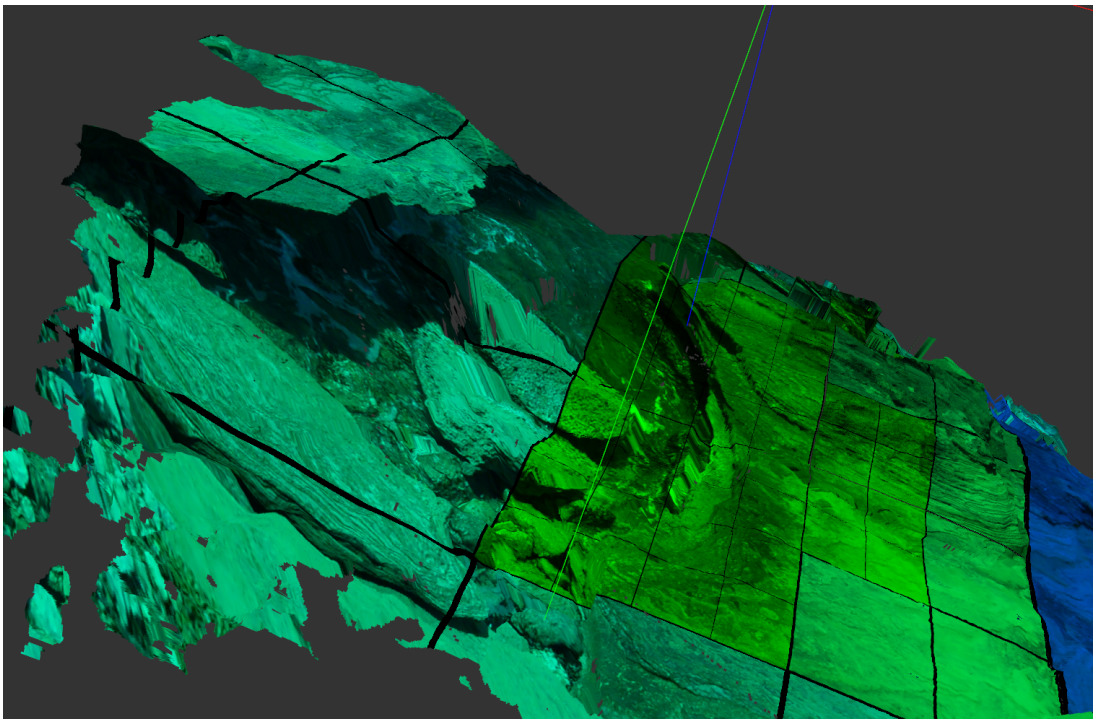


Figure 3.1: This figure shows a surface model in lod false colors divided into several quadratic patches of the same grid size but not the same resolution. The blue patches omit the lowest resolution, the light green a medium resolution and the green symbolize the patches rendered in the highest resolution. This image was produced by the rendering and visualization tool of the team of VRVis [VRVis, 2013].

Figure 3.1 shows the division of a terrain surface model into patches of different detail levels. All patches shown have the same grid size but not the same resolution. The level of detail is shown in false colors. This image - as well as all following render images of surface models - was produced by the rendering and visualization tool by the team of VRVis [VRVis, 2013]. It implements the visualization method of Ortner et al. [2010] when rendering surface data stored in the *Ordered Point Cloud* file structure.

3.2.1 Patch data

Each patch consists of 3D point cloud data which is organized in a quadratic grid and an arbitrary number of textures. Corresponding mapping information is connected to each texture. 3D points residing in the grid are ordered in columns and rows by their neighbor relations. That enables the system to efficiently serve the *Ordered Point Cloud* file structure. This neighborhood dependency is also preserved at the borders of patches throughout the whole model. Neighboring patches feature an overlap at the borders

to enable a watertight visualization as mentiond in Section 3.1.3. Every neighbor of a vertex in the grid is also a natural neighbor of the same vertex in the surface. This relation is illustrated in Figure 3.2.

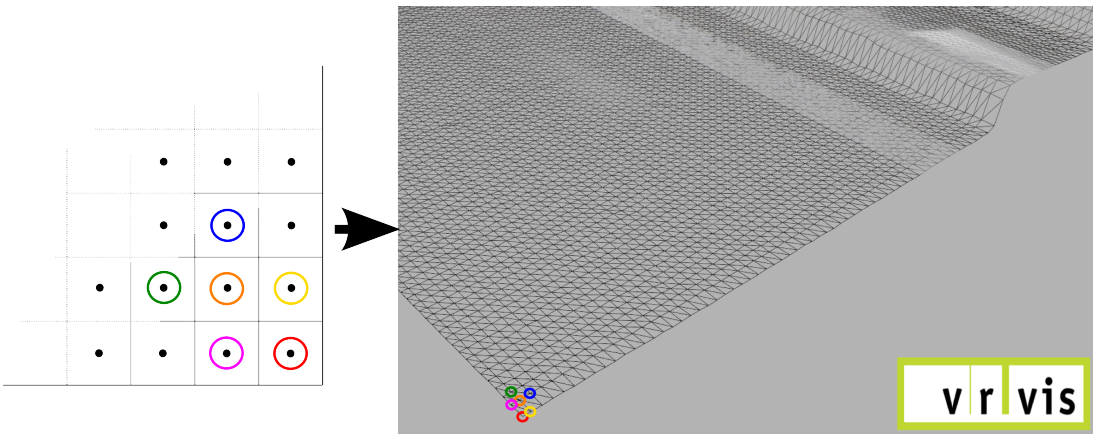


Figure 3.2: This figure illustrates that each neighbour of a certain vertex in the patch grid is also a natural neighbour of the same vertex in the mesh. On the left a border corner of a patch's raster or grid with its vertices in it is symbolized. On the right the corresponding mesh is rendered. The correlating vertices are marked in the same color. The mesh image was produced by the rendering and visualization tool of the team of VRVis [VRVis, 2013].

This condition also represents the surface very intuitively. Searching for specific areas, inserting or replacing data in the right place can be done quickly. Furthermore memory for explicit linking vertices to edges and polygons can be saved.

Each patch can be equipped with texture information. Textures are embodied by raster images of one or more channels. Basically, an arbitrary number of such textures could be possible but is impractical. To connect the texture to the 3D surface, texture mapping was implemented. The mapping information for each texture has to be stored in the model as well. It is represented as a two-layered quadratic floating point raster. Each floating point number represents a pixel coordinate in the image raster of the corresponding texture. One of the layers is responsible for holding the column coordinates and one for the row coordinates. The texture can also be seen as one portion of the overall surface texture. It is also divided into quadratic parts. The dimensions of the texture raster and the point grid do not need to be the same. On the one hand, for any visualizations a higher resolution of the textures is more important to the human eye than vertex resolution. On the other hand, recording hardware devices can exhibit different output resolutions. For example, when a tunnel is recorded by a TLS system and a camera for the texture information, the laser scanner and the camera will likely exhibit different surface densities. To pay attention to these hardware characteristics without losing information or adding additional computation effort by resampling the input data different raster sizes for the texture and the 3D data raster must be possible. Possible limitations by adopting the OPC file structure and the prospects to cope with them are described later in Section 5.2.

3.2.2 Patch hierarchy

It was described earlier that the patches of one surface are connected in neighborhood relations and are ordered in hierarchical pyramid levels. While Ortner et al. used these levels to allow overview descriptions of each surface, the novel idea in this work is to use these levels to describe locally differing surface densities or resolutions within the surface. This, along with the major differences between Ortner's work and this project, which introduce novelties in the concept, are discussed and illustrated in more detail in section 3.5.4.

To define a pyramid-like level of detail structure for the model, the patches are organized in parent-children relations. In our prototype implementation one patch can be the parent of m children where $0 <= m <= 4$. A parent contains the same surface information as its children but at half of their resolution. Every child patch has its specific position relative to its parent. It refines either the upper-left, upper-right, bottom-left or bottom-right quarter of its parent. There may be fewer than four children for one parent when the surface has no defined data in that area. When there is not at least one defined vertex in the area of a patch this patch is omitted. This is shown in Figure 3.3. This small example surface consists of 16 base patches. Five of them can be omitted because they do not contain parts of the represented surface. Furthermore one patch of the second level can be omitted as well combining only undefined base patches. These patches which are omitted are shown in red. In the top level patch these omitted patches are represented as an undefined area. Undefined data parts commonly occur at the surfaces borders.

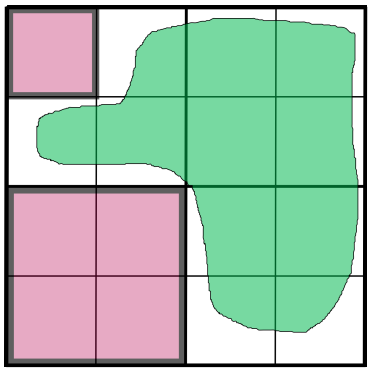


Figure 3.3: This figure illustrates the omitting of patches completely unfilled. The green area illustrates the surface within this model. The red patches can be omitted because they do not contain parts of the surface.

The top of the pyramid is defined by one patch enclosing the whole surface model as the overview level with the lowest resolution. This is the only patch in one model which does not have a parent. The bottom depth of the level of detail pyramid can vary within the surface. This is achieved by introducing sub-pyramids in the areas of higher detail. The higher the sub-pyramid or the deeper the bottom level int that area is, respectively, the higher the resolution is of which the detail data can be supplied. The patches with the locally highest resolution available for that specific part of the model define the bottom of the pyramid and do not have children. The absolute height of the pyramid is defined by the number of levels between the top patch and the bottom patches with the highest resolution in the whole model. Figure 3.4 illustrates one parent patch with three children. This is one of the surface models shown later in Chapter 4 and used for our experiments. In this example in the bottom right corner there is no surface data to completely fill the child level. Consequently, that child is omitted completely.

In Figure 3.5 a patch hierarchy is modeled in 2D view. This example again shows the result of not completely filled child levels. It also illustrates the differing pyramid depths by sub pyramids of different height at the bottom of the pyramid. These areas feature higher surface density than others. On the right side a patch of the second level can be seen missing the right part of its descendants pyramid. The highest resolution can differ locally by changing pyramid depth. The absolute pyramid height in this example is six levels. This allows very high resolution differences in one model and is the basis for merging models with different resolutions or with a high variation of the level of detail, respectively. If, for example, a close-up reconstruction of a rock with special interest of a large, already reconstructed terrain should be integrated into that model, it would only cover a small area of it. By adding only a small sub-pyramid at the bottom of that large model pyramid, inserting a completely new level can be avoided. To illustrate this concept with an example, lets assume having a 3D surface model of a crater in large scale and low resolution taken by a fly over and a close-up reconstruction of a specific rock within the crater. Each

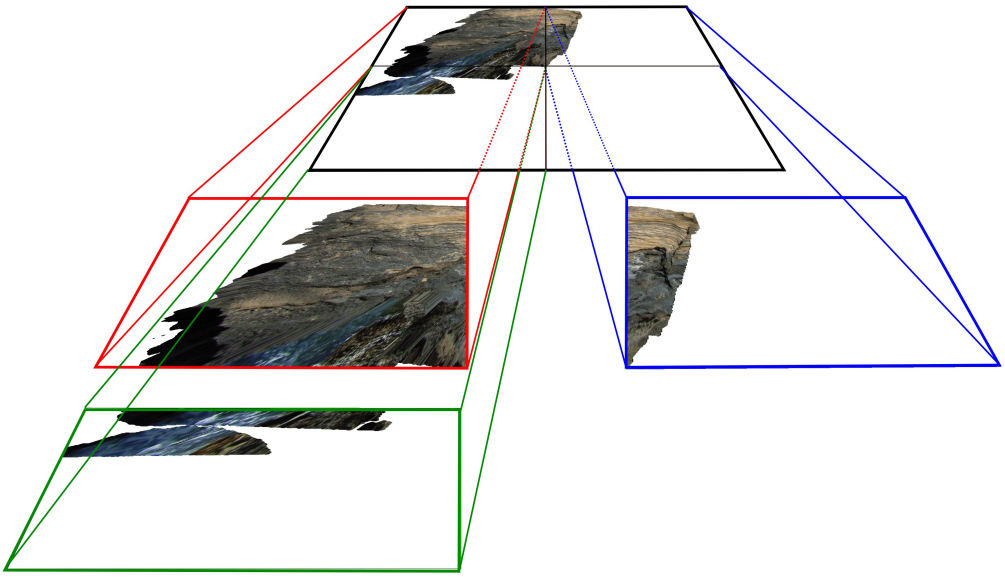


Figure 3.4: This figure shows how patches of different resolution form a level of detail pyramid. Patches are organized in a parent-children hierarchy.

model can have some overview levels providing the surface in different resolutions which are organized in pyramid layers. The rock reconstruction can be seen as sub-pyramid of the crater model, covering only a small specific area of it with significant detail information, when it is added to the crater model structure and positioned hierarchically under the bottom pyramid layer of the crater model.

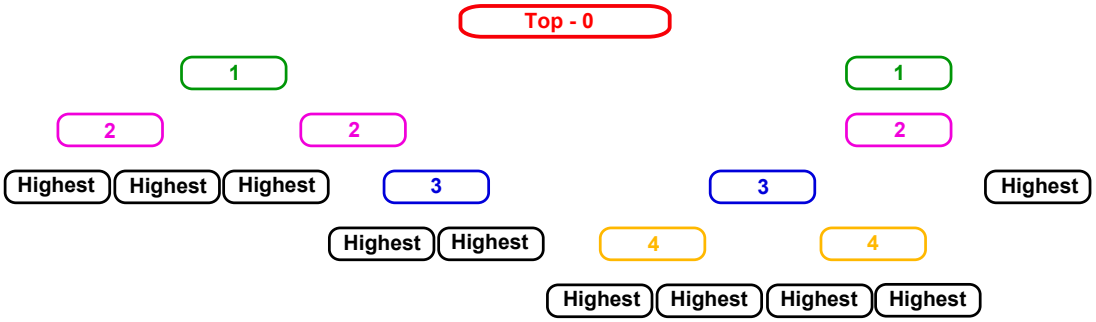
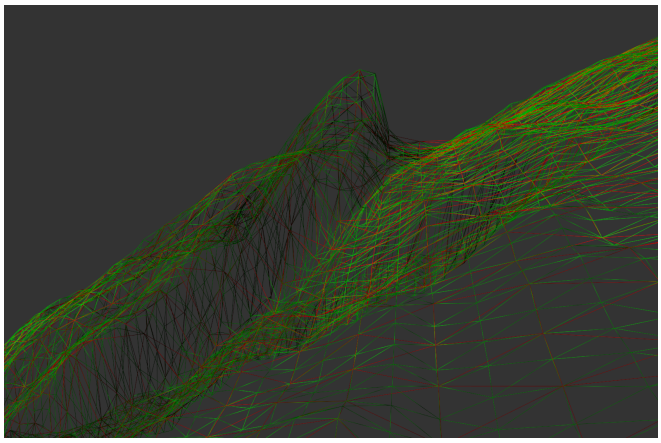


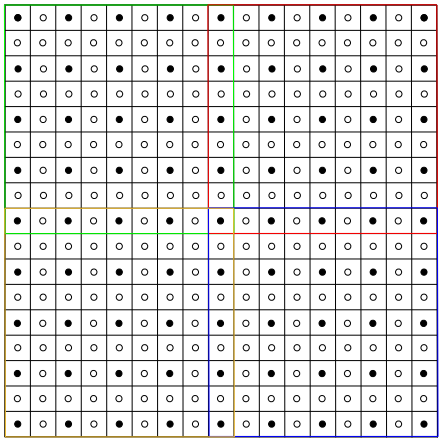
Figure 3.5: This figure illustrates a deeper and more complex patch hierarchy in 2D. This example has an absolute pyramid height of six levels and areas with lower resolution featuring heights of four or five levels.

