Scalable Realtime Rendering and Interaction with Digital Surface Models of Landscapes and Cities

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Roland Wahl
aus Bonn

Bonn 2015

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Reinhard Klein
2. Gutachter: Prof. Dr. Andreas Schilling
Tag der Promotion: 21.07.2016
Erscheinungsjahr: 2016

Abstract

Interactive, realistic rendering of landscapes and cities differs substantially from classical terrain rendering.

Due to the sheer size and detail of the data which need to be processed, realtime rendering (i.e. more than 25 images per second) is only feasible with level of detail (LOD) models. Even the design and implementation of efficient, automatic LOD generation is ambitious for such out-of-core datasets considering the large number of scales that are covered in a single view and the necessity to maintain screen-space accuracy for realistic representation. Moreover, users want to interact with the model based on semantic information which needs to be linked to the LOD model.

In this thesis I present LOD schemes for the efficient rendering of 2.5d digital surface models (DSMs) and 3d point-clouds, a method for the automatic derivation of city models from raw DSMs, and an approach allowing semantic interaction with complex LOD models.

The hierarchical LOD model for digital surface models is based on a quadtree of precomputed, simplified triangle mesh approximations. The rendering of the proposed model has proven to allow real-time rendering of very large and complex models with pixel-accurate details. Moreover, the necessary preprocessing is scalable and fast.

For 3d point clouds, I introduce an LOD scheme based on an octree of hybrid plane-polygon representations. For each LOD, the algorithm detects planar regions in an adequately subsampled point cloud and models them as textured rectangles. The rendering of the resulting hybrid model is an order of magnitude faster than comparable point-based LOD schemes.

To automatically derive a city model from a DSM, I propose a constrained mesh simplification. Apart from the geometric distance between simplified and original model, it evaluates constraints based on detected planar structures and their mutual topological relations. The resulting models are much less complex than the original DSM but still represent the characteristic building structures faithfully.

Finally, I present a method to combine semantic information with complex geometric models. My approach links the semantic entities to the geometric entities on-the-fly via coarser proxy geometries which carry the semantic information. Thus, semantic information can be layered on top of complex LOD models without an explicit attribution step.

All findings are supported by experimental results which demonstrate the practical applicability and efficiency of the methods.

Acknowledgments

The work presented in this thesis would not have been possible without the help and support of many colleagues, students, friends and mentors.

First of all, I am grateful to Patrick Degener a comrade since the time of our diploma studies, with whom I shared an office for many years. We had many fruitful discussions and would often ask each other for advice on new ideas before sharing with anyone else. Actually, differing in degree this holds for everyone else in the computer graphics group, students and co-workers alike, I owe them a lot. Anyhow, I would like to mention Marcin Novotni, Gabriel Zachmann, Jan Meseth, Gerhard Bendels, Alexander Gress, Michael Guthe, Martin Schneider, Ruwen Schnabel, Markus Schlattmann, Raoul Wessel, Manuel Massing, Marcel Körtgen and Sebastian Möser with whom I worked intensely at least on one occasion and with most of whom I also shared leisure-time activities.

As most of my research associate position was funded by the Deutsche Forschungsgemeinschaft, I am very grateful to them. Even more so, as they offered me the opportunity to be part of an interdisciplinary research team, whose regular meetings were an invaluable source of inspiration. In place of all the members of the research group I will just address Prof. Monika Sester and Prof. Wolfgang Förstner who were the driving forces behind the project and its organisation. Thanks and Greetings to all the hosts, colleagues and advisors from the Skalen-Bündel.

Of course, at the core of all my research is the influence of my advisor Reinhard Klein. Since I entered his office for the first time applying for a student assistant position in 2001, we developed countless ideas of which this thesis can only reflect a tiny part. From our first encounter, he never would make me feel subordinate but always treated me as a younger colleague. Over the time we became so familiar that often we would agree on one topic before even phrasing the solution. I will always keep him in good memory.

The last sentence of the Acknowledgments, which happens to be the last written sentence of this thesis, is dedicated to my relatives and friends. They will know that I mean them when they read this and feel my gratefulness.

Contents

Abstract i

Acknowledgments iii

Contents v

List of Figures ix

List of Tables xi

List of Abbreviations xiii

1 Introduction 1
- 1.1 Challenges . . . 2
- 1.2 Contributions . . . 5
- 1.3 Publications . . . 8

2 Scalable Compression and Rendering of Textured Terrain Data 11
- 2.1 Introduction . . . 11
- 2.2 Related work . . . 13
- 2.3 Overview . . . 14
- 2.4 Tile tree construction . . . 15
  - 2.4.1 Error Bounds . . . 16
  - 2.4.2 Simplification . . . 17
  - 2.4.3 Textures . . . 18
  - 2.4.4 Compression . . . 19
- 2.5 Rendering . . . 19
  - 2.5.1 Quadtree Update . . . 20
  - 2.5.2 Repairing Cracks . . . 20
  - 2.5.3 Caching & Prefetching . . . 20
  - 2.5.4 Output Sensitivity . . . 21
  - 2.5.5 Occlusion Culling . . . 21
  - 2.5.6 Impostors . . . 21

2.6 Implementation & results . . . 22
2.7 Conclusion & future work . . . 23
2.8 Acknowledgements . . . 24

3 Hybrid Rendering 25
3.1 Introduction . . . 25
3.2 Related Work . . . 27
3.3 Preprocessing . . . 30
3.3.1 Plane Detection . . . 30
3.3.2 Texture Generation . . . 33
3.3.3 Compression . . . 33
3.4 Rendering . . . 34
3.5 Implementation & Results . . . 35
3.6 Conclusion . . . 38
3.7 Acknowledgements . . . 39

4 Constrained DSM simplification 41
4.1 Introduction . . . 42
4.2 Previous Work . . . 43
4.2.1 Topology-Preserving Simplification . . . 43
4.2.2 Topology-Changing Simplification . . . 44
4.2.3 Out-of-Core Simplification . . . 44
4.2.4 Remeshing . . . 44
4.3 Overview . . . 45
4.4 Geometric Simplification . . . 45
4.4.1 Distance Metric . . . 46
4.5 Semantic Constraints . . . 47
4.5.1 Edges & Corners . . . 47
4.5.2 Constrained Simplification . . . 49
4.6 Results . . . 50
4.6.1 Conclusion . . . 51
4.6.2 Acknowledgements . . . 52

5 Out-of-core Constrained Simplification 53
5.1 Introduction . . . 53
5.2 Related Work . . . 55
5.2.1 Automatic City modeling . . . 55
5.3 Overview . . . 56
5.3.1 Shape Detection . . . 56
5.3.2 Constrained Simplification . . . 57
5.3.3 Problem Analysis . . . 59

5.4 Topological Constraints 61
5.4.1 Topological Corners and Edges 61
5.4.2 Treatment of Points without Shape Information - Tunnel avoiding 62
5.4.3 Topological filtering 64
5.5 Out-of-Core, Parallel Computation 65
5.6 Results 66
5.7 Conclusions 68

6 Semantic Interaction **71**
6.1 Introduction 71
6.2 Realtime Terrain Rendering 74
6.2.1 High detail terrain and city models 74
6.2.2 Insufficiencies of terrain rendering 74
6.2.3 The rendering model 76
6.3 Interactive Visualization 77
6.3.1 Implicit modeling of semantics 78
6.3.2 Interaction model 80
6.3.3 Implementation issues 81
6.3.4 Advanced interaction 81
6.4 Results 83
6.5 Conclusion 84
6.5.1 Future Work 84

7 Conclusion and Future Directions **87**
7.1 Rendering of Digital Surface Models 87
7.2 Planes in Point Clouds 89
7.3 Semantically improved modeling 90
7.4 Semantic Interaction 91

Bibliography **93**

List of Figures

2.1 The preprocessing stage. 14
2.2 The rendering stage. 15
2.3 Relationship of errors depicted in 2D. 16
2.4 Frame rates for a Puget sound fly-over. 23
2.5 Snapshot of Turtmann valley fly-over. 24

3.1 Scanned Welfenschloss point cloud exhibiting high frequency material details. 26
3.2 Texture packing. 33
3.3 Rotation of planes to visually close cracks between octree cells . . 35
3.4 Choir screen point cloud rendered with our method 36
3.5 Fitted quads and remaining points for the Welfenschloss model at octree level 4. 37

4.1 Shape detection results 48
4.2 Simplification results (unconstrained vs. constrained) 51

5.1 Rendering of the automatically reconstructed city model of downtown Berlin on top of a corresponding DTM. 54
5.2 Example of a Shape Map depicted as RGB values 58
5.3 Two cases with incomplete shape constraints. 59
5.4 Schematic depiction of critical shape constellations. 62
5.5 Left: initial situation, pseudo edge connected to the left facade, the lower edge of the right facade is discontinuous due to vegetation. Right: possible result with the original approach, black dots mark vertices without shape assignment 63
5.6 Simplification results. Left: without tunnel constraint. Right: with active tunnel constraint. Less artefacts are caused by vegetation. . 63
5.7 Simplification results. Left: without topological filtering. Right: with topological filtering. Removing spikes drastically decreases the triangles count. 64

5.8 Left: Two spikes at roof-edges, the left is protected by shape constraints and the right one only by its size, Right: spikes are removed using the topological filtering 64
5.9 Example of parallel computation. 66
5.10 Simplification results: Whole dataset 67
5.11 Simplification results. Left: unconstrained. Right: With topological constraints. Important roof-structures are preserved. 68

6.1 Quadtree layout of terrain rendering. Each part of the model belonging to a quadtree cell (called tile) has the same raster size independent of LOD. 75
6.2 Screenshot from the rendering application showing a highlighted feature of the semantic dataset within a visualization of the Berlin DSM. 84
6.3 Screenshot from the rendering application. In picking mode semantic entities are highlighted as the mouse hovers over them in the image. The user thus gets an instant feedback with which objects and on which semantic LOD he is about to interact. 85

List of Tables

2.1 Geometry statistics of tested models. 22

3.1 Preprocessing times for different point clouds. 36
3.2 Simplified point cloud and generated planes with remaining points for the Welfenschloss. 37
3.3 Simplified point cloud and generated planes with remaining points for the choir screen. 38
3.4 Frame rates for the Welfenschloss. 38
3.5 Frame rates for the choir screen. 38

List of Abbreviations

BRDF	bidirectional reflectance distrbution function
CityGML	city geography markup language
DEM	digital elevation model
DSM	digital surface model
DTM	digital terrain model
GIS	goegraphic information system
GPS	global positioning system
GPU	graphics processing unit
LiDAR	light detection and ranging
LOD	level of detail
RANSAC	random sample consensus
S3TC	S3 texture compession
TIN	triangulated irregular network, i.e. triangle mesh

CHAPTER 1

INTRODUCTION

During the last decade, a remarkable change in the experience of virtual landscapes and cities has taken place. At the end of the 1990s, we had digital terrain models usually with a resolution of 10m or lower describing only the landform. Thus, real-time terrain rendering seemed to be a solved problem. Advances in sensor technology (radar, lidar, digital photography) along with new methods for stereo reconstruction have led to extensive high-resolution and high-quality datasets. The broad public interest, raised by free online mapping services such as Google Earth has further boosted the development in these fields. This poses new challenges for computer graphics as now data from all over the planet is available with details up to 10cm in urban areas. For sites of interest, there even exist huge raw point clouds with spatial resolution in the range of single centimeters.

Considering the resulting datasets, we face the problem how to process the tremendous amount of data to make it accessible to the user. Realtime rendering methods together with appropriate navigation and interaction metaphors are required.

In order to clarify the topics of this thesis, I shortly introduce the general setup. The coarsest perspective shows three main processes of a typical data flow: acquisition, modeling, and interactive rendering.

Acquisition From the real world, different sensors acquire sensor data. The employed sensors are important as they define the quality and modality of the data. Thus I focus on data from optical sensors for photo-realistic colors and from range finding sensors for accurate geometry.