We use quadratic patches and a quadratic number of children to be able to reduce them to one parent patch easily. We decided that the best grid resolutions for our patches are equal to $(2^n + 1)^2$ where $n \in \mathbb{N}$. That includes a one point overlapping line between each neighboring patch pair. As described previously, these overlaps avoid holes in the surface at any patch borders and support a watertight mesh. While this is necessary for the visualization, natural distance, area or volume measurements at or over the patch borders or convenient registration are also reasons for applying that overlap. In addition, it is very helpful to build a consistent level of detail pyramid. When reducing children to a parent patch, its points are decreased by the factor of 2 per side respecting the overlap. The union of the children results in $2^{n+1} + 1$ points per side. By preserving every point with an odd index in the parent we create a grid with $(2^n + 1)^2$ points again. Thus every patch regardless of its level of detail has the same number of points in it if it is fully defined. Neighboring conditions can be satisfied, the point order is obtained and any "jumping" points in the visualization of the surface can be avoided when switching between two levels.

The overlap between the patches also makes sure that the bordering points between two patches are also preserved in their parents. The reduction of high resolution patches to their parent patch is illustrated in Figure 3.6. The left image shows an overlay of the meshes formed by child and parent patches. Only every second vertex per direction is taken from the child patch. Traversing from a coarser patch to its children refines the mesh uniformly, avoiding moving vertices. The right image symbolizes the vertex reduction and preserving the one point overlap between neighboring patches and the right vertex number per patch. The color lines border the four child patches. The filled points illustrate points in children as well as in the parent.



(a) An overlay of two different levels of detail of the same surface. This image was produced by the rendering and visualization tool of the team of VRVis [VRVis, 2013].



(b) A patch raster of a whole child level and its parent patch.

Figure 3.6: Illustration of the surface relation between two levels of detail. Figure (a) shows an overlay of the different meshes formed by the vertices in parent and child patches. The parent is shown in red, the children in green. In (b) the reduction of four children into one parent is shown. The filled points symbolize points present in the parent and its children as well.

The pyramid representation of the surface enables different resolutions of one and the same part of the surface for efficient data processing. Furthermore, different areas of one surface model may have a completely unequal pyramid height deviating from the absolute pyramid height, as depicted in Figure 3.5. That means one part of the surface can support a significantly higher resolution than another by simply having more ancestors providing the highly detailed data.

Additionally, through the patch hierarchy every neighbor relation between the patches on the same level is clearly defined.

3.3 Surface merging

The extension of existing surfaces by highly detailed surface parts was one core requirement of our project as described in 1.3. Surface models representing terrains or objects of larger scale may be completed by close-up records of parts of the terrain or the object made by a different sensor or different settings. In Section 3.2.2, we presented an overview of the structure features supporting multi-resolution models. This section describes the inserting and merging procedure of two surface models. These models describe parts of the same object. While the model to be inserted represents a detail aperture of the object, the other model represents a large scale view of it. In Chapter 4 we will show our experimental results using that merging procedure to show the computing capabilities of this work.

We assume the models have been transformed into the same global coordinate system before the

merging process. In case the models were calculated from DEM-represented models, we expect both of them to have been projected to planes with the same inclination and orientation before.

The surfaces to be merged are both represented by a set of patches. To insert surface data from one model into another the correct combinations of source and target patches must be found. Each patch in the detail model will contribute surface data to one or more patches in the large scale model depending on the patch overlaps. So before being able to insert any data, the patches' bounding boxes must be matches to each other to find the sources from and the targets to insert.

3.3.1 Patch matching

Let all bottom patches of the detail model be the set D and all bottom patches of the target model be the set T. First we have to find the patches in T which assimilate the new data from D. To simplify the problem we make two further assumptions. Because we intend to insert a highly detailed model into an existing large model with lower detail we assume

- the detail model to have a smaller elongation than the model where to insert
- and the detail models surface to be completely surrounded by the other.

Therefore we have to compare the bounding boxes of the patches of T with the bounding box of the detail model. That results in the set A where $A \subseteq T$ and for each patch $a \in A$ the intersection of its bounding box with the bounding box of the detail model is not $\emptyset$. The next step is the comparison of the bounding boxes of all patches of A with the bounding boxes of the bottom patches of D. This results in all patches of the source model as a subset $B \subseteq D$ linked to the patches in the subset A of the target model which will receive the data, which means $\forall a \in A \; \exists \; b \in B \mid$ *the intersection of the bounding boxes of* a *and* b *is not* $\emptyset$. The patch matching process is also shown in Algorithm 1 in Chapter A.

The new division of the patches of detail model by the patches of the target model is illustrated in Figure 3.7. In this example the model contoured by the green line is assimilated by the model above it. The bounding boxes of the patches are compared and it is determined which part of the green patches represents new children of a related patch of the upper model. In the example, the parts of the left bottom patch and the left upper patch of the small model on the left side of the red border line would be resampled into new descendants of the left upper patch of the large model. The red border line in the small model represents the patch border of the large scale model. The right parts of these source patches and the remaining patches which are not segmented would be part of the descendants of the right upper patch of the large model.

3.3.2 Surface insertion

The next step is to determine the resolution of the level where the new data will be inserted and interpolated. As already described in Section 3.2.2, the resolution of a child is double that of the parent. To preserve this condition, a number of new child levels must be added at the matching position in the model to be able to insert the new surface data. The resolution of the detail model will most likely not correlate with a number equal to a power of the basis of two of the resolution of the target model. That means the new surface must be interpolated to the next number satisfying that condition.

Let ra be the resolution of the patch a and rb the resolution of the corresponding patches in B. This resolution shall be given in the mean distances between all neighboring vertices. Then the resolution difference dr between them is ra/rb. To be able to insert the highly detailed data for the patch a, new child levels at the correct resolution must be created. To do so we add n child levels to each patch $a \in A$ where $2^n >= dr$ and $n \in \mathbb{N}$. The new surface is resampled in the bottom level of these inserted child levels. Thus, the new resolution the new surface data will be resampled to is $ra/2^n$. This is shown in 3.8.

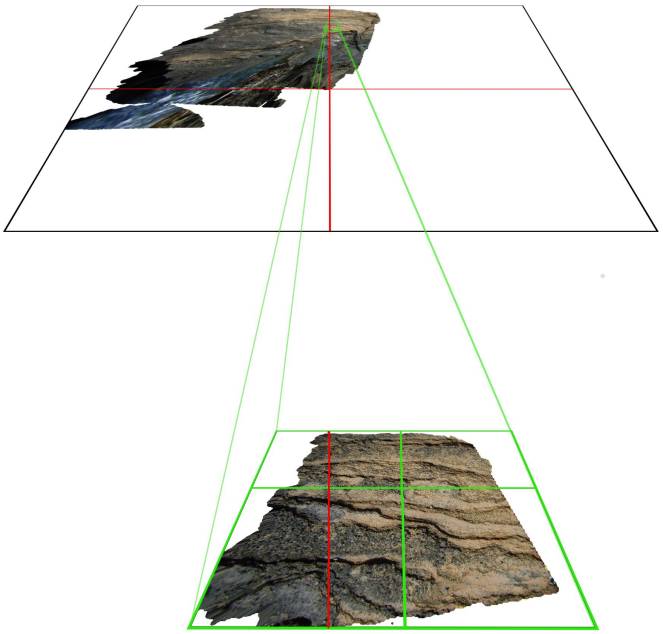


Figure 3.7: This figure symbolizes the break up of one model to be inserted into another during the patch matching. The green contoured model represents a model of high detail which is inserted into the upper, black contoured model. The division into patches of each model is shown by the red lines in the upper and the green lines in the bottom model. The bottom model is divided into two parts symbolized by the red line to assign the data portions to the patches of the upper model.

In this example the resolution difference between the new surface and the former bottom patches must be $2 < dr <= 4$. Because of that two new levels are inserted and the surface is resampled into the second which is illustrated in blue. Assuming we have transformed both model parts into the same coordinate system before, we can match the surface data extracted from the detail model into the new grid.

Next the vertices have to be resampled to their new grid positions in the new raster added along with the new levels. Both models are organized in an ordered grid and we already assumed the new points were part of the existing surface and with a more detailed resolution. So when describing the same surface, the detailed vertices can be seen as residing somewhere between the points in the target surface as well as in its grid. That means we have to find the position and orientation of the surfaces to each other in their grids to insert both into a new grid preserving the order of the vertices.

By having transformed the detail surface into the same coordinate system like the target surface we approached the matching process by matching the points with the extrema values. This means the vertices contain either the x, y or z coordinate positive or negative extrema value. It is very likely that these points with extrema values in the coordinates lie on the borders of the detailed surface but at least they tend to lie far away from each other in relation to the total elongation of the surfaces portion. Therefore matching these points describes the orientation of the surfaces to each other. We use the positions and distances of these points to each other to determine the parameters for resampling the detail surface points into the new grid. The vertices resampled into the raster positions between these matched vertex positions are linearly interpolated by taking four neighbors to the interpolated position in the original grid. The left, right, upper and bottom neighbor to the newly interpolated position are taken for interpolation. Their values are weighted by the distance to the interpolated position and summed up. If any of these vertices are undefined the sum is re-weighted with the distances of the remaining valid vertices to avoid invalid results due to multiplications with too low weight sums. The weighted sum of these vertices results in

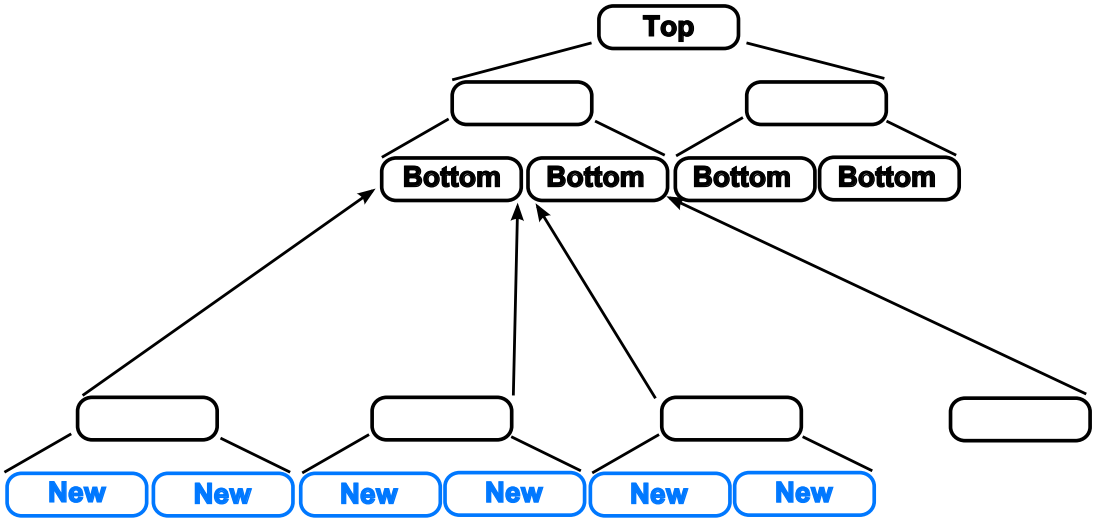


Figure 3.8: This figure illustrates the merging process in 2D. The new data is inserted at the bottom of a new sub-pyramid.

the new vertex.

By using this method processing time could be saved. Unfortunately this premises both models being recorded with the same sensor geometry. As already mentioned in Section 3.1 different sensor geometrics should be supported to allow maximum variance and freedom in modeling and processing. In Figure 3.9 surface data from different sensor geometries is symbolized. The black points represent vertices from a large scale model while the red points represent vertices from a smaller model with higher detail to be inserted to the other. To fit the new data into the other model, a number of patch levels resulting in the most suitable resolution for the detail data would be added and the raster positions for the new vertices within the patches' grids would be interpolated. Using a linear interpolation would deform the shape when inserting the red vertices into the same system as the large surface. If only a few points were matched and the raster positions of the rest were linearly interpolated, the vertices of the more dense area would be moved towards the less dense area in the raster. The inserted red points would not reside next to the black vertices which are their actual neighbor in the surface shape. Using linear interpolation it is not possible to merge such models into to same raster. For this reason, our prototype does not support the merging of surfaces of different sensor geometry at the moment. We are, however, considering it as part of the future work for this project to find alternative approaches.

3.3.3 Surface interpolation and reduction

The patches containing the new resampled surface data now represent the new bottom of a sub pyramid inserted at the location of the set A. That means these patches contain the actual surface information for that location in the model and not just an overview level. Every patch positioned higher in the hierarchy represents an overview level of this new surface part. To re-establish a consistent model the patches of the levels inserted between the new bottom and all patches $a \in A$ which are now overview levels must contain a reduced version of the surface as described in Section 3.2.2 as well as all $a \in A$ and their ancestors. This means this approach of inserting a detail model of a large scale terrain into an existing model is not registration and merging, but rather replacement. The part of the surface in all patches $a \in A$ covered by the new surface data is replaced by a low resolution version of it.

It is also very likely the new resampled surface does not completely fill the newly inserted bottom patches. The new surface was considered to be completely surrounded by the surface of the old model. So there is at least coarser surface data available to fill those patches. To gain a consistent level of detail pyramid these areas are filled with data from the respective patches in A. Because the overview

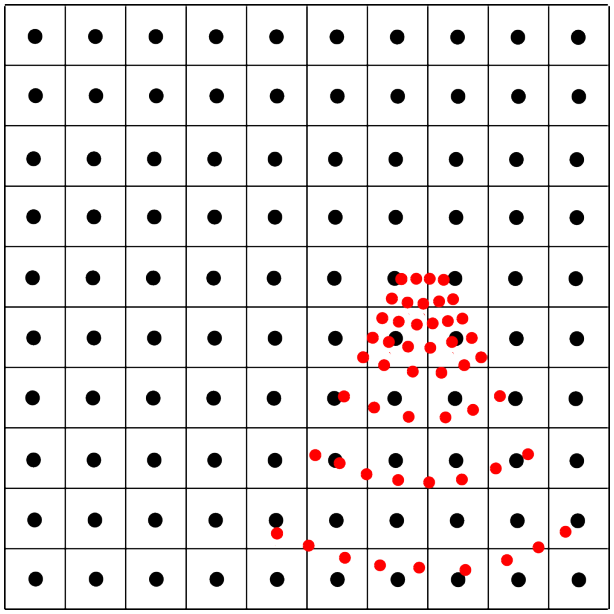


Figure 3.9: This figure illustrates differences in the geometric system of the recording sensor. The black points represent the vertices of a model recorded with one and the red points the vertices of a model recorded with a different different sensor. While the red ones symbolize a significantly more detailed aperture of the surface the black ones show a part of the object of large scale. Merging the detailed surface into the other would lead to a change in the neighbor relations and therefore to a deformation of the resulting mesh.

levels have a significantly coarser version of the surface it must be interpolated. The data is taken from original bottom level of the patches in A and interpolated to double the vertices each level downwards. Without this mechanism parts of the model describing the same part of the object or terrain would exhibit different surface data and therefore represent completely different shapes. The areas where no data is available in the detail model would be filled with undefined values in the new levels while the surface itself is defined and recorded at that location. This procedure is shown in Figure 3.10. The patches are underlined with colored lines which represent the surface data within the patches. The green part represents the new surface data which was inserted to the new added patches. The red lines indicate data from the original model which the new data is inserted into. As shown the new data does not completely fill the new patches at the bottom. While the new surface data is transferred to the original bottom patches and the overview top patch in reduced form the empty spaces in the new patches have to be filled with interpolated data from above to create a consistent model.