Modeling Sensor-data is post-processed into different types of standardized data products. Point clouds are very close to the original sensor data as they can be directly derived from depth images, a typical product of laser scanners or dense matchers. 2.5d digital surface models (DSMs) are derived by resampling the data into an ortho-rectified raster image. Orthotextures are similarly derived from image data by reprojecting the images onto the surface

model. Digital terrain models (DTMs) are more reduced data products which cover only the bare earth surface and no buildings, trees and the like. Although there are loads of other models, I focus on the mentioned raw models, which are derived by fully-automatic processing and supply the most details.

Interactive rendering The user navigates the camera and the model is rendered into an image showing the data from the chosen perspective. To convey the impression of a smooth camera flight, it is essential to render in realtime (i.e. more than 25 images per seconds).

Ideally, we want to present the user a virtual world which resembles the real world as closely as possible. Moreover, the user profits from task-specific additional informations (semantics)—most commonly the ones provided by geographic information systems (GIS)—and wants to interact on the basis of these informations (e.g. click on a building to find out its function and address, or fly to a specified destination).

So, this thesis deals with problems related to the realtime rendering of and the interaction with highly detailed landscape and city models.

1.1 Challenges

The challenges consist in the combination of the following aspects:

- high detailed datasets
- out-of-core datasets
- automatic processing
- efficient processing
- high quality rendering
- realtime rendering
- semantic enrichment
- semantic interaction

One major challenge in this context is the development of techniques that allow adapting the complexity of the rendered model to the output. From the original gigabytes of model data only roughly a million of colored pixels need to be computed in a few milliseconds. Fortunately, in perspective views of a scene,

with increasing distance to the observer, parts of the model can be represented with less accuracy without loss of quality. Based on this observation, level of detail (LOD) techniques were invented that represent the data at multiple resolutions and are able to adapt the model to the given perspective.

So the common basic approach is to simplify model parts and store them so that they can be rendered instead, depending on the distance to the camera. By culling against the camera frustum and choosing appropriate resolutions, the rendering model continuously adapts to the given perspective and only covers a fraction of the data. A major issue in the design of such an LOD scheme is how well the method scales with the complexity of the data. Especially the overheads introduced by using the LODs for rendering, but also the cost of the preprocessing must be considered.

Regarding the inherent complexity, there is already a fundamental difference between rendering huge terrain data sets which are modeled as DTM and rendering even comparatively small DSMs. Given a data accuracy and density in the range below meters yields a lot of complexity in otherwise harmless (i.e. flat) data sets. The reason for this is the disproportionate distribution of features to scales. On a 20m scale only hills, rivers, shores and mountains dominate the complexity, whereas on a single meter DSM even in flat regions every tree, bush, car, building etc. leaves its high-frequency footprint. As a result of such details a visualization of a single city on a meter scale can be more demanding regarding the level-of-detail scheme than visualizing planet earth on a 20–30 meter basis. Moreover, the representation of the surface as 2.5d is less appropriate for high detail scales than it is for classical terrain models.

Rendering digital surface models. One problem studied in the thesis is how to design a scalable LOD approach that considers this observation and scales starting with simple DTMs up to complex DSMs. I started considering digital surface models given as rastered height data that can easily be triangulated, and pursued the idea of combining a quadtree based hierarchy (1) with accurate mesh approximations (2) of the model.

Ad 1) In order to come up with a view-dependent model efficiently, model part approximations are precomputed in a hierarchy with different accuracies and then recombined at run-time. The quadtree hierarchy seems suitable to this problem because coarser approximations are built for larger parts such that the resulting screen size of a model part is more or less constant.

Ad 2) The basic approach for approximating the model is first to come up with a distance metric that yields accurate error bounds between model and simplified model and that can be computed efficiently. While the Hausdorff distance metric between meshes allows for accurate screen space errors, it is very complex to

compute. Therefore, I estimate the distance between an approximation and the original data by successively measuring against already computed approximations and accumulating the resulting distances.

Rendering 3d point clouds. If the underlying model consists of raw 3d point clouds that cannot be easily triangulated, we face another problem. However, the points are usually not arbitrarily distributed, but reflect the real world structre. Especially in the context of city models planar features prevail, such that polygonal models seem more appropriate. Therefore, I examined how these inherent structures can be exploited for efficient and high quality rendering of point clouds. The basic idea of my approach is to detect planar regions and represent them as planes. In order to maintain all high-frequency details I keep the original points wherever no planes are found, resulting in a hybrid point-polygon model. Regarding the LOD scheme, the quadtree structure used to organize the 2.5d data is replaced by an octree which leads to a hierarchical structure that can be applied to arbitrary geometry.

Semantically improved modeling. I also address the problem that, especially in the presence of building models, often purely geometric approximations lead to unintuitive results, as they do not respect the characteristic features of the buildings like orthogonal walls. E.g. a box is an intuitive coarse approximation of a house, whereas geometric approximation results in tent-like structures. In this context I propose the use of additional constraints, which inhibit model changes that alter characteristic features. To this end, I first detect planar structures in the input model and then label their vertices accordingly. Based on the topological relations of these labels, model changes are encouraged or disallowed.

Semantic interaction. Finally, a general problem concerning the interaction with models is the discrepancy between the representation used for rendering and the semantic models used for domain specific applications. While a building may consist of thousands of polygons or points, the semantically interesting entities like walls, windows, stories or balconies are potentially independent of the underlying geometric primitives. This fact hinders intuitive interaction with the model. Even picking of meaningful parts is not possible at all in a point cloud or triangle soup. While in 2d mapping this is easily solved by using layers for the semantic model so that the desired information is projected onto the map, extending this approach to 3d is non-trivial. I address this problem by introducing proxy geometry for the semantic parts that are invisible to the user, but still allow addressing and picking of these parts.

1.2 Contributions

Scalable Compression and Rendering of Textured Terrain Data. In [Wahl, Massing, Degener, Guthe, and Klein, 2004, chapter 2], I present a method for the realtime rendering of high-detail digital terrain and surface models along with orthorectified aerial imagery. It basically consists of two stages:

1. In the preprocessing stage, the input data is cut into tiles, which are subsequently approximated and recombined to larger tiles of the next coarser LOD, thus forming a quadtree hierarchy in a bottom-up manner. I define the simplification of the geometry tiles with edge-collapse operations and how the Hausdorff distance metric is employed to assess tight error bounds efficiently. Then I discuss the approximation of texture tiles and conclude the preprocessing with compression and serialization of the data.

2. In the rendering stage, I build a view-dependent sparse quadtree in a top-down fashion, covering only the visible regions with the required LOD. The tile data associated with the leaf nodes of the quadtree represent the rendering model.

I further give technical details how to fit together neighbouring tiles of different levels, and how to cache and prefetch tile data.

My experiments show that the combination of a quadtree hierarchy, which ideally suits the needs of multi-resolution modeling of 2.5d data, with arbitrarily triangulated textured tiles is a very efficient realtime rendering method. The mesh approximation with Hausdorff distance control used for the geometric tiles guarantees pixel-true geometric details and smooth, almost unnoticable LOD transitions. Moreover, the guaranteed accuracy is used to fill potential gaps between representations of different LODs and enable an efficient occlusion culling. An important result is the scalability of the presented approach which means that on the one hand, the complexity of the necessary preprocessing remains manageable and on the other hand, the complexity of the view dependent models extracted during rendering is only increasing logarithmically with the size of finer or larger models. Effectively, raw digital surface models of vast landscapes and whole cities can be rendered in realtime with full visual detail.

Identifying Planes in Point Clouds for Efficient Hybrid Rendering. When it comes to city data, 2.5d modeling is less appropriate, as especially facades are not well represented. Apart from the missing radiometric information, which can be partially solved by texturing the facades based on oblique or terrestrial imagery, the geometric details of cities that are hidden under roofs, balconies or trees need to be

represented. So the next step is to use data from terrestrial sensors. Consequently, in [Wahl, Guthe, and Klein, 2005, chapter 3] I cover realtime rendering of raw city data, but with a focus on 3d point clouds. I present an approach which is based on an octree hierarchy of cells, analogously to the just described tile quadtree of [Wahl et al., 2004]. Instead of geometric simplification, I employ known point cloud hierarchies and exploit the fact that planar regions can be rendered more efficiently as polygons.

The key to my hybrid point-polygon representation is the efficient search for point sets which are well representable as rectangular patches. These point sets are identified using a novel RANSAC-based approach with a cost function that favors large, compact, and densely covered patches. Texturing these patches with the original point colors and with a transparency indicator makes them a far less complex representation of the corresponding point set. I further augment these patches with the remaining points, which results in my hybrid point-polygon representation. The polygons of this hybrid representation can be rendered far more efficiently than the original points. It has the additional advantage that color information can be encoded using state-of-the-art image compression, which makes the hybrid model even more compact than the point cloud. I combine this representation with an octree hierarchy to arrive at an LOD scheme in which the substitution of point sets by textured patches replaces the simplification step. The resolution of patch textures and cells is carefully chosen so that the accuracy of the rendering model is exact to single pixels.

As my experiments on different huge point clouds demonstrate, the rendering of this hybrid model is an order of magnitude faster than purely point-based methods.

From Detailed Digital Surface Models to City Models Using Constrained Simplification. While the above-mentioned point cloud rendering solution of [Wahl et al., 2005] only focussed on the perfomance benefits of structural information in building datasets, the constrained simplification approach of [Wahl, Schnabel, and Klein, 2008, chapter 4] utilizes the structural information to arrive at model abstractions. This addresses the problem described in the paragraph *semantically improved modeling* (p. 4).

Most classical model reconstructions try to match prototype models to the data and thus only detect objects which are covered by these prototypes. The basic idea of my novel approach is to start from a polygonal simplification and add constraints in order to maintain important building features. Technically, I use my plane detection method in order to retrieve structural hints from raw DSM data. These structural hints are then used to define additional non-metric constraints based on a virtual topology, which are integrated into the purely distance-based edge collapse simplification of [Wahl et al., 2004].

The experiments show that the quality of the thus constrained simplification is much better than pure geometric approximation, as the detected structures remain well represented. In contrast to prototype methods, all features of the original dataset, also very untypical ones, are maintained.

Out-of-core Topologically Constrained Simplification for City Modeling from Digital Surface Models. In order to improve some typical artefacts of the original approach and extend the method to out-of-core data, I introduce a refined version in [Möser, Wahl, and Klein, 2009, chapter 5]. Additional topological constraints cover cases in which important parts of the topology are not detected. Moreover, I introduce a topological filtering step which avoids inconsistencies between actual and virtual topology. Last but not least, a parallelized processing scheme is introduced, which makes the method scalable and thus applicable to arbitrary large input data.

Experimental results with a huge high-resolution dataset of downtown Berlin demonstrate the quality and efficiency of the approach.

Towards Semantic Interaction in High-detail Realtime Terrain and City Visualization. The last topic of my thesis, presented in [Wahl and Klein, 2007, chapter 6], deals with the problem of semantic interaction in the context of high-detail 3d models. I begin with a review of scalable rendering techniques as those described in chapters 2 and 3 and term the employed model *rendering model* (least common denominator of LOD techniques). Then I motivate why one should refrain from coupling semantic metadata to the rendering model directly and introduce an *interaction model* as a means to implicitly map semantics. My key observation is that it is essential to separate semantic models from the 3d landscape and city models. This separation allows the user to choose freely from the semantic models and their LOD independent of the photorealistic rendering. Although this is similar to the way thematic layers are used along with 2d maps, the extension to 3d is not trivial as one has to consider occlusion in the scene. In my approach, I use proxy volume geometries to carry the semantic information into the scene, together with a hardware-supported volumetric intersection implementation to apply it to the rendering.

I illustrate my method with an efficient implementation of picking and highlighting of semantic parts in a large 3d dataset of Berlin, without touching the rendering model.