To create a model which consistently represents a terrain or object, all hierarchy levels must describe the same shape where defined data exists. Because of that, a reduction of the new model data into the overview levels as well as interpolation of the undefined areas in the new bottom level is mandatory.

3.3.4 Registration

To successfully merge one surface model into another, both must have been transformed into the same coordinate system. The merging mechanism described in Section 3.3 presumes this condition to be satisfied. The process of matching 3D surfaces and transforming them into the same coordinate system is called registration. Registering 3D surfaces is outlined and an overview of several existing registration algorithms was summarized in Section 2.4. The prototype of this work implements an *ICP* based approach. In the future, different concepts have to be evaluated. The test data used in this work had a good pre-alignment and the *ICP* based approach was sufficient for a general analysis on the effects of

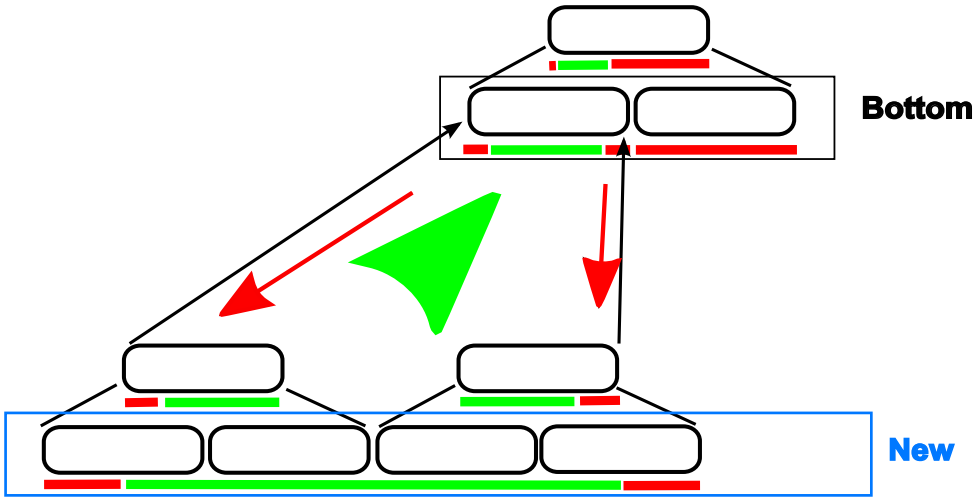


Figure 3.10: This figure illustrates which parts of the merged surfaces are interpolated or reduced to a lower resolution. The green line under the new inserted patches represents the new surface data which has been rastered into these patches. It is inserted into the overview patches in reduced form while the empty parts of the new patches are filled with interpolated data from the original bottom patches.

co-registration on surfaces represented by this works special data structure.

In the case of this work, the surfaces to be registered to each other are both represented by a set of patches, each containing only a part of the surface. To keep the surface consistent, each part must be transformed in the same way. Performing the registration process for each patch of one surface separately with each of corresponding patches of the other surface could lead to several problems. First, one patch in one surface can have more than one corresponding patch in the other surface. So the registering process could be performed several times, every time applying a slightly different transformation. This means wasting computation time with no earnings. Second, registering the patches separately may result in differing transformations for each patch. That could result in a surface which was formerly closed and watertight drifting apart and creating holes in between if the registration of one patch moves it in a different direction than its neighbor.

So the surface to insert must be transformed as a whole. Putting together all bottom patches into one surface before starting the registering process may result in a very large surface structure requiring a huge amount of memory. Furthermore the registration process may take very long. At that point the process can take advantage of the level of detail pyramid of the data structure. Applying an *ICP* registration algorithm to coarser versions of the surfaces will result in a slightly coarser but correct transformation. This transformation can then be applied to the surface data which shall be inserted into the other model. This way, each patch is transformed in the same way along with all others avoiding holes. Additionally unnecessary computation time is omitted. In one case computing a transformation for one and the same patch multiple times because it has several neighbors can be avoided. In the other case the registration of point clouds with a high resolution or a very large number of vertices is omitted. The prototype implementation is illustrated in Algorithm 2 in Chapter A.

3.3.5 Summary

The proposed method for merging two surface model into one joined model consists of four steps of which one is optional. The surface representations of this approach the surface is divided into several units, called patches. Because of that, the first step for a complete joined model is to determine the concerned patches of the models involved in the merging process. The patches of both models are matched to each other resulting in connected sets of source and target patches. After each target patch is

connected to possible source patches of the other model both surfaces can be co-registered optionally as the second step. A transformation for the whole source model is calculated and applied to each source patches' surface part before being integrated in new patches added to the bottom of the target patches. The integration into new patches of the target model is the third step. The vertices must be matched and resampled into the new raster. The last step is the surface interpolation and reduction into the other layers of the hierarchy pyramid. Empty areas of the new patches are completed with interpolated data from the original target patches and every layer above the new patches is filled with a reduced overview version of the new surface. After that the new joined model is consistent and the process is finished.

3.4 Memory management

When working with very large models or with more than one model at the same time the amount of data can exceed the memory capacity of common workstations. These models can not be stored in the memory as a whole. But for many applications this is not necessary. Often only a part of a surface model is needed at the same time. For instance, when rendering a large model and displaying it to the user, only the part in the current viewing frustum is needed. When performing a filter operation on a model, normally, also only a specific portion is needed at one point in time. As another example merging two surfaces with different resolutions should be mentioned. This process was described in detail in Section 3.3 but it was indicated in Section 3.2.2 that merging a high detail surface model to an existing large scale terrain model will likely affect only a small part of the terrain model. To cope with this challenge, a surface is divided into patches as described previously to be able to handle a model out-of-core. Only requested patches reside in the main memory. The data of the patches like 3D coordinates or textures are only loaded into the structure on request.

To provide dynamic read and write access to these patches, the patch hierarchy has to be present in the memory. That means for each existing patch of the model, an object supplies the system with information about the related data in the file structure and the parent-child relations of the patch. These must be known to navigate through the model and in case a patch must be updated to its descendant's changed data as well. Each patch object is marked with a certain flag indicating its current state. This mechanism is also used when a patch is changed and all its ancestors must be updated before further use. The ancestors of a changed patch are notified and marked to be in need of a surface update. In Figure 3.11 the different states of a patch are illustrated. In this example, new detail data was added to the bottom right patch which was updated to its new children's data afterwards. The right child of the top patch is still needs to be updated before the top patch itself can be updated. After doing so, it can be used as an overview level again containing the correct data.

The total amount of loaded data is monitored for all models currently processed. For each model allocated in the process, the patches which are already loaded into main memory are known as well as these patches' conditions. When memory is requested for loading an existing patch from the hard disk or for creating a new patch and a certain threshold is exceeded patch data needs to be removed from memory. At the time of this writing a *first-in-first-out* system is used in the prototype implementation. We also pay attention to which patches need writing operations and which do not before these can be removed from memory. Changed data is written to the hard disk's file system as described in Section 3.5, while unchanged data can be discarded. So read-only patches are freed first before changed data has to be written before its memory can be freed. The memory management of the patches is shown in Figure 3.12. It illustrates the above-described functionality with the help of a sample process. In that example, new detail data is added at the bottom of a model. New patches are created and their parents are updated with the new data inserted. The figure illustrates the swapping and flagging process. We consider it to be future work to evaluate different algorithms to find the most efficient algorithm for swapping.

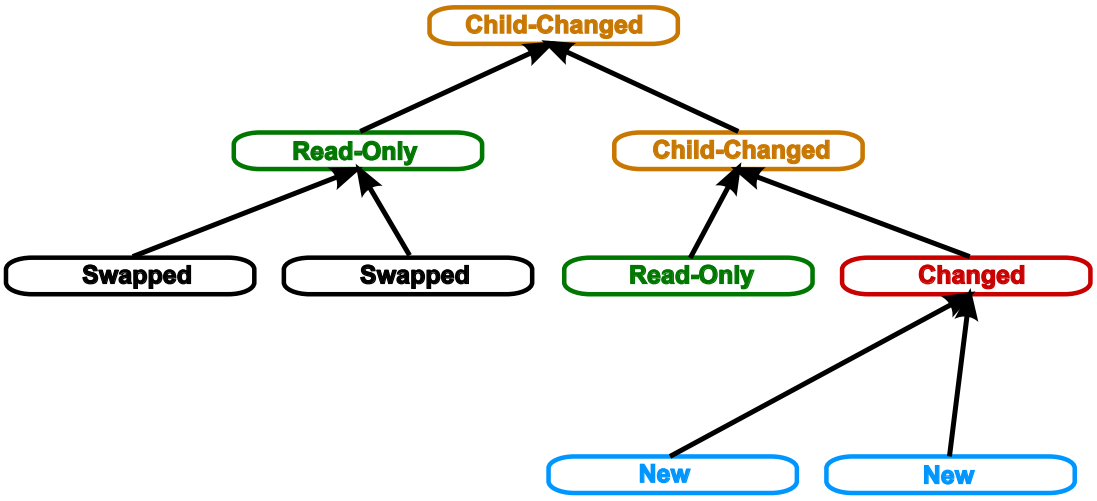


Figure 3.11: This figure illustrates the data flow in the patch hierarchy with the help of the state of each patch. When a patch is changed, its ancestors' states indicate that they must be updated for further use. The right bottom patch was changed by being updated to its new added detail children's data.

3.5 File system

Ortner et al. [2010] proposed an *out-of-core* (OOC) visualization system for huge 3D surface models. Its system is implemented in the rendering and visualization tool of VRVis we used for the presentation of our results and experiments. The OOC data management of this work is based on a so-called *Ordered Point Cloud* (OPC) file structure which stores the surface data to be rendered in smaller portions of surface data. This file system was described in detail in a technical report written by Ortner and Tobler [Ortner and Tobler, 2010]. The following sections will summarize the details of the file system, the mechanism to access and manipulate the data and any extensions made to the original structure.

3.5.1 The OPC file structure

When a 3D surface is stored in the OPC file format it is split into patches which represent a quadratic section of the surface data which is rastered into a fixed size grid. This applies to both 3D data and texture data.

The 3D data is stored as a linear vertex array which represents a rectangular aperture of a maximum quadratic section of the surface this patch contains. It can be supplied with an index array to link the vertices to edges of certain polygon types if the vertices should not be implicitly linked together in quads by their neighborhood relations. For each texture image which should be applied to that surface part contained in the patch, an array of texture coordinates is stored within the patch. It is possible that the arrays contain only a subset of the possible vertices and texture coordinates representing a rectangular section of the quadratic patch. The visualisation system will render and display the 3D and texture data within a patch regardless of its size.

All these patches are linked together in a hierarchy describing several detail levels of the whole surface. A logic structure of an OPC file system is established by using sub-directories. The directory of an OPC file structure contains a sub-directory for the images representing the texture parts and a sub-directory for the patches. Each patch has its own sub-directory within the patch's directory named by a unique identifier which clearly labels the patch. This directory contains all necessary patch data excluding the raster image files storing the textures. Because one texture can be linked to more than one patch these are stored separately and can be identified by their unique file names within the image directory. The hierarchical links between the patches are achieved by an *Extensible Markup Language*

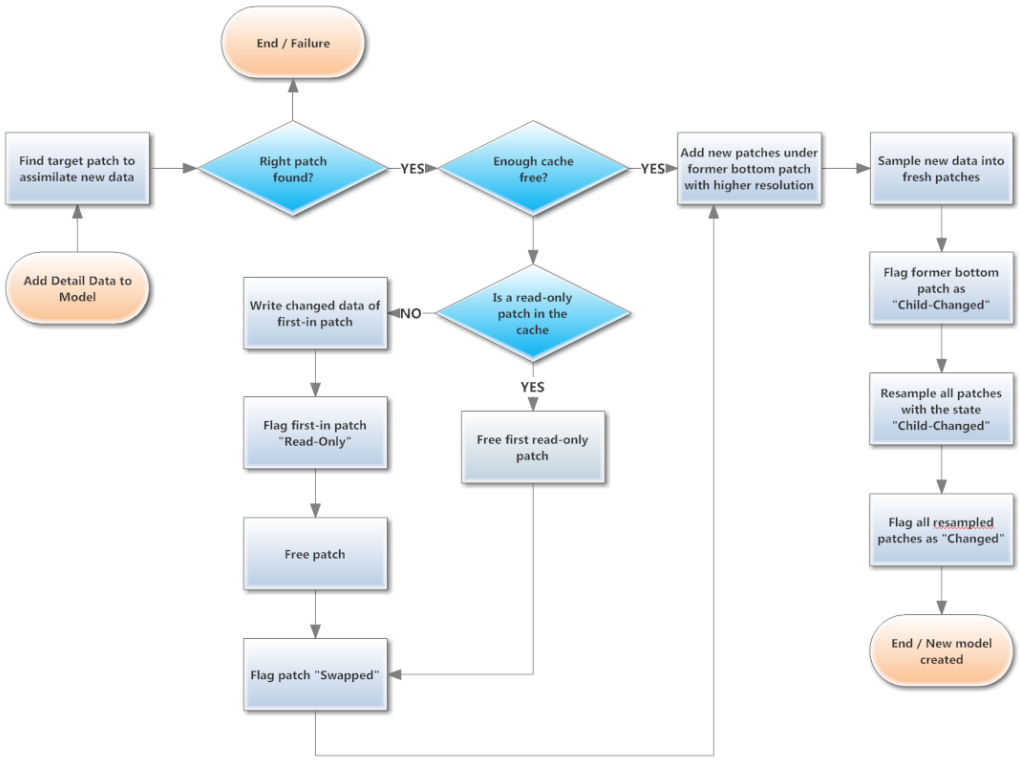


Figure 3.12: This figure illustrates the memory management in a sample process.

(XML) file containing the identifier of the root patch, the top pyramid patch, and all parent-child relations between all existing patches in the model. Patches which are not listed within this file can not be processed by the means Ortner's visualization method. This hierarchy definition file is stored in the patches' sub-directory. The whole directory and file hierarchy of an OPC file system is depicted in Figure 3.13. The brown rectangles show the directories while the black rectangles symbolize the files within these directories.

While the patch hierarchy XML file links together the single patches to a hierarchy of surfaces each patch is also supplied with an XML file within its own sub directory which links several information parts together to the patch itself. Both the patch hierarchy file and the patch XML files are all named uniformly for easy read access. All the relevant information for that specific patch is associated within this file. This means the names of the related files which includes the optional index array file, the vertex array file, the texture files and the corresponding texture coordinate array to map these to the vertices. Additionally it is supplied with the bounding box information for the rendering and visualization tool. Depending on this information it is decided the data of which patch is rendered or not. Patches of the detail level to process which have a bounding box intersecting the viewing frustum are rendered and the others not. Therefore, this information has to be maintained and valid through processing. An individual transformation matrix for each patch is also supported. So a patch can be transformed into its own individual local coordinate system. All array files are stored in the so called *Aardvark Array File* format. This is a generic array file format supporting several data types from simple primitives to a variation of two, three or four dimensional vectors. These files are stored in binary format. While plain text files have the advantage of direct manipulation and verification possibilities saving the data in binary format saves important hard disk space.