1.3 Publications

The main material of this thesis has already been published in conference proceedings and journals. Here is the list of relevant publications in the order of appearance in the following chapters:

ROLAND WAHL, MANUEL MASSING, PATRICK DEGENER, MICHAEL GUTHE, AND REINHARD KLEIN. Scalable compression and rendering of textured terrain data. *Journal of WSCG*, 12(3):521–528, February 2004. ISSN 1213-6972. 5, 6, 11, 75, 76

ROLAND WAHL, MICHAEL GUTHE, AND REINHARD KLEIN. Identifying planes in point-clouds for efficient hybrid rendering. In *The 13th Pacific Conference on Computer Graphics and Applications*, October 2005. 6, 25, 77

ROLAND WAHL, RUWEN SCHNABEL, AND REINHARD KLEIN. From detailed digital surface models to city models using constrained simplification. *Photogrammetrie, Fernerkundung, Geoinformation (PFG)*, 3:207–215, July 2008. 6, 41, 53, 55, 56, 58

SEBASTIAN MÖSER, ROLAND WAHL, AND REINHARD KLEIN. Out-of-core topologically constrained simplification for city modeling from digital surface models. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-5/W1, February 2009. ISSN 1682-1777. 7, 53

ROLAND WAHL AND REINHARD KLEIN. Towards semantic interaction in high-detail realtime terrain and city visualization. In U. Stilla, H. Mayer, F. Rottensteiner, C. Heipke, and S. Hinz, editors, *PIA07: Photogrammetric Image Analysis*, number XXXVI (3/W49A) in International Archives of Photogrammetry and Remote Sensing, pages 179–184. September 2007. ISBN 978-80-223-2292-8. 7, 71

Here I list, in chronological order, co-authored papers which are closely related to this thesis, but not included:

GERHARD H. BENDELS, PATRICK DEGENER, ROLAND WAHL, MARCEL KÖRTGEN, AND REINHARD KLEIN. Image-based registration of 3d-range data using feature surface elements. In Y. Chrysanthou, K. Cain, N. Silberman, and Franco Niccolucci, editors, *The 5th International Symposium on Virtual*

Reality, Archaeology and Cultural Heritage (VAST 2004), pages 115–124. Eurographics, December 2004. ISBN 3-905673-18-5.

RUWEN SCHNABEL, ROLAND WAHL, AND REINHARD KLEIN. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007a. 45, 47, 89

RUWEN SCHNABEL, ROLAND WAHL, AND REINHARD KLEIN. RANSAC based out-of-core point-cloud shape detection for city-modeling. *Schriftenreihe des DVW, Terrestrisches Laser-Scanning (TLS 2007)*, December 2007b. 56, 89

RUWEN SCHNABEL, RAOUL WESSEL, ROLAND WAHL, AND REINHARD KLEIN. Shape recognition in 3d point-clouds. In V. Skala, editor, *The 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2008*. UNION Agency-Science Press, February 2008. ISBN 978-80-86943-15-2. 48, 89

SEBASTIAN MÖSER, PATRICK DEGENER, ROLAND WAHL, AND REINHARD KLEIN. Context aware terrain visualization for wayfinding and navigation. *Computer Graphics Forum*, 27(7):1853–1860, October 2008. 91

H. TAUBENBÖCK, N. GOSEBERG, N. SETIADI, G. LÄMMEL, F. MODER, M. OCZIPKA, H. KLÜPFEL, ROLAND WAHL, T. SCHLURMANN, G. STRUNZ, J. BIRKMANN, K. NAGEL, F. SIEGERT, F. LEHMANN, S. DECH, ALEXANDER GRESS, AND REINHARD KLEIN. "Last-mile" preparation for a potential disaster – interdisciplinary approach towards tsunami early warning and an evacuation information system for the coastal city of Padang, Indonesia. *Natural Hazards and Earth System Science*, 9(4):1509–1528, August 2009. ISSN 1561-8633. 89

CHAPTER 2

SCALABLE COMPRESSION AND RENDERING OF TEXTURED TERRAIN DATA

Abstract

Several sophisticated methods are available for efficient rendering of out-of-core terrain data sets. For huge data sets the use of preprocessed tiles has proven to be more efficient than continuous levels of detail, since in the latter case the screen space error has to be verified for individual triangles. There are some prevailing problems of these approaches: i) the partitioning and simplification of the original data set and ii) the accurate rendering of these data sets. Current approaches still trade the approximation error in image space for increased frame rates.

To overcome these problems we propose a data structure and LOD scheme. These enable the real-time rendering of out-of-core data sets while guaranteeing geometric and texture accuracy of one pixel between original and rendered mesh in image space. To accomplish this, we utilize novel scalable techniques for integrated simplification, compression, and rendering. The combination of these techniques with impostors and occlusion culling yields a truly output sensitive algorithm for terrain data sets. We demonstrate the potential of our approach by presenting results for several terrain data sets with sizes up to 16k $\times$ 16k. The results show the unprecedented fidelity of the visualization, which is maintained even during real-time exploration of the data sets.

This chapter corresponds to the article [Wahl et al., 2004].

Keywords: Terrain rendering, level of detail, out-of-core rendering, compression

2.1 Introduction

Rendering of textured terrain models has become a widely used technique in the field of GIS applications. Due to the mere size of the data sets, out-of-core techniques must be used to process and visualize such models. Sampling the area of

the United States of about 9.2M km^2 with a sampling rate of 10 meters would result in a data set of about 300k $\times$ 300k height values. In most cases corresponding texture data is sampled at an even higher resolution. In urban areas sampling rates of 25 cm are common.

To achieve real time rendering without sacrificing accuracy, several aspects have to be considered. On one hand, to exploit the full performance of current GPUs, transmission of large data chunks is advantageous. On the other hand, no unnecessary data should be submitted, since bandwidth and I/O are often the bottleneck of current graphics systems. Furthermore, with the growing GPU power the management of fine-grained LODs on the CPU becomes more and more the limiting factor, and in many rendering applications the GPU is not working at full capacity.

A high-performance terrain rendering system should comprise the following characteristics:

- represent the input data faithfully
- allow for output sensitive rendering, in order to retain scalability (i.e. readily support LODs, occlusion culling, impostors)
- submit and process textures and geometry with adequate granularity to take advantage of GPUs, without taxing the CPU.
- allow for compact storage and on-the-fly decompression of textures and geometry to minimize bus bandwidth and storage requirements.
- local accessibility of geometry and textures without global interdependency, in order to maximize concurrency and to avoid management overhead.

Our method subdivides the geometry as well as the associated textures into equally sized blocks, which we refer to as tiles, and organizes them in a quadtree hierarchy. Tiles from coarser levels correspond to large areas, those from fine levels to small areas. Each geometry tile in the quadtree is represented by a triangulated irregular network (TIN). The vertices are placed on a local regular grid, which has constant resolution for all tiles of the hierarchy. Likewise, textures are stored with constant resolution.