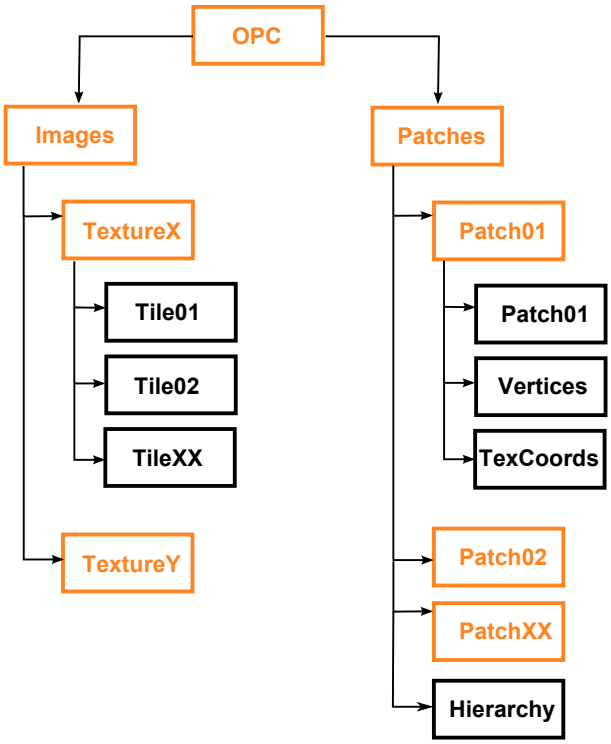


Figure 3.13: This figure illustrates the file and directory hierarchy of an OPC file system. The brown rectangles symbolize directories while the black ones show files.

3.5.2 File access and manipulation

When using this file structure for this work, every part of our model can be loaded and stored independently. But the model's consistency must be maintained for every patch being accessed. At the beginning of this project we were able to profit from the experience of the Joanneum Research team[Research, 2013]. They used this format for visualizing different processing results and were able to export other formats to an OPC file structure. We could take advantage of their writing procedures for the file structure. That included the linear indexing of the array files for a correct representation of a rectangular section of the surface which can be interpreted by the VRVis rendering tool. The writing procedure of the patch and the patch hierarchy XML files and the correct build up of the directory structure when creating a fresh OPC file system could also be used based on Joanneum's experience. Because their procedures were used for visualization export only, any read procedures which includes maintenance of the models consistency when manipulating parts of it were not sufficiently developed.

Either when creating a new OPC file structure or writing a manipulated piece of data to an existing OPC file system the main directory structure must be established in advance. When creating a new OPC structure every initially created patch can be stored independently as long as the directory hierarchy is maintained. But every patch must be integrated into a patch hierarchy which can be saved to the hierarchy XML file at the end of the processing to form a consistent file system. When working on an existing OPC the hierarchy file is also updated at the end of the processing chain if necessary so as to avoid repeated access but maintained through the whole process in the memory. Patch files are only written in total to avoid inconsistent patch files. This means the responsible mechanism writes all files relevant for that specific patch. None of these are manipulated alone. Surface vertex coordinates and texture coordinates for the mapping are transferred to *Aardvark Array* structures which are in control of the writing of the array file correctly indexed. The correct file names have to be stored in the XML patch file to link the patch structure together. The patch identifier must correlate with the directory name of the patch and is stored in the hierarchy structure which is written after the processing. By repeating this mechanism

every patch can be stored independently to the file structure.

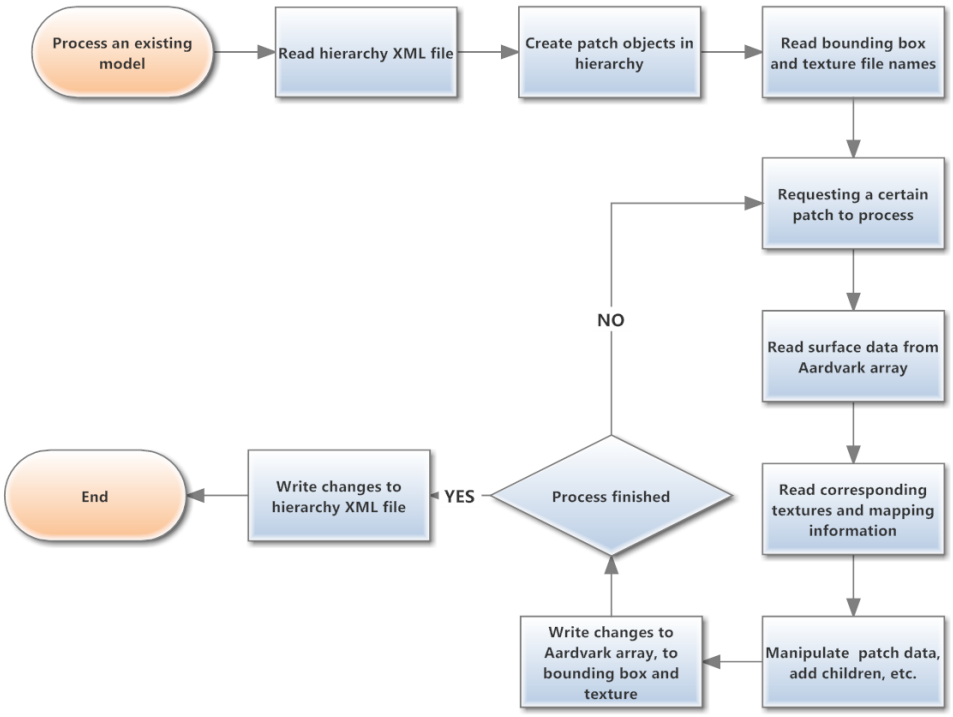


Figure 3.14: This flowchart illustrates the process of file access and manipulation when processing one model.

While a one-way data transfer to an OPC file structure is sufficient to enable the usage of Ortner's visualization mechanism to realize the concept of this work both read and write access to the OPC file structure must be supported. On the one hand it was the intention to manipulate existing OPC based surface models. On the other hand it was part of the concept to use the file system as dynamic out-of-core file basis for all manipulations on surfaces in this work. Because of these reasons, read procedures had to be implemented. The information about the patch directories is saved within the hierarchy file. Therefore it has to be read first before getting access to any patch or surface data. Further file information can be read from the patch XML file within the patch directory afterwards. To be able to match each patch to a specific section of the surface model the bounding box is also read from the XML file and stored in the corresponding patch structure. Throughout the processing patch vertex data or texture data can be read and swapped on demand as long as the file paths are maintained. The texture data can then be read from the image raster files and the vertex and texture mapping information from the *Aardvark Array* files to the corresponding patch structure in the memory for each patch independently. Manipulating an existing OPC based model is illustrated in the flowchart in Figure 3.14.

3.5.3 Handling undefined regions

For more efficient handling of larger undefined regions within a patch, we extended the file system to our needs while preserving compatibility with the originally introduced structure by [Ortner and Tobler, 2010].

As already explained, a patch defines a square subsection of the OPC. For each patch a vertex array is supplied which contains a rectangular section of the maximum square subsection the patch defines. To save hard disk space in a case where the grid contains undefined regions these can be omitted if the order

of the defined points in the grid can remain unchanged and the neighborhood relations are preserved when indexing the vertices. Therefore completely undefined columns or rows at the outer borders of the grid can be left out. The rendering and visualization method of Ortner is not affected if the patch contains only a subsection of its square part or if it is completely filled. Originally designed as an Out-Of-Core rendering file system it supplied the renderer with portions of 3D data regarding its position to the camera. The level of detail and its position in 3D space to the camera determined if one piece was rendered or not. The position to the camera could be easily calculated by using bounding box information for each patch. Information about the relation of the grid of each patch to the others was not necessary. For further processing of existing models, however, especially at the surfaces borders we needed to add a 2D bounding box information if the vertices had not been stored in the original squared grid.

We decided to take advantage of that possibility in the visualization process and the data structure. Before storing a patch to the OPC file system the first and last columns and rows containing a defined and valid vertex are determined. These undefined areas are omitted when storing the grid only as a rectangular sub grid of the original quadratic grid. To be able to locate the correct position of the vertices within the original square section of the patch the omitted columns and rows are stored as front and back offset into the patch XML file as an extension. Again, the visualization process is not affected by additional information within the XML file as long as the mandatory tags are consistent.

Hence, the patches can be restored into the system correctly and hard disk space can be saved. When reading the patch files the offsets can be applied to the point cloud and the vertices reside in their original positions in the squared grid. This will happen especially at the surface's borders. It is most likely that the surface does not contain exact that number of vertices to completely fill the border patch's grid. It also may happen that in inner regions of the surface information is missing and large holes may appear where the sensor did not record sufficient surface information. These holes can also result in partly filled patches.

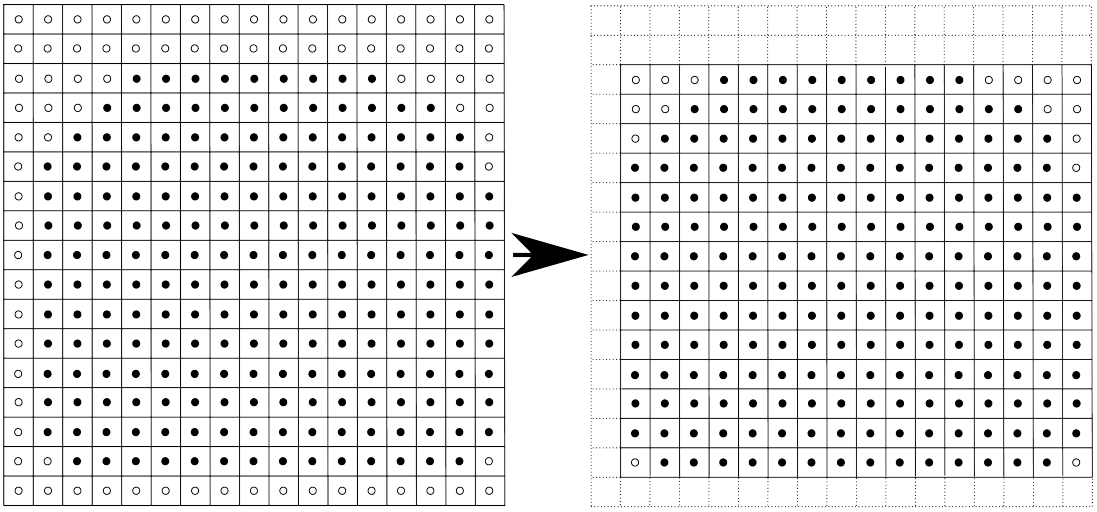


Figure 3.15: This figure illustrates how whole defined columns or rows in the point grid are omitted when saving the point array of a patch to save hard disk space. On the left side the working patch is shown while the left raster illustrates the saved array. Unfilled points illustrate undefined vertices.

In Figure 3.15 this mechanism is illustrated. The filled points represent defined vertices in the point grid, the unfilled undefined vertices. Whole rows and columns with undefined vertices are omitted when saving a point array of a patch. The working grid is shown on the left while the right raster represents the saved array.

Like for the 3D vertex grid for the textures offset information must be saved as well. Each texture tile is also a quadratic section of the overall texture image supplied for that OPC surface. The position of

partially defined texture parts in the virtual full grid must be available for further processing if undefined pixel columns and rows are not saved as well. While any patch is supplied with an XML file linking all files together and containing necessary meta data like bounding boxes or matrices, an image tile does not have any additional XML files in the original file structure definition. To be able to store the necessary information we extended the file structure with such an XML file which is stored along with the image tile. This contains the offsets of the image tile in a quadratic section which is the maximum coverage of that tile within the full texture image.

3.5.4 Comparision of this thesis to the OPC system

In the previous sections, the OPC file system was described. It was outlined how the OPC file structure is used and how this work could benefit from it. The very close connection to the OPC system and concept as well makes it difficult to draw a line between the scope of it and the scope of this work. This section shall confine the OPC concept and system from this work and outline the main differences which were partly described in earlier sections.

The *Ordered Point Cloud* (OPC) file system was designed to serve as a file basis for Ortners visualization method. The surface data for this out-of-core visualization method for virtually infinite 3D surfaces was in need of an appropriate data management. Therefore, storage and out-of-core data management was the scope of the OPC file system. Data manipulation and processing were not part of it. This work, on the contrary was developed for the manipulation of large, multi-resolution 3D surface structures. It should enable automated or partly automated processing of 3D surface representation models. The use of Ortner's visualization method for viewing and processed results was mandatory due to corporate project collaboration. This work was intended to serve these projects. This means Ortner's work as well as this work uses the OPC file system design as a file basis for 3D surface representation models but with different scope. This can be seen in Figure 3.16. While Ortner introduced a rendering and visualization method this work proposes a method for data manipulation and processing.

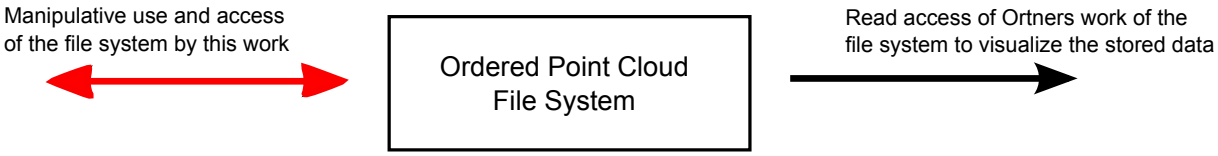


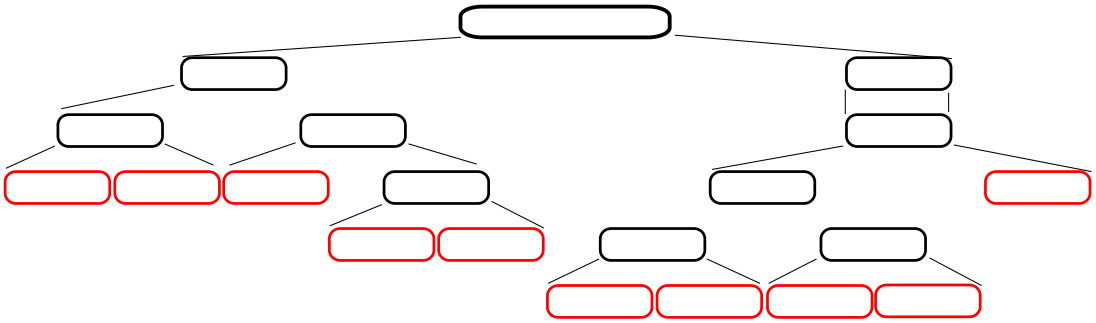
Figure 3.16: This figure demonstrates the data flow and the use of the OPC file system within the different works. The arrows symbolize the data flow.

Setting up a 3D surface representation as a set of hierarchically ordered patches was also used in the OPC structure. There the hierarchy was used to introduce several overview levels of the same surface. During rendering and visualization elements of the surface farther away from the user's camera were rendered at a lower level using one of the overview representations of the surface to save rendering time. Rendered as a small object on screen the user was not aware of the resolution difference. Objects with a nearer distance to the camera were rendered at a higher resolution. This work used the concept of this hierarchy structure to represent arbitrarily detailed surfaces. By allowing locally differing numbers of hierarchy levels each part of the surface was able to differ in detail and resolution depending on the availability of surface data. Also inserting new surface data providing the model with higher detail information is in the need of introducing new hierarchy levels at the area it has to be inserted. Therefore, the hierarchy idea of the OPC structure was to provide overview levels above a certain defined actual surface level. In contrast, this work does not define an actual surface level in its hierarchy. The actual surface level differs locally. It is always the lowest level. Each insertion operation can insert newer lower levels providing more surface details at a higher resolution. The basic hierarchy idea was adopted but developed further creating a similar but different concept. This difference is shown in Figure 3.17. The left image shows the hierarchy concept of this work while the right one shows the OPC hierarchy

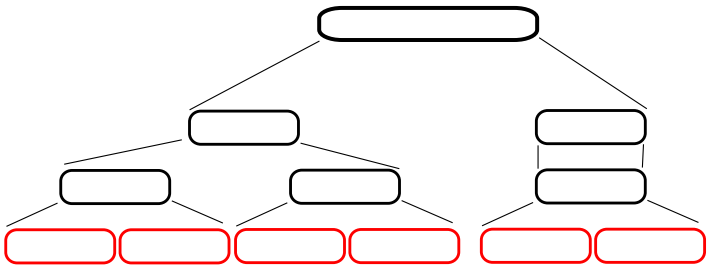
concept. The red-colored patches represent patches with the actual surface information. It is illustrated that the OPC has only one level with actual surface information and every other layer represents an overview level of the surface, while in this work every layer can contain parts of the actual surface due to the locally differing detail density of the surface. The differences between the concept of this work and the concept of Ortner's OPC as well as the contributions of this work are summarized in Table 3.1.

	OPC	This work
Hierarchy	Used for storing overview levels	Used for locally differing detail grade
Processing scope	Out-of-core visualization for human interaction	Dynamic and automated processing
Data manipulation	Not envisaged	Model manipulation and extension by inserting new detail data

Table 3.1: The table compares the methods and scopes of this work with the OPC of Ortner.



(a) The hierarchy and the position of the actual surface information of this work illustrated.



(b) The hierarchy idea of the OPC file system illustrated.

Figure 3.17: These figures illustrate the differences in the hierarchy concept of this work and the OPC file system as designed for Ortners visualization process. Figure (a) shows the concept of a patch hierarchy of this work in contrast to the OPC hierarchy concept in Figure (b). The red patches represent the patches containing the actual surface information while the black ones represent patches with overview surface levels. In this work the actual surface can be existent in various levels to represent different detail densities within the model.

Chapter 4

Experimental Results

The approach of this thesis was tested intensively using surface models created by remote sensing applications. The surface models were recorded at two different test sites. The first one was recorded in Germany the second in Mallorca. The goal was to evaluate the capability of this approach to handle large data amounts, representing arbitrary surfaces and working on multi-resolution surface representations. We addressed all these issues by calculating merges of surface models as described in Section 3.3 on different stereo reconstructions of the same terrain site. For this scenario we created several terrain reconstructions resulting in surface models ranging from large field-of-view and low level-of-detail models to very small field of view and high level-of-detail models. As test system a modern computer equipped with an Intel Core i7-3770 was used.

The merging process involves two surface models at a time. It addresses several issues regarding the features of our concept we want to prove functional. This includes

- the capability of representing multi resolution models,
- the merging to a new model with locally differing detail resolution,
- the processing of two models which can be very large,
- differing amounts of data per model which are actually used in the process

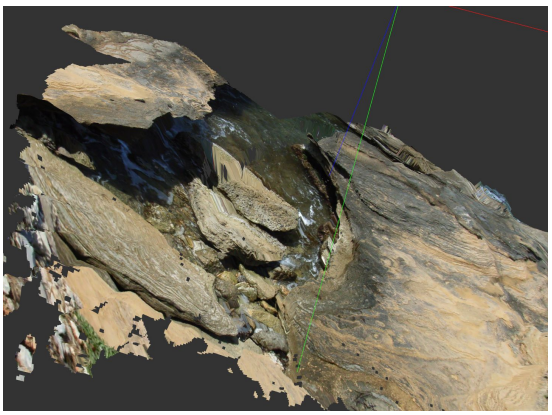
4.1 Mallorca dataset

For the first terrain merging test several models reconstructed from a test site near Betlem at Mallorca, Spain, were used. The pictures used for photogrammetric reconstruction were taken with a digital camera, model *Ricoh Caplio R2*. Figure 4.1 shows the test site and a 3D model of it which was reconstructed by using photogrammetric stereo image reconstruction. The models we used were terrain reconstructions of the same surface with different levels of detail and fields of view. They consist of 3D data and an RGB texture.

We established five different surface models of the area. In Table 4.1 the model specifications and differences are shown. Every model covers a part within the more coarse models providing more detail about that area. In Figure 4.2 *Model 00* and *Model 02* are shown to illustrate the differences from the initial models. The resolution of the more detailed model is 10 times more accurate than that of the coarse one. The area of the terrain covered by the detailed model is marked by the red rectangle in the large one to show the difference of the area size.

When merging two *Versatile Surface Models* the resulting model needs significantly more space than the sum of the two merged surfaces due to intermediate detail levels in the new pyramid. This is also

(a) The terrain site in Mallorca



(b) A reconstructed model of the testsite. This image was produced by the rendering and visualization tool of the team of VRVis [VRVis, 2013].

Figure 4.1: The terrain used for our experimental test. The terrain was recorded in Mallorca. (a) shows an overview image of the whole area. (b) shows one resulting model.

Name	total patches	detail levels	Size	Mean resolution in 3D points
Model 00	43	4	183 MB	0.1^2 per mm^2
Model 01	37	4	157 MB	0.25^2 per mm^2
Model 02	14	3	59 MB	1.0^2 per mm^2
Model 03	35	4	149 MB	2.5^2 per mm^2
Model 04	14	3	59 MB	12.5^2 per mm^2
Embedded Models 00 - 03	286	9	1215 MB	3.2^2 per mm^2
Embedded	326	11	1385 MB	12.8^2 per mm^2

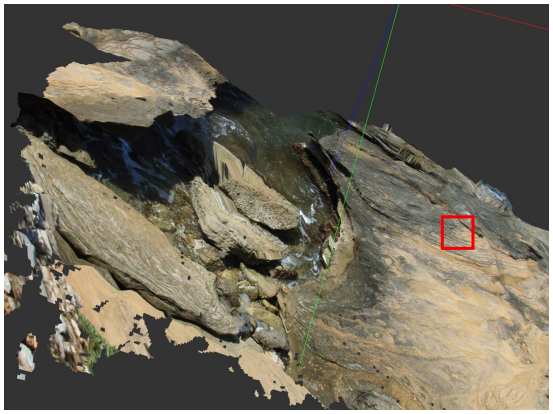
Table 4.1: The table compares the statistics of the different models used in the Mallorca experiment.

shown in the Table 4.1. The surface resulting from embedding all models into the most coarse one is illustrated in Figure 4.3. In Figures 4.3b, 4.3c and 4.3d the surface is represented as point cloud and not as mesh to emphasize the resolution difference.

Our concept allows having only the currently used part of the surface in the main memory. The parts of the surfaces not needed stay on the hard disk until they are requested. In the last iteration of the merging process, the raw data of the model widely exceeded one gigabyte but only 25% of the surface data had to be present in the main memory at all times of the iteration. Because of the relatively small size of the data and the small amount actually processed of the whole surface swapping patches to free memory was not necessary. But by the same token, this experiment shows the reduction of memory consumption by dividing a surface into interconnected portions. Due to the fact the vertices are ordered in a regular grid accessing the right part at the moment it is needed is intuitive and effective. Fitting this model into the memory as a whole could cause problems in slower and more limited computers while it is easier with our approach.

4.2 Pellheim dataset

An other terrain merging test was performed on different models reconstructed from records taken in a clay-pit in Pellheim near Dachau, Germany. The images used for reconstruction were taken during a field trial by Joanneum Research and the Technical University of Munich in late October of 2013. The camera



(a) The most coarse model of the test site with the largest field of view.

(b) A more detailed model of a smaller angle of the test site.

Figure 4.2: This figure shows two different detailed parts of the test site. The model shown in (b) shows a detailed aperture of the model in (a) where its share of the terrain is marked with the red rectangle. These images were produced by the rendering and visualization tool of the team of VRVis [VRVis, 2013].

Name	total patches	detail levels	Size	Mean resolution in 3D points
Model 02	14	4	59 MB	0.078^2 per mm^2
Model 03	16	4	68 MB	0.156^2 per mm^2
Model 05	51	4	217 MB	1.25^2 per mm^2
Model 06	58	4	246 MB	5^2 per mm^2
Model 07	15	3	64 MB	10^2 per mm^2
Embedded	352	9	1496 MB	10^2 per mm^2

Table 4.2: The table compares the statistics of the different models used in the Pellheim experiment.

used was a DSLR camera, model *Olympus E-1*. The target of the reconstruction was a rock face with a artificial target attached. Figure 4.4 gives an overview of the test site showing a wide angle picture and bird's eye view overview image indicating the position of the reconstructed rock face and of the positions of camera during the recording.

Eight image series were taken for photogrammetric reconstruction as illustrated in Figure 4.4b. Seven of those could be reconstructed into 3D surface models. For the merging test five of them could be used and were joined into a complete model. The result of this process can be seen in Figure 4.5. Like with the Mallorca dataset, each model covers a smaller part of another model providing higher detail. In Table 4.2 the model specifications and differences are shown. The artificial target was in the focus of the test. The most detailed models, providing the highest resolution, covered the area of the target only.

The different detail levels within the merged result are illustrated in Figure 4.6. The black lines illustrate the patch borders. Patches drawn in blue or bluish color are rendered in the lowest resolution while the more red the color is the higher the details of the surface part are. It can also be observed that patches of higher detail are smaller in scale than the others. Because the models with the higher details were provided for the artificial target attached to the rock face the patch pyramid has the most levels and the highest details in that area of the merged model. This illustrates the versatility of the approach. A model can provide high resolution data for regions where high detail data is available. If high details are not necessary for a specific process overview levels for those regions are available for fast processing.

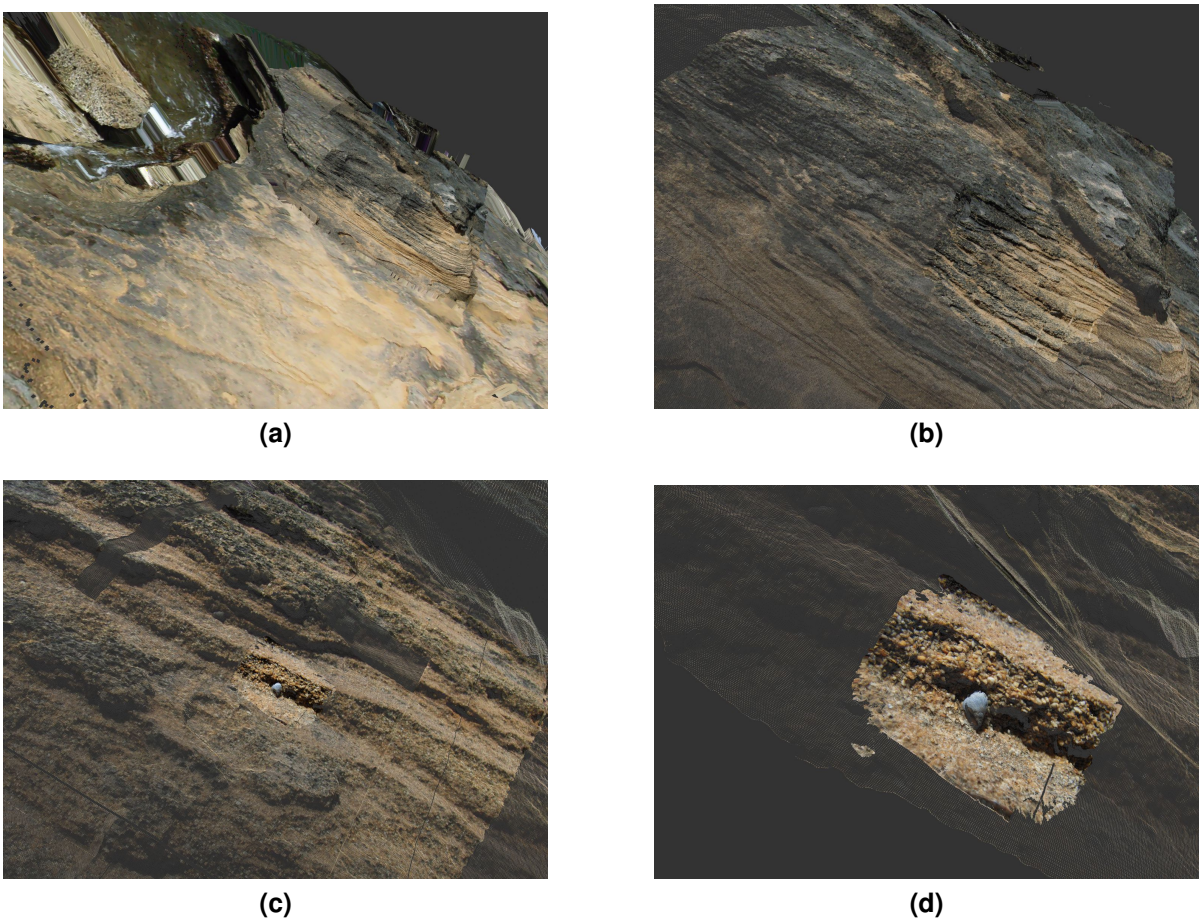
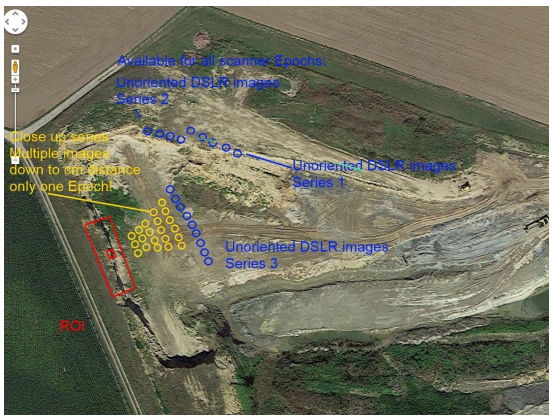


Figure 4.3: This figure shows the result of merging the models documented in Table 4.1. The subfigures (a), (b), (c) and (d) show the embedded detail model at different zoom levels. These images were produced by the rendering and visualization tool of the team of VRVis [VRVis, 2013].

4.3 Terrain registration

In Section 3.3.4 we outlined our approach of registering two models to each other before merging them together. Co-registering is an optional extension to this merging algorithm for not exactly aligned models. Most of the used models were aligned very good and did not require co-registration. To provide a versatile solution, a general idea how to implement co-registration for this merging approach was developed. It is envisaged to contribute to that field in future work. We tested the co-registration approach with models 00 and 01 of the Mallorca dataset as described in Table 4.1. This models were not aligned as good as the other Mallorca models. Without registration the smaller model does not fit correctly into the surface of the larger one. To correct this error it is registered to it. The results can be seen in Figure 4.7. The left image shows the same merging process as the right one but without registering. Therefore, the surface in the left one shows a significant offset of the inserted detail surface while that offset can not be seen in the right image. For models, which are slightly misaligned, this co-registration approach can produce better and more accurate results but needs 129% more computation time. Merging these models without registration took our test system 17 minutes while with registration it needed 39 minutes.

(a) The terrain site in Pellheim



(b) A overview of the testsite. The rock face which was reconstructed is marked in red and the camera positions are marked in yellow and blue.

Figure 4.4: The Pellheim terrain used for our experimental test. (a) shows an unprocessed image of the whole area. (b) shows an overview image of the site marking the positions of the camera and the rock face which was reconstructed from the camera images.

Models Merged	Mallorca [min:sec]	Pellheim [min:sec]
Model 01 and 02 / 02 and 03	17:43	2:54
Model 02 and 03 / 03 and 05	12:28	33:58
Model 03 and 04 / 05 and 06	19:01	20:10
Model 04 and 05 / 06 and 07	6:58	6:56

Table 4.3: The table compares the processing times of the different merging steps of the used data sets.

4.4 Experiment summary

The experiments described in this chapter were envisaged to address several issues. The capability of representing multiple resolutions of the same surface in one model was demonstrated as well as a possibility of merging two models of the same object to such a multi-resolution model. Furthermore, it was shown that this approach is able to process more than one large 3D model out-of-core. Additionally, an optional co-registration step for the merging algorithm was demonstrated. In table 4.3 the computation times of the merging steps are compared. All steps were computed without using the optional co-registration method. The average computation time for one merging process was 15:01 for an average of 30 patches per model merged into an other.