Furthermore, for each tile in our model a guaranteed error bound is available. The approximation error doubles from level to level and is therefore a constant ratio of the tile extent. In contrast to common multi-resolution meshes, which start with a global, coarse approximation of the triangle mesh and decide on a per-triangle basis if further subdivision is necessary, we restrict ourselves to one decision per quadtree cell. This means that each geometry tile only holds one

precomputed triangulation, whose connectivity is stored using state of the art compression algorithms. During rendering we perform view-frustum and occlusion culling for the quadtree nodes, which are represented by bounding boxes as long as the geometry is not needed. The accuracy guarantee for the TINs is used to restrict the screen space error to be at most one pixel.

With these techniques we are able to render the simplified data with an image fidelity equal to a rendering of the full-resolution dataset in real time, even if the input data becomes arbitrarily large.

In the next section of this paper, we take a look at related work. In section 2.3 we give an outline of our algorithm. Section 2.4 describes how our discrete-LOD model is created in a preprocessing step and section 2.5 shows how we perform the rendering of our data structure. Then we show some results we deduced from real data sets. Section 2.7 concludes the paper.

2.2 Related work

Fast rendering of terrain datasets with viewpoint adaptive resolution is an active area of research. After the initial approaches by [Gross et al., 1995; Puppo, 1996; Lindstrom et al., 1996], many different data structures have been proposed. Since giving a complete overview is beyond the scope of this paper, we refer to recent surveys [Lindstrom and Pascucci, 2002; Pajarola, 2002] and only discuss the approaches most closely related to our work.

Considering existing approaches for the efficient processing and display of terrain datasets, one can differentiate between two main classes. The first class consists of approaches that employ regular, hierarchical structures to represent the terrain, whereas approaches of the second class are characterized by the use of more general, mainly unconstrained triangulations.

The most established methods of the first class make use of triangle bin-/quad-trees [Lindstrom et al., 1996; Duchaineau et al., 1997; Cline and Egbert, 2001], restricted quadtrees [Pajarola, 1998; Gerstner, 2003], RTINs [Evans et al., 2001] and edge bisections [Lindstrom and Pascucci, 2002]. These structures facilitate compact storage due to their regularity, as topology and geometry information is implicitly defined.

Approaches of the second class use less constrained triangulations. They include data structures like Multi-Triangulations [Puppo, 1996], adaptive merge trees [Xia and Varshney, 1996], hypertriangulations [Cignoni et al., 1997] and the adaptation of Progressive Meshes [Hoppe, 1996] to view-dependent terrain rendering [Hoppe, 1998]. As proven by Evans et al. [2001], TINs are able to reduce the number of necessary triangles by an order of magnitude compared to regular triangulations since they adapt much better to high frequency variations.

However, in order to capture irregular refinement or simplification operations and connectivity, a more complex data structure is needed. To alleviate these drawbacks, either Delaunay triangulations [De Floriani and Puppo, 1992; Rabinovich and Gotsman, 1997] or a modified quadtree structure have been used to represent irregular point sets [Pajarola et al., 2002].

Since all adaptive mesh generation techniques spend considerable computation time to generate the view-dependent triangulation, the extraction of a mesh with full screen-space accuracy is often not feasible in real-time applications. Many authors have proposed techniques to reduce the popping artifacts due to the insufficient triangle count [Cohen-Or and Levanoni, 1996; Hoppe, 1998] or to amortize the construction cost over multiple frames [Duchaineau et al., 1997; Hoppe, 1997; Lindstrom et al., 1996]. Another approach is to reduce the per-triangle computation cost by assembling pre-computed terrain patches during run-time to shift the bottleneck from the CPU to the GPU like the RUSTiC [Pomeranz, 2000] and CABTT [Levenberg, 2002] data structures. These methods were further refined, by representing clusters with TINs in a quadtree [Klein and Schilling, 2001] or bintree domain [Cignoni et al., 2003a]. To incorporate textures into the above mentioned hierarchies, the LOD management can be either decoupled from the geometry (e.g. the SGI clip-mapping extension and the 3Dlabs Virtual Textures), which requires special hardware, or they can be handled by explicitly cutting them into tiles and arranging them into a pyramidal data structure [Döllner et al., 2000]. However, this leads to severe limitations on the geometry refinement system, since corresponding geometry has to be clipped to texture tile domains.

2.3 Overview

Our method consists of a separate preprocessing stage and the actual rendering stage. A typical input dataset for the preprocessing consists of a digital elevation model (DEM) of the terrain and associated texture maps (e.g. orthophotography).

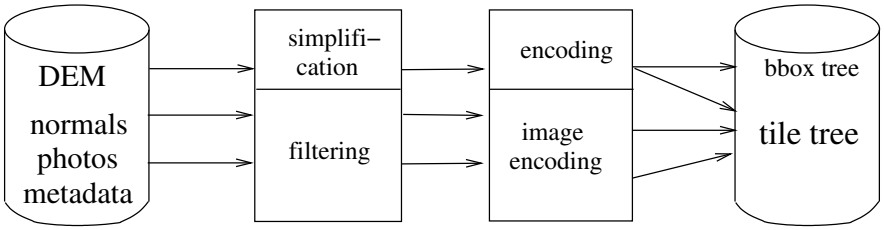

Figure 2.1: The preprocessing stage.

If desired, a map of surface normals (normal map) can be extracted from the DEM and processed in the same way as the textures. As detailed in the following

section, the preprocessing (fig. 2.1) recursively builds a LOD hierarchy of tiles (tile tree) through geometry simplification or texture filtering. Finally all resulting tiles are specifically encoded and stored. During geometry encoding, a separate bounding box hierarchy is extracted.

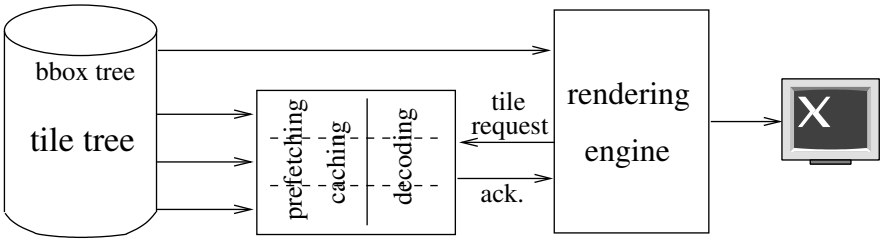

Figure 2.2: The rendering stage.