Figure 4.5: This figure shows the result of the merging test on five surface models of a rock face in Pellheim near Dachau, Germany. This image was produced by the rendering and visualization tool of the team of VRVis [VRVis, 2013].

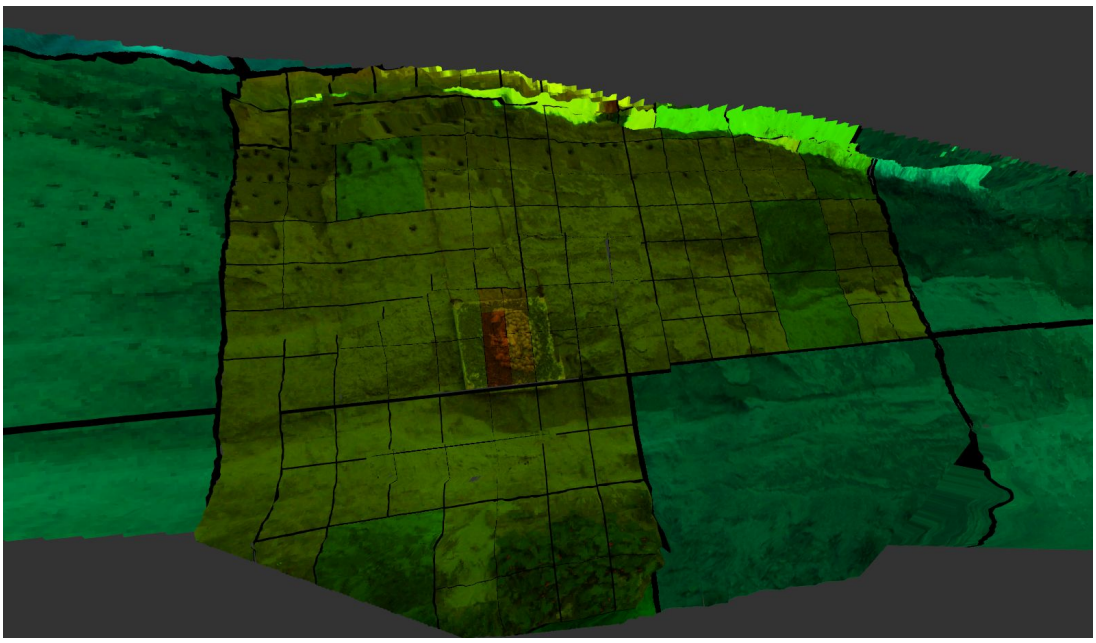
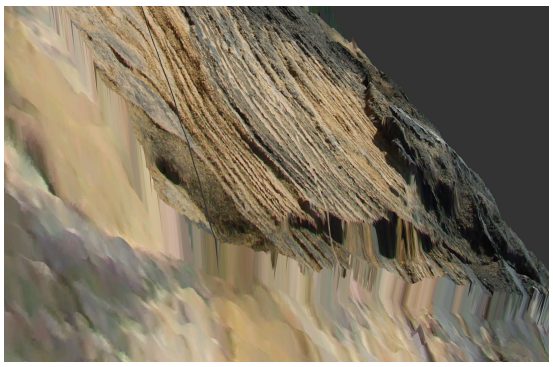
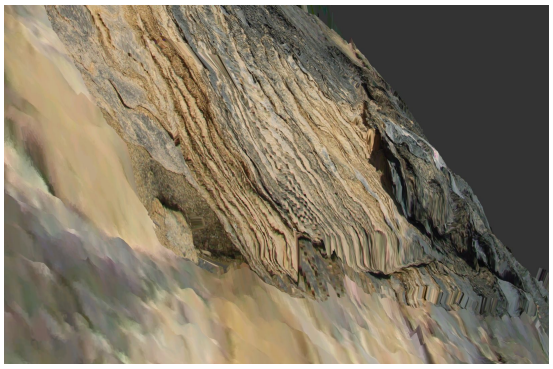


Figure 4.6: This figure shows the Pellheim merge result in level of detail false colors. It illustrates areas of different detail within the model. The more red the color is the higher the detail of the surface. This image was produced by the rendering and visualization tool of the team of VRVis [VRVis, 2013].



(a) Merging two models without registering.



(b) Merging two models with registering.

Figure 4.7: This figure shows the differences when merging models with or without registering them before. The model shown in (a) shows merging without prior registering while (b) shows the same merge with registering. These images were produced by the rendering and visualization tool of the team of VRVis [VRVis, 2013].

Chapter 5

Future Work

During the project, a few initial requirements could not be met at the time of this writing due to conceptual re-design and time constraints and were considered to be future work. These will be described in this chapter. Beside these issues, there are different ways for the future to improve or extend this work beyond the original specification or the initial requirements. Very promising approaches are outlined in this chapter as well as the prospective field of activity of our work. We find the most promising fields of improvement to be memory management and the simplification process. In both cases we see potential in either computation efficiency or memory consumption or both. In the following sections we will compare the initial requirements and the initial concept to the outcome and contrast implemented features to those which had to be considered as future work although being addressed in the initial concept in section 5.1. The Sections 5.2 and 5.3 will describe the concepts to cope with these issues. Moreover, we will outline first drafts for improvements in that field in Sections 5.4 and 5.5. While this work is still under experimental evaluation at the time of this writing there are potential applications which can take advantage of the *Versatile Surface Model*. This chapter will also describe different applications and future domains for this work in Section 5.6.

5.1 Contrasting juxtaposition of concepts

During the advancement of the work the underlying concept made its own evolution. At the beginning of the work on this thesis the requirements of the project stated in Section 1.3 led to an initial concept trying to follow all these needs. The idea of this concept was to create models on the basis of connected DEM patches as described in 3.1. This idea threw up several conflicts between different requirements and therefore evolved to the concept based on point cloud patches and a level of detail hierarchy similar to the OPC file structure which we used for storage and swapping. But due to time constraints, it was not possible to meet all the requirements in the final concept which we first wanted to address in the initial concept. Although some goals of the projects were missed, the core functionality was developed. Several features could be implemented and tested including

- the development of a multi-resolution surface model
- which can represent arbitrarily-oriented objects
- in a watertight surface,
- enables overview levels of the model,
- allows updating the model by highly detailed data parts e.g. close-up reconstructions on a large scale terrain

- and serves the *Ordered Point Cloud* file system used by Ortner's visualization method.

Comparing this disposition to the projects requirements and the basic ideas which were addressed in the initial concept, there are still some attributes we have not been able to develop until the time of this writing but which had originally been planned.

Not only do objects or terrains of interest vary in remote sensing; recording hardware and sensors do as well. The different kinds and shapes of surfaces were required to be addressed by this work. Surfaces which are arbitrary in shape can be represented, but the variance in the recording geometry is limited. This limitation mainly comes into effect when surface models created by different sensor geometry need to be merged. Differences in the geometry of the recording sensor may not only result in different coordinate systems; they also affect local densities in the surface. The current prototype of this project is limited to Cartesian-based systems and corresponding sensor geometries when it comes to merging surface models.

Adopting the hierarchy structure of the OPC file system brings in several advantages in efficiency, but disadvantages also exist. As the *Ordered Point Cloud* file system was created as a file basis for an out-of-core visualization system, the human perception was brought into focus. For the human perception the quality of a surface model's texture is much more important than the quality of the vertex mesh. Therefore, for the visualization a much higher image than mesh resolution is expected. We want to address this limitation by further extending the OPC format without losing compatibility with Ortner's visualization method, which at the time of this writing was not possible due to time constraints.

On the contrary, the possibility to directly manipulate a surface model stored on an OPC basis was not initially planned. A connection to Ortner's visualization method creating an OPC model was necessary, though any manipulation of existing models was not envisaged. Also the file structure was extended to this work's needs. This development evolved from the concept changes goes beyond the initial project specifications and creates new prospects for using results of this thesis.

While not every requirement could be met during the development of this work, the core system could be created serving the main needs stated in Section 1.3. Owing to the evolution of the concept and time constraints, the missing specifications have to be considered to be future work. But the projects results have enormous potential in extension, further development and reaching the future work's goals. Furthermore, it went beyond the original specifications and provided additional benefit to corporate users to legitimate concept changes and the outcome.

5.2 OPC resolution limitation

As the *Ordered Point Cloud* (OPC) was introduced as a file basis for a rendering and visualization system, human perception was of high importance. In human perception the quality or resolution of a surface's texture is more important than its mesh's resolution. Thus, to meet the users' demands, the OPCs are generally created with a texture resolution several times higher than the vertex resolution. To cope with these needs, this work must supply each model with the same feature beside the general compatibility requirements to the file structure, which also means that for the highest available texture resolution the supplied 3D data resolution can only be a fraction of it. Depending on the recording sensors, the actual resolution of the 3D data could be much higher in relation to the texture resolution. In some cases this data loss may be not acceptable.

The future goal is to extend the OPC file format without losing compatibility with Ortner's visualization method. The different data pieces of one OPC surface representation are composed of a logic directory structure. The different directories and files are then linked together by several *Extensible Markup Language* (XML) files. In this work the OPC file structure design was already extended by a few independent XML entries and additional XML files. The first idea to avoid this data loss is to extend the OPC structure by additional levels of detail which are invisible for the visualization method and

therefore do not compromise its functionality. Patches of any resolution only exist in the visualization method if they are linked to the hierarchy in the patch hierarchy XML file. These new patches added to contain the original resolution of the 3D data of the surface would not be enlisted in that file but in an other one. A second extensive hierarchy file could be added containing the necessary information. That means the OPC file system would feature an extended surface hierarchy for this work only. These levels would be invisible to Ortner's visualization process by being linked to the structure in an additional file. This is illustrated in Figure 5.1. The black drawn patch levels symbolize the surface levels in congruence with the original OPC file structure definition. The blue patch level symbolizes the additional 3D data unused by the visualization which must be defined in an additional XML file.

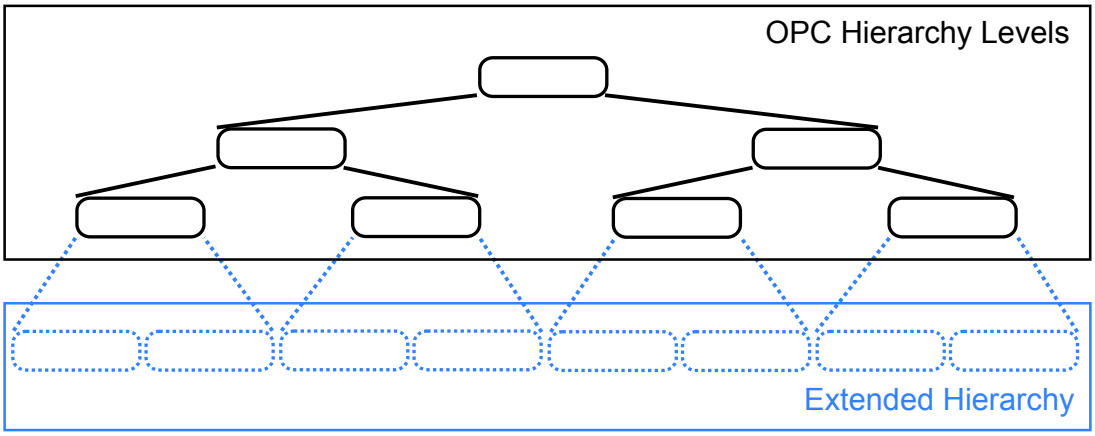


Figure 5.1: This figure illustrates the hierarchy extension to avoid data density loss. The black levels symbolize the surface levels within the OPC hierarchy. The blue level shows the extended level defined by an additional hierarchy file containing the surface structure's highest resolution unused by the visualization.

An upcoming challenge would be merging two of those models. The bottom layers would then feature a special characteristic. Because non-existent to the visualization, nothing can be added or inserted under it in the patch hierarchy file. Every level added to those could not be rendered and visualized. When adding high detail surface data to such a model, it must be done at the bottom of the lowest level available for the visualization. This could cause the loss of the data within the extended levels if newly added patches replaced these partly and did not fully cover the area of the replaced patches. The uncovered parts would be filled with interpolated surface parts of higher and coarser levels and the some surface details preserved in the extended levels would be lost. This problem is demonstrated in Figure 5.2. A special treatment for those models would be necessary to attain both visualization compatibility and avoidance of data loss. In the case of the example in this figure the surface data of both left patches in the extended level must be preserved for the outer areas which are uncovered by the detail data to be inserted.

5.3 Geometry limitation

Different sensor geometries can not be managed and processed in an adequate way. This takes effect when merging surface models where each model was recorded by sensors with different geometry. When resampling the vertices of the source model into the raster of the target model, it is not possible to use the current method. This method is based on matching a few significant vertices of the models. For the raster positions between theses vertices, the vertices are linearly interpolated from four neighbors of the original raster. This would result in deformed shapes when sensor geometries differ. One raster could not be match into the other linearly. This was previously described in Section 3.3.2. In Figures 5.3 and 5.4 this process is symbolized. Figure 5.3 shows the two surface models in their original rasters to be

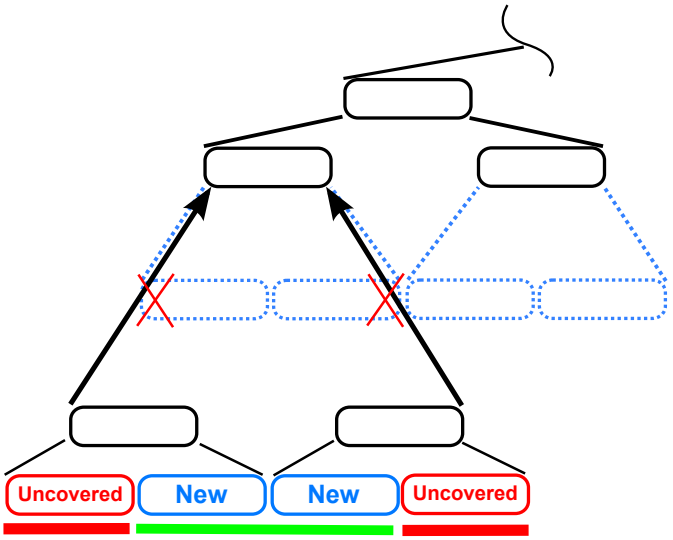


Figure 5.2: In this figure the insertion of surface data of higher detail into an extended model is shown. Without taking care of the special characteristics of this model, the surface data in the outer areas of the extended patches on the left would be lost. It would be replaced by the coarser version of the surface one level above.

merged together. The vertices of the smaller surface to be inserted into the other are rastered into grid of the patches added to the other model by matching a few vertices and interpolating the others between them linearly. The raster geometry of the new patches is the same of the patches' parents geometry to have a correlation between them. Therefore, the vertices being interpolated between the matched vertices are interpolated in the raster geometry of the target model not the source model. This results in the deformation which is represented in Figure 5.4.

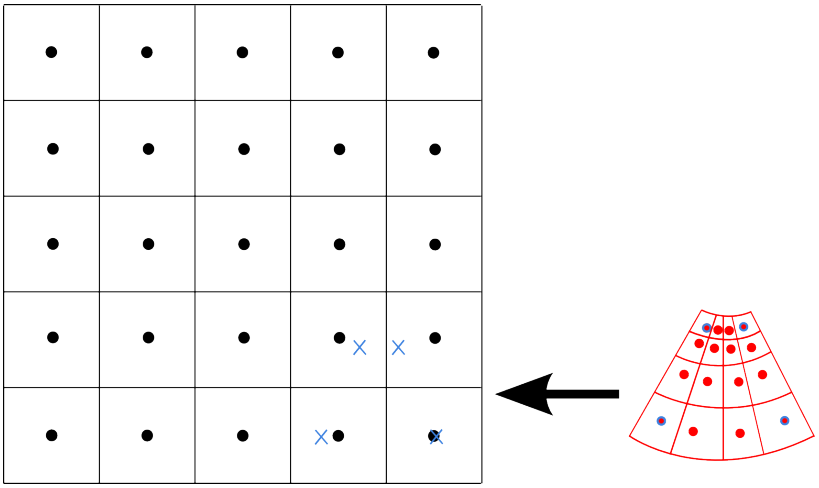


Figure 5.3: Illustration of two surfaces with different geometry to be merged. The figure shows the vertices of these surfaces and symbolizes four of them to be matched into the larger surface. The others between them would be linearly interpolated to fit in the target raster.