Rendering is essentially parallelized among two threads. The main thread selects cells for rendering by considering their visibility and detail. An additional caching thread performs the asynchronous retrieval of associated cell data (e.g. geometry and texture maps). Once all pending requests are completed, the rendering thread hands over the cell data to the graphics hardware. In order to avoid bursts of high workload, the caching thread can also perform prefetching of tiles based on the history of requests or a prediction of the camera path.

Since all operations are handled on a per-tile basis, and no interdependencies among tiles exist, this approach allows for very flexible compression and prefetching schemes.

Therefore, this architecture is able to handle huge terrains, including textures and normal maps. As will be shown in section 2.5, the number of tiles to be rendered is generally constant. As a consequence, the frame rate is not limited by the amount of input data, but only depends on the complexity of the visible data and on the available graphics hardware.

2.4 Tile tree construction

In this section, we describe how the geometry is processed into a multiresolution data structure, which we call the *tile tree*. Basically, the tile tree imposes a quadtree hierarchy on a set of tiles built from the input geometry and textures. The object space error is bounded throughout the whole pipeline.

The tile tree root holds geometry and texture tiles that cover the whole domain of the dataset, and children partition their parents' domain into equally sized quarters. Texture tiles at the leaves are initialized with the input texture data. Tiles on higher levels are then assembled from their children and downsampled by a

factor of 2, that is, the texture resolution remains constant for all tile tree levels. Analogously to the texture sub-sampling process, we partition the input mesh into geometry tiles, which are stored at the tile tree leaves. Geometry tiles on higher levels are built by approximating the input mesh with half the accuracy of their children. We use the symmetric Hausdorff Distances [Klein et al., 1996] between two meshes as a measure of their approximation accuracy. Both texture and geometry tiles are discretized and compressed before storage.

2.4.1 Error Bounds

All LOD algorithms strive to bound the screen space error, while rendering as few polygons as possible. In the general case, the screen space error ε depends on all viewing parameters: the eye position E, the viewing direction n_i, the field-of-view ϕ and the screen resolution r.

Since a precise calculation of the screen space error for a tentative simplification is too expensive, one approach is to establish only upper bounds on the object space error δ. The screen-space error can then be easily derived at runtime from the precomputed object space error. From intercept theorems, we have that $\varepsilon = \delta \cdot \cos(\alpha) \cdot d_i/d$ where $d_i = \cot(\phi) \cdot r/2$ and $d = (P - E) \cdot n_i$ (fig. 2.3).

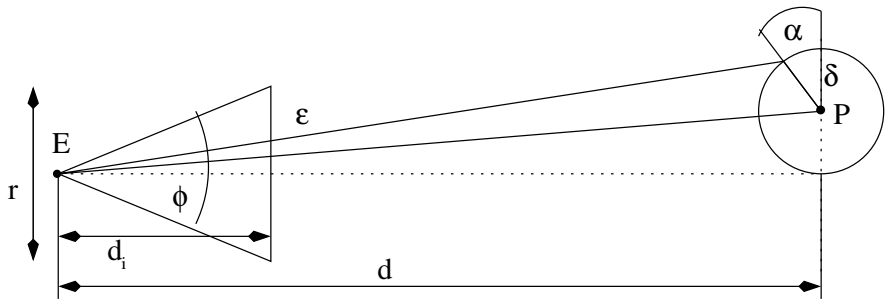

Figure 2.3: Relationship of errors depicted in 2D.

To further simplify the problem, the direction of the object-space error (i.e. α) is neglected and only its magnitude δ is regarded. This means that we do not consider the eye position, but only the distance of the observer. We do so for three reasons: First, considering the viewing direction does not save significant amounts of triangles, as Hoppe [1998] has pointed out. Secondly, we do not only want to reproduce the correct contours, but also the correct texture coordinates, which requires the object space error to be bounded isotropically anyway.[1] And finally the reduction of dimensions is exactly what we need to build discrete LODs without having too much redundancy in the data.

[1] Though the L^∞ metric would be sufficient in this case.

Consider a tile T with an associated bounding box B. When the object space error for this tile is known to be less than δ_T and we want to guarantee a screen space error below a threshold τ we can use this tile, whenever B lies fully behind a plane with normal n_i and distance $d_i \cdot \delta_T/\tau$.

This means that doubling the observer distance allows us to double the permitted object space error, while maintaining the same screen space error bound. Furthermore, this allows us to represent the geometry of the considered tile on a local grid of constant resolution, because the relative accuracy within a tile is also constant.

In comparison to a continuous view dependent approach (CVLOD), we render a larger number of triangles because the screen space error is overestimated in most places. If one considers an optimal CVLOD mesh, and the mesh complexity falls off quadratically with the permitted Hausdorff error $n \approx \frac{n_0}{\delta^2}$, the number of triangles would remain constant for a fixed viewing direction. In this case, the mesh complexity of our discrete LOD representation would exceed the CVLOD by at most a factor of 4 in the top-down view. When approaching from above, the average overhead would be $\int_1^2 x^2\,dx = \frac{7}{3}$, which is at the same time the maximum factor for a lateral view. Since looking from above is the simplest case for rendering (no overdraw, localized texture accesses), the over-estimated mesh complexity does not have a significant impact on performance, and is well worth the cost for the simple, low-cost mesh generation, and the flexibility and complete independence of the data tiles.

2.4.2 Simplification

The geometry simplification starts by splitting the DEM, which typically is given by a regularly sampled heightfield, into equally sized base level tiles (e.g. 129 $\times$ 129 samples each, with overlapping borders). Then, a reasonable triangulation (e.g. regular) is imposed on the height-samples, and a presimplification with error bound δ_{pre} is performed on this mesh. The pre-simplification is meant to accommodate the fact that the input is a regular grid with a given discretization error, so δ_{pre} will be about one half inter-pixel spacing, as this is the amount of uncertainty inherent in the data. These presimplified base-tile meshes are then stored at the leaves of the tile tree, and all subsequent error metrics refer to these meshes.