In the future, a method has to be found to enable the merging of models recorded with different sensor geometries. Matching only a few vertices saves processing time and was therefore an interesting approach. A possible solution would be to match each vertex individually into the new raster. But matching each single point to be inserted into the target model to its raster would cause the process to take multiple times that of the current method. The advantage, on the other hand, would be to be independent

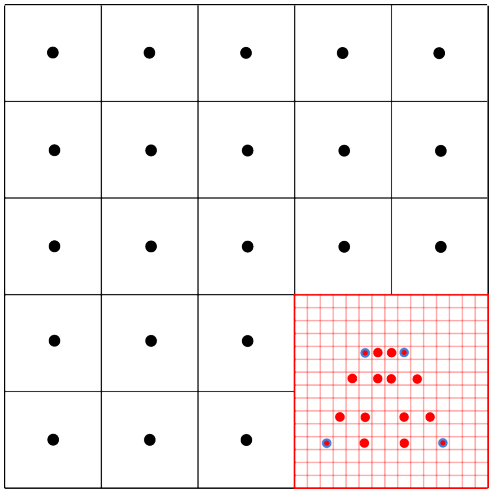


Figure 5.4:
This figure shows the two surfaces from 5.3 merged together. The surface part of the red model is deformed by being linearly interpolated to fit in the raster.

of any geometry differences. Another approach could be a different polynomial interpolation method depending on the different geometries to be matched to each other. Partly linear interpolation between a number of control points could also be evaluated for certain geometries, which means not only matching a few significant vertices like currently used, but matching a certain number of vertices between these significant points too to apply different interpolation parameters between these sub-sections.

Nevertheless, another problem arises when rastering vertices into a model raster of different sensor geometry. Transferring the vertices into another system would cause different densities in the new raster. This would mean that in one area the vertex density could have a completely different resolution than in another area. The red raster symbolizes the raster of the newly added patches. It must be significant higher to be able to insert the vertices at a correct resolution. Therefore, creating a raster suitable to the highest resolution would cause the raster in other areas to have multiple holes as symbolized by the high number of empty raster cells. These areas would require interpolation, too, to fill the empty areas caused by resampling a model of different geometry. This can also be seen in Figure 5.4.

5.4 Advanced memory management

The memory management of our data structure was explained in Section 3.4. It uses the general idea of virtual memory management as used by operating systems. The total amount of data is divided into portions. While these portions are referred to as pages when talking about memory, in operating systems, the equivalent in this work are the patches. The basic idea of only loading data into memory on request can also be compared to a page fault. This occurs when some data is requested, but not available in memory. This can be applied to pages of memory and patches of a 3D surface as well. When working on large models frequent paging may occur. Especially when working with low-end computers, the threshold of allowed patches in memory may be very small, which is why an efficient paging or swapping algorithm could speed up computing significantly.

Currently we use a slightly modified FIFO approach. FIFO or *first-in-first-out* swaps that portion of data out which was allocated first. Slightly similar to *Second-chance* [Tanenbaum, 2007], which adheres the reference bit of a page, our idea was to pay attention to the state of the patch. Modified patches get a second chance while read-only patches are swapped immediately. Various paging algorithms have been developed. The evaluation of the different algorithm principles could lead to a more efficient patch swapping algorithm. Nevertheless, we are facing different problems than operating systems when work-

ing with memory pages. The portions of data we are dealing with are more complex than one page of memory. One patch consists of 3D points, which can be requested separately, and textures whose pixels can also be accessed independently. It is nearly impossible to observe every access to every data piece in a patch to influence the swapping mechanism. The decision criterion which patch or page to swap is based on information about usage and access to it. The challenge will be to find an efficient way to decide which patch to swap and on the same hand, to also find an efficient method to gain all information necessary for this decision.

5.5 Surface simplification

Surface simplification is not only an issue of fast computing techniques but also of fast data transfer and efficient data storage. In Section 2.3 we gave a short overview of existing simplification techniques. Introducing an advanced simplification approach to extend and improve our level of detail pyramid hierarchy as described in Section 3.2.2 would be a worthwhile extension of this work. Several existing algorithms for simplification were designed for real-time application. One early approach designed to serve real time visualization of 3D meshes was the approach of **?**. It was an extension to his earlier version of *Progressive Meshes* [Hoppe, 1996].

The higher levels of the level of detail pyramid store the coarser representations of the models surface. A technique needs to be found to calculate these surface parts on demand. The vertices in the coarser representations depend on the vertices of the original surface and therefore on the bottom level. Calculating every vertex in a coarse patch as needed instead of storing them would reduce the memory consumption significantly. An important requirement for such an algorithm would also be computational efficiency.

Additionally, preserving distinct regions of the surface in the overview levels can be an issue. At the same time the vertex order of the data structure must be kept upright. Treating both at the same time is a challenge facing future work on this topic.

5.6 Potential applications

This work is intended to serve applications during the different processing steps of working on multiresolution terrain models and large surface or object representations. While this definition is very general, we focused on the field of remote sensing in our experiments and with the outlook for the future field of activity as well. A number of promising and particular applications are discussed in this section. What the following applications all have in common is that they demand large amounts of data depending on the different possible use cases for each one.

First to mention are path planning applications. These may work on very large terrain representations depending on the area in which a vehicle should be navigated autonomously. Digital inspections of tunnels are also of particular interest. Tunnels are an essential part of traffic infrastructure. Damages to them may carry higher risk than damages to simple transport routes while being more difficult to maintain. Surface data for maintenance can very easily reach large amounts of data with long tunnels. Last but not least, terrain models for georisk management are also in need of highly detailed models of large areas and need to store changes over time in several models of the same area.

These applications can benefit from this work. Future work could possibly be evaluating different approaches in the context of our work.

5.6.1 Path planning

Computer-aided path planning can navigate vehicles through hazardous terrain without human interaction. In the last years a lot of progress has been made in exploring distant worlds. Time and space make it impossible to navigate investigating vehicles on the surface of other objects in the universe in real-time. The latency of the data transfer would be too long. Therefore, autonomous path planning especially for planetary rovers, has become an important issue. The need to classify the rovers' surroundings and avoid hazards are part of this task. Complex structure representation can be helpful and essential for this application. We found two specific parts of the process of path planning worth mentioning, which could profit from our work. Integrating those into a path planning framework with a continuously updated and extended surface model could be a future domain for our approach.

Obstacle detection and terrain classification

Obstacle collisions must obviously be avoided when navigating a vehicle autonomously through off-road terrain. Manduchi et al. [2005] introduced a point cloud based terrain classification algorithm. Obstacles in the direction of motion are segmented and detected. They also proposed a solution of vegetation classification to provide a vehicle with further information regarding the optimal path and velocity traversing a certain terrain. Acquired 3D information of the terrain must be processed efficiently. When moving the vehicle through the terrain new information continuously extends already-existing data. Depending on the specific implementation of the algorithm it can benefit from this work.

Roughness map creation

The roughness of a certain terrain also defines the difficulty of traversing it. While even terrain can be crossed more quickly and involves less danger of damage to the vehicle or risk of failing for the mission, uneven terrain must be traversed with slower speed and may be even impassable. Elkaim and Loh envisaged a method of roughness mapping for planetary rovers [Loh et al., 2013] based on a point cloud data structure to increase the rovers' efficiency. With their results they showed the possibility to perform an approach for such an appliance in real time directly on the rover. The implementation of a real-time solution for roughness map creation for our data structure is a valuable approach for the use with large data sets which may be updated frequently as the rover moves.

5.6.2 Tunnel planning and maintenance

Tunnels are an essential element of modern transportation facilities, for both road and rail. Planning and maintenance of tunnels benefit from detailed and geometrically exact representations. Ortner et al. [2010] developed a surface documentation technique to support engineers in all phases of a tunnels lifetime, construction, maintenance and repair. As outlined in Chapter 3, recording a tunnel's surface can result in massive amounts of data. To process long tunnels, a data structure must be able to handle these numbers. This work is designed to cope with the requirements these large surface models are in need of.

In [Ortner et al., 2011] the advantages of digital high-resolution tunnel surface documentation are underlined. Measurements can be performed without closing the tunnel, and it enables the combination of different data sources to allow new insights and perspectives on the object of investigation. Ortner et al. found several applications which benefits from these advantages, including

- investigation of railway tunnel clearance,
- cross-section calculations and
- localization and measurement of cracks in the tunnel surface.

Ortner et al. [2011] suggested human interacted digital tunnel inspections. To support this, they introduced visualization techniques for digital transportation tunnel representations. Automated algorithms could replace subtasks done by human interaction during the digital tunnel inspection process. Calculations on the digital surface models for those tasks would profit from this work.

5.6.3 Georisk management

To monitor natural and geological hazards, forecast and prevent possible outcomes of geological and soil movement respective areas and objects must be measured and recorded repeatedly. This can apply to several distinct region or terrain types. This means coastal areas or regions near rivers. These landscapes are exposed to the risk of landslides or flooding. Gutierrez et al. [2001] proposed surveillance methods for these regions.

In [Lovas et al., 2008] the authors suggest the use of laser scanners in deformation measurement in civil engineering. They process the point clouds of laser scanners to detect structures or buildings deformations.

TLS based georisk analyses can also be useful for the evaluation of transport infrastructure. Miller et al. [2008] introduced a TLS based inspection of transport embankments to detect changes due to slope instability.

Several applications which were developed to avoid geological risks are based on digital 3D data like point clouds resulting from digital laser scans. Most of them need to update their datasets regularly. As shown above, several algorithms were introduced working on large 3D data sets observing terrains to avoid geological hazards. This work also provides handling techniques for such large datasets. It allows adding newly acquired data to existing models and as well as the representation of complex structures as nature creates.

Chapter 6

Conclusion

In our work we describe a solution for working with 3D surface models with multiple resolutions. The proposed representation method enables surface models featuring arbitrary levels of detail for differing areas of the surface. This is based on a novel hierarchical patch pyramid representation. Resolution of texture and 3D structure can differ depending on the recording hardware and software. Applications can add newly retrieved surface data to an existing model to a certain area of special interest within the surface. These surface parts can show a significantly more detailed view of that specific surface part.

This approach was tested intensively with different terrain reconstructions. A multiple resolution model was created iteratively from several terrain reconstructions providing detail data for specific areas of the terrain. These models showed different areas of the same terrain and exhibited a different degree of detail. That satisfies the needs in the field of remote sensing when reconstructing natural objects from different sources. Different areas of a terrain can be of different interest and therefore require lower or higher details. Furthermore, different kinds of terrain of different applications require different reconstruction methods or sensor types which exhibit differences in resolution and detail. With this thesis a versatile multi-resolution surface model representation for those issues is suggested. Working with such multi-resolution data sets or creating them from surface models with different resolutions of the same object or terrain can be done in a very intuitive and natural way through the organization of the surface within the data structure.

This work also describes a solution for working on large surface models with limited computational capacities. The capability of handling very large data sets is achieved by the division of the model into patches. Parts which are currently not needed in the process can be swapped out effectively. That means by dividing one surface model into several portions of data only the current requested parts need to be present in the main memory while the rest of the model only needs to reside on the hard disk. Large models can be handled out of core when they do not fit into the main memory as a whole. The whole hierarchical level of detail pyramid structure which allowed locally differing levels of detail and several overview levels is divided into a number of homogeneous patches. Each patch comprises a portion of the surfaces vertex and texture information in a quadratic regular grid within its detail level of the surface. The vertices are organized in an ordered point cloud, which means that each neighboring vertex in the grid is a direct neighbor in the surface. That characteristic makes access to requested regions very intuitive.

This is also demonstrated in our experiments. Each merging step of the multi-resolution data set shown increases the memory consumption of the model. The experimental process involved only parts of the whole surface at one time, like many applications do. It is shown in Chapter 4 during our test cases that only about one $\frac{1}{4}$ of the whole model was ever present in the memory due to the out-of-core model management.

We present different applications which can benefit from this work in Chapter 5. These applications can utilize the contributions of this work summarized in this chapter. Mainly the needs of the field of

remote sensing are addressed. This work seems especially profitable for applications which need to work with multiple resolutions of a terrain or object. Nevertheless, we also find our method useful for any application in need of working with a wide variety of different surface details within the same terrain or object. Applications representing large amounts of 3D surface data can also benefit from this work along with any application featuring a work flow similar to the applications mentioned in Chapter 5. At the time of this writing, the project is being evaluated in corporate business and we will continue improving and extending our approach.

Appendix A

Implementation Supplements

A.1 Realization of patch matching

Section 3.3 describes the process of merging two surface models. In mainly consists of three steps. First the patches of each model are matched to identify the patches of each model which are involved in the merging process. The second step is the surface insertion and the third step the interpolation and surface reduction. In this section the patch matching process is illustrated in Program 1.

Algorithm 1 Matching of patches

Require: $ModelA$ and $ModelB$ be surface models where $ModelB$ gets merged into $ModelA$

```
function MATCHMODELPATCHES(ModelA, PatchB)
    ListConcernedPatchesA ← ∅
    for i ← 0 to ModelA.GetNumberOfPatches() do
        IsOverlapping ← ISOVERLAPPING(ModelA.GetPatch(i),ModelB.GetRootPatch())
        if IsOverlapping & ModelA.GetPatch(i).GetLevel() = LowestLevel then
            ListConcernedPatchesA := ListConcernedPatchesA ∪ {ModelA.GetPatch(i)}
        end if
    end for

    ListMatchedPatches ← ∅
    for i ← 0 to ListConcernedPatchesA.Length() do
        ListConcernedPatchesB ← ∅
        for j ← 0 to ModelB.GetNumberOfPatches() do
            IsOverlapping ← ISOVERLAPPING(ModelB.GetPatch(j),ListConcernedPatchesA[i])
            if IsOverlapping & ModelB.GetPatch(j).GetLevel() = LowestLevel then
                ListConcernedPatchesB := ListConcernedPatchesB ∪ {ModelB.GetPatch(j)}
            end if
        end for
        CurrentPair ← (ListConcernedPatchesA[i], ListConcernedPatchesB)
        ListMatchedPatches := ListMatchedPatches ∪ {CurrentPair}
    end for
    return ListMatchedPatches
end function
```

A.2 Realization of ICP registration

In Section 3.3.4 it is stated that for the registration of two surfaces, an ICP-based approach has been implemented in the prototype of this work. The basis for this implementation was the *Point Cloud Library* (PCL) which was introduced by Rusu and Cousins [2011]. The PCL library implements an ICP algorithm registering two PCL point cloud objects. As presented in Section 3.3.4, the vertices of each patch of the surface should be transformed in the same way to avoid creating holes in the surface. The registration of the PCL library is used to calculate the transformation on coarser overview levels of the models in order to save computation time. For the more detailed model, the coarsest level available is taken, while for the other model, the level fitting best to it in resolution and density. The transformation is then applied to each patch which is due to be merged to the other model. The workflow of this procedure is shown in pseudo code in Program 2.

Algorithm 2 Realization of the ICP registration

Require: $ModelA$ and $ModelB$ be surface models where $ModelB$ gets merged into $ModelA$

```
Transformation ← ∞
if doRegistration then
    Transformation ← REGISTERMODELS(ModelA, ModelB)
end if
for i ← 0 to ModelB.GetNumberOfPatches() do
    if ModelB.GetPatch(i).GetLevel() = LowestLevel then
        if doRegistration then
            ModelB.GetPatch(i).TransformPointCloud( Transformation )
        end if
        ModelA.MergePatchData( ModelB.GetPatch(i) )
    end if
end for

function REGISTERMODELS(ModelA, ModelB)
    pclA ← ModelA.GetHighestOverviewPatch().GetPointCloud()
    LevelDesityA ← ModelA.GetLevelDensity( HighestLevelA )
    MinDistance ← ∞
    BestLevel ← ∞
    for i ← 0 to ModelB.GetNumberOfLevels() do
        if LevelDesityA − ModelB.GetLevelDensity(i) < MinDistance then
            MinDistance ← LevelDesityA − ModelB.GetLevelDensity(i)
            BestLevel ← i
        end if
    end for
    pclB ← ∅
    for i ← 0 to ModelB.GetNumberOfPatches() do
        if ModelB.GetPatch(i).GetLevel() = BestLevel then
            pclB ← pclB + ModelB.GetPatch(i).GetPointCloud()
        end if
    end for
    return ITERATIVECLOSESTPOINT(pclA, pclB)
end function

function ITERATIVECLOSESTPOINT(pclA, pclB)
    ICP implementation of Point Cloud Library [PCL, 2014b]
    return Transformation
end function
```

Bibliography

Allen, Peter K and Paul Michelman [1989]. *Acquisition and interpretation of 3-D sensor data from touch*. In *Interpretation of 3D Scenes, 1989. Proceedings., Workshop on*, pages 33–40. IEEE. (Cited on page 7.)

Andrews, Keith [2006]. *Writing a Thesis: Guidelines for Writing a Master's Thesis in Computer Science*. Graz University of Technology, Austria. http://ftp.iicm.edu/pub/keith/thesis/. (Cited on page ix.)

Besl, Paul J. and Neil D. McKay [1992]. *A Method for Registration of 3-D Shapes*. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2), pages 239–256. ISSN 0162-8828. doi:10.1109/34.121791. http://dx.doi.org/10.1109/34.121791. (Cited on page 9.)

Carson, George S., Richard F. Puk, and Rikk Carey [1999]. *Developing the VRML 97 International Standard*. *IEEE Comput. Graph. Appl.*, 19(2), pages 52–58. ISSN 0272-1716. doi:10.1109/38.749123. http://dx.doi.org/10.1109/38.749123. (Cited on page 8.)

Chen, Yang and Gérard Medioni [1992]. *Object modelling by registration of multiple range images*. *Image Vision Comput.*, 10(3), pages 145–155. ISSN 0262-8856. doi:10.1016/0262-8856(92)90066-C. http://dx.doi.org/10.1016/0262-8856(92)90066-C. (Cited on page 9.)

de Boor, C. [2001]. *A Practical Guide to Splines*. Number Bd. 27 in Applied Mathematical Sciences, Springer. ISBN 9780387953663. http://books.google.at/books?id=m0QDJvBI_ecC. (Cited on page 5.)

DeCoro, Christopher and Natalya Tatarchuk [2007]. *Real-time mesh simplification using the GPU*. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 161–166. I3D '07, ACM, New York, NY, USA. ISBN 978-1-59593-628-8. doi:10.1145/1230100.1230128. http://doi.acm.org/10.1145/1230100.1230128. (Cited on page 9.)

Dorai, Chitra, Gang Wang, Anil K Jain, and Carolyn Mercer [1998]. *Registration and integration of multiple object views for 3D model construction*. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1), pages 83–89. (Cited on page 10.)

Elfes, Alberto [1989]. *Using occupancy grids for mobile robot perception and navigation*. *Computer*, 22(6), pages 46–57. ISSN 0018-9162. doi:10.1109/2.30720. (Cited on page 7.)

Foley, James, Andries Van Dam, Steven Feiner, and John F. Hughes [2009]. *Computer Graphics: Principles and Practices*. 3Rev Ed Edition. Addison-Wesley Publishing Company, USA. ISBN 0321399528, 9780321399526. (Cited on pages 5 and 8.)

Foley, James D., Andries van Dam, Steven K. Feiner, and John F. Hughes [1996]. *Computer graphics (2nd ed. in C): principles and practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. ISBN 0-201-84840-6. (Cited on pages 5 and 6.)

Friedman, Jerome H, Jon Louis Bentley, and Raphael Ari Finkel [1977]. *An algorithm for finding best matches in logarithmic expected time*. *ACM Transactions on Mathematical Software (TOMS)*, 3(3), pages 209–226. (Cited on page 11.)

Gallup, David, Jan-michael Frahm, and Marc Pollefeys [2010]. *M.: A heightmap model for efficient 3d reconstruction from street-level video*. In *In: 3DPVT (2010)*. http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.187.6455. (Cited on pages 7 and 8.)

Garland, Michael and Paul S. Heckbert [1997]. *Surface simplification using quadric error metrics*. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA. ISBN 0-89791-896-7. doi:10.1145/258734.258849. http://dx.doi.org/10.1145/258734.258849. (Cited on page 9.)

Gruen, Armin and Devrim Akca [2005]. *Least squares 3D surface and curve matching*. volume 59, pages 151–174. (Cited on page 9.)

Grzeszczuk, Robert P. and Charles A. Pelizzari [1996]. *Real-time merging of visible surfaces for display and segmentation*. In Höhne, KarlHeinz and Ron Kikinis (Editors), *Visualization in Biomedical Computing*, *Lecture Notes in Computer Science*, volume 1131, pages 93–98. Springer Berlin Heidelberg. ISBN 978-3-540-61649-8. doi:10.1007/BFb0046941. http://dx.doi.org/10.1007/BFb0046941. (Cited on page 11.)

Gutierrez, Roberto, J Gibeaut, R Smyth, T Hepner, J Andrews, Christopher Weed, William Gutelius, and Mark Mastin [2001]. *Precise airborne lidar surveying for coastal research and geo-hazards applications*. *International Archives of Photogrammetry Remote Sensing And Spatial Information Sciences*, 34(3/W4), pages 185–194. (Cited on page 50.)

Hearn, Donald and M. Pauline Baker [1996]. *Computer graphics (2nd ed.): C version*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. ISBN 0-13-530924-7. (Cited on page 5.)

Hearn, Donald D., M. Pauline Baker, and Warren Carithers [2010]. *Computer Graphics with Open GL*. 4th Edition. Prentice Hall Press, Upper Saddle River, NJ, USA. ISBN 0136053580, 9780136053583. (Cited on pages 5 and 6.)

Hoppe, Hugues [1996]. *Progressive meshes*. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108. SIGGRAPH '96, ACM, New York, NY, USA. ISBN 0-89791-746-4. doi:10.1145/237170.237216. http://doi.acm.org/10.1145/237170.237216. (Cited on pages 9 and 48.)

Hoppe, Hugues [1997]. *View-dependent refinement of progressive meshes*. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 189–198. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA. ISBN 0-89791-896-7. doi:http://doi.acm.org/10.1145/258734.258843. (Cited on page 9.)

Hu, Liang, Pedro V. Sander, and Hugues Hoppe [2009]. *Parallel view-dependent refinement of progressive meshes*. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 169–176. I3D '09, ACM, New York, NY, USA. ISBN 978-1-60558-429-4. doi:10.1145/1507149.1507177. http://doi.acm.org/10.1145/1507149.1507177. (Cited on page 9.)

Johnson, Andrew Edie and Sing Bing Kang [1999]. *Registration and integration of textured 3D data*. *Image and vision computing*, 17(2), pages 135–147. (Cited on page 10.)

Kraus, K. [2000]. *Photogrammetrie 3. Topographische Informationssysteme*. Walter de Gruyter, Berlin. ISBN 9783110181647. (Cited on page 7.)

Kraus, Karl [2007]. *Photogrammetry - Geometry from Images and Laser Scans - 2nd edition*. Walter de Gruyter, Berlin. ISBN 978-3-11-019007-6. (Cited on page 7.)

Kraus, Karl and Norbert Pfeifer [1998]. *Determination of terrain models in wooded areas with airborne laser scanner data*. *ISPRS Journal of Photogrammetry and remote Sensing*, 53(4), pages 193–203. (Cited on page 7.)

Li, Xiaokun and W.G. Wee [2004]. *Range image fusion for object reconstruction and modeling*. In *Computer and Robot Vision, 2004. Proceedings. First Canadian Conference on*, pages 306–314. doi:10.1109/CCCRV.2004.1301460. (Cited on page 11.)

Li, Z., C. Zhu, and C. Gold [2010]. *Digital Terrain Modeling: Principles and Methodology*. Taylor & Francis. ISBN 9780203486740. http://books.google.at/books?id=JvEo41LqjtUC. (Cited on pages 7 and 8.)

Lindstrom, Peter [2000]. *Out-of-core simplification of large polygonal models*. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 259–262. SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA. ISBN 1-58113-208-5. doi:10.1145/344779.344912. http://dx.doi.org/10.1145/344779.344912. (Cited on page 9.)

Lindstrom, Peter and Greg Turk [1998]. *Fast and memory efficient polygonal simplification*. In *Proceedings of the conference on Visualization '98*, pages 279–286. VIS '98, IEEE Computer Society Press, Los Alamitos, CA, USA. ISBN 1-58113-106-2. http://dl.acm.org/citation.cfm?id=288216.288288. (Cited on page 9.)

Loh, Jonathan E., Gabriel H. Elkaim, and Renwick E. Curry [2013]. *Roughness Map for Autonomous Rovers*. In *American Control Conference (ACC13)*. Submitted. (Cited on page 49.)

Lorensen, William E and Harvey E Cline [1987]. *Marching cubes: A high resolution 3D surface construction algorithm*. In *ACM Siggraph Computer Graphics*, volume 21, pages 163–169. ACM. (Cited on page 11.)

Lovas, T, Á Barsi, A Detrekoi, L Dunai, Z Csak, A Polgar, A Berényi, Z Kibedy, and K Szocs [2008]. *Terrestrial laser scanning in deformation measurements of structures*. *International Archives of Photogrammetry and Remote Sensing*, 37(B5), pages 527–531. (Cited on page 50.)

Manduchi, R., A. Castano, A. Talukder, and L. Matthies [2005]. *Obstacle Detection and Terrain Classification for Autonomous Off-Road Navigation*. *Autonomous Robots*, 18, pages 81–102. ISSN 0929-5593. http://dx.doi.org/10.1023/B:AURO.0000047286.62481.1d. 10.1023/B:AURO.0000047286.62481.1d. (Cited on page 49.)

Matiukas, V. and D. Miniotas [2011]. *Point Cloud Merging for Complete 3D Surface Reconstruction*. In *ELECTRONICS AND ELECTRICAL ENGINEERING*, volume 113, pages 73–76. Kaunas University of Technology. doi:10.5755/j01.eee.113.7.616. (Cited on page 10.)

Miller, PE, JP Mills, SL Barr, M Lim, D Barber, G Parkin, B Clarke, S Glendinning, and J Hall [2008]. *Terrestrial laser scanning for assessing the risk of slope instability along transport corridors*. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37, pages 495–500. (Cited on page 50.)

Mortenson, M.E. [1997]. *Geometric modeling*. Wiley. ISBN 9780471129578. http://books.google.at/books?id=TPVQAAAAMAAJ. (Cited on page 5.)

Newcombe, Richard A, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon [2011]. *KinectFusion:*

Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE. doi:10.1109/ISMAR.2011.6092378. (Cited on page 11.)

Ortner, T, G Hesina, H Kontrus, and R Glatzl [2011]. *Hochauflösende flächendeckende Dokumentation von Tunneloberflächen. in Proceedings of Symposium und Fachmesse für Angewandte Geoinformatik (AGIT 2011)*, pages pp. 914–923. (Cited on pages 49 and 50.)

Ortner, T, G Paar, G Hesina, R Tobler, and B Nauschnegg [2010]. *Towards True Underground Infrastructure Surface Documentation. in Proceedings of CORP 2010*, pages pp. 783–792. http://www.vrvis.at/publications/PB-VRVis-2010-008. (Cited on pages 3, 15, 16, 17, 28 and 49.)

Ortner, T and R Tobler [2010]. *VILMA Data Structure Design*. Technical description of a filestructure, VRVis, VRVis, Donau-City-Strasse 1, Vienna, Austria. (Cited on pages 28 and 31.)

PCL [2014a]. *Point Cloud Library (PCL) registration documentation*. http://docs.pointclouds.org/1.0.0/group__registration.html. (Cited on page 10.)

PCL [2014b]. *Point Cloud Library (PCL) Website*. http://docs.pointclouds.org/trunk/a02953.html. (Cited on page 55.)

Research, Joanneum [2013]. *Joanneum Research Website*. http://www.joanneum.at/digital/das-institut/forschungsgruppen/bildanalyse-und-messsysteme.html. (Cited on page 30.)

Rusu, Radu B. and Steve Cousins [2011]. *3D is here: Point Cloud Library (PCL)*. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4. IEEE. doi:10.1109/ICRA.2011.5980567. (Cited on pages 8 and 54.)

Samet, Hanan [1984]. *The Quadtree and Related Hierarchical Data Structures*. *ACM Comput. Surv.*, 16(2), pages 187–260. ISSN 0360-0300. doi:10.1145/356924.356930. http://doi.acm.org/10.1145/356924.356930. (Cited on page 6.)

Shotton, Jamie, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore [2013]. *Real-time human pose recognition in parts from single depth images*. *Communications of the ACM*, 56(1), pages 116–124. (Cited on page 7.)

Soucy, Marc and Denis Laurendeau [1992]. *Multi-resolution surface modeling from multiple range views*. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 348–353. IEEE. (Cited on page 10.)

Strugar, Filip [2009]. *Continuous Distance-Dependent Level of Detail for Rendering Heightmaps*. *Journal of Graphics, GPU, and Game Tools*, 14(4), pages 57–74. doi:10.1080/2151237X.2009.10129287. http://www.tandfonline.com/doi/abs/10.1080/2151237X.2009.10129287. (Cited on page 9.)

Tanenbaum, Andrew S. [2007]. *Modern Operating Systems*. 3rd Edition. Prentice Hall Press, Upper Saddle River, NJ, USA. ISBN 9780136006633. (Cited on page 47.)

Thibault, William C. and Bruce F. Naylor [1987]. *Set Operations on Polyhedra Using Binary Space Partitioning Trees*. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 153–162. SIGGRAPH '87, ACM, New York, NY, USA. ISBN 0-89791-227-6. doi:10.1145/37401.37421. http://doi.acm.org/10.1145/37401.37421. (Cited on page 6.)

VRVis [2013]. *VRVis Website*. http://http://www.vrvis.at/impressum/. (Cited on pages ix, 17, 18, 21, 36, 37, 38, 40 and 41.)

Zhang, Li, Brian Curless, and Steven M Seitz [2002]. *Rapid shape acquisition using color structured light and multi-pass dynamic programming*. In *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pages 24–36. IEEE. doi:10.1109/TDPVT.2002.1024035. (Cited on page 7.)

Zhang, Zhengyou [1994]. *Iterative point matching for registration of free-form curves and surfaces*. *Int. J. Comput. Vision*, 13(2), pages 119–152. ISSN 0920-5691. doi:10.1007/BF01427149. http://dx.doi.org/10.1007/BF01427149. (Cited on page 9.)