To make up a tile of the next tile-tree level l, four neighboring tiles are stitched together. The resulting mesh is then simplified to approximate the reference mesh with an error bound δ_l, which is chosen to guarantee an error against the base mesh of $2^l \cdot \tau$. The tile outlines are preserved, but simplifying the borders is allowed if the error implied in the neighboring tile also lies below δ_l. This is an important property, since otherwise the number of border triangles would explode on huge datasets. To avoid unbounded complexity of the reference mesh, which

would increase fourfold on every level using a naïve approach, we always measure the Hausdorff error against the penultimate simplification level. That way, the additional error already immanent in the reference can be conservatively estimated as $\frac{1}{4}$, so the overestimation adds up to $1 + \frac{1}{4} + \frac{1}{16} + \ldots \leq \frac{4}{3}$. In order to maintain the overall Hausdorff error bound, a conservative estimate of the rounding error committed during compression is subtracted from the permitted simplification error bound for a tile.

The simplification of a tile is highly local, since all measurements during simplification of a tile relate to the tile itself, one of its neighbors, or the corresponding reference tiles.

A parallelization of the simplification is straightforward and the algorithm scales well since the memory requirement for simplifying a tile is bound by a constant. One can even avoid the dependency on neighboring tiles completely if the permitted error along the affected borders is restricted to half the magnitude of the allowed simplification error. That way, the difference between two neighbors is guaranteed to be less than the pixel-threshold, and resulting cracks can be handled as described in section 2.5.2.

2.4.3 Textures

Bounding the Hausdorff distance between the original and the simplified mesh guarantees the correct representation of contours for a given tolerance, but does not guarantee the correct coloring of the surface. In addition to conventional decal texture maps, we employ normal maps extracted from the input dataset. With normal maps, shading detail is preserved even in regions of coarse triangulation, which would otherwise be discarded by geometry-based shading (e.g. Gouraud shading). Of course, textures taken from photographs may already contain shaded and shadowed features, but nevertheless normal maps help to reveal the structure of the terrain, especially if additional moving light sources are used.

Since terrain is rather flat, the textures can be projected from above with sufficient accuracy, and the level-of-detail for a texture tile can be chosen in the same way as for geometry tiles. This way, we establish a one-to-one correspondency between texture- and geometry tiles. As already mentioned, texture maps are constructed bottom-up from the input data by downsampling, which basically means building a standard image pyramid on top of the underlying input image (e.g. by averaging 4 neighboring pixels). This also holds for the normal maps, since the defect in length accounts for the roughness of the surface.

During rendering, we apply anisotropic filtering instead of a mip-mapping scheme. This does not only enhance rendering quality, but improves locality because the level of filtering is chosen by the maximum partial derivative.

2.4.4 Compression

One major drawback of using TINs compared to quadtree triangulations [Lindstrom et al., 1996; Duchaineau et al., 1997] is that the connectivity is no longer implicit. Fortunately, there are very efficient methods for coding and decoding connectivity [Gumhold and Straßer, 1998; Rossignac, 1999] which rarely use more than 4 bits per vertex. Regarding the coordinates we factor the information into a bounding box – whose xy-coordinates are implicit and whose minimum and maximum elevation are explicitly stored in a separate structure – and a local grid address. As already mentioned before, the grid inside a tile's bounding box may have a constant resolution independent of the level. If we use 129×129-tiles for geometry the inner vertices can be addressed with $14 + \lceil \log h \rceil$ bits, where h denotes the height of the bounding box measured in level-dependent units. Typically one will further discretize the bounding box axes with a constant number of bits, so that the rounding procedure does not dominate the Hausdorff error and thereby increase the triangle count. However, all in all the number of bits per vertex even in mountainous terrain rarely exceeds 32 bits per vertex. For experimental results see section 2.6.

To compress our textures and normal maps, we employ standard compression algorithms such as S3TC and JPEG. S3TC compressed textures offer the great advantage that decoding is implemented on most standard graphics hardware, thus sparing the CPU from decompression. Moreover, they reduce bandwidth and texture memory requirements, as the textures may reside in memory in their compressed form. The main disadvantage of S3TC are block artifacts, which are especially noticeable with normal maps, and the minimal level of control over the compressed image quality. JPEG offers better compression ratios and therefore lessens the load on the I/O, but needs to be decoded in the CPU, which can become a bottleneck. Also, the artifacts are more disturbing. Later standards as JPEG2000 featuring wavelet-codecs are desirable, especially for their inherent support of texture hierarchies, but need to be hardware supported to achieve similar efficiency. Since the tiles can be encoded independently, it is easy to mix different encoding schemes or use lossless formats whenever the signal to noise ratio falls below a certain threshold, but then one needs to take care that the tiles' borders do not become visible due to quality changes.

2.5 Rendering

We divide the rendering into two stages, the update stage and the cell rendering stage. During the update stage, the CPU traverses the bounding box hierarchy depth-first and decides which tiles need to be rendered.

2.5.1 Quadtree Update

The update stage can be implemented using a simple top-down traversal of the quadtree hierarchy. Each tile visited in that manner is first checked against the viewing frustum. If the tile's bounding box lies completely outside of the frustum, descent can stop. Otherwise, we need to decide whether the tile in question satisfies our error bound. Since this object-space error bound is fixed throughout a whole quadtree level, and there are no constraints regarding the LOD-difference of neighboring tiles, the selection of an appropriate level of detail is straightforward. All tiles which may be rendered with a LOD of d or coarser lie completely behind a virtual plane which is a shifted copy of the image plane at distance 2^d. If the tile is found to have sufficient detail, it is considered for rendering and, if necessary, geometry, texture and normal map for the tile are requested from the cache. If the tile LOD is not sufficient, we continue our descent.

In a second stage, the tiles found to be visible are rendered.

2.5.2 Repairing Cracks

In case the LOD of two neighboring tiles differ, it is not sufficient to simply render the geometry. Even though the geometric errors between the tiles would fall below the pixel projection threshold, small cracks may become visible due to discretization in the rasterizer stage. But since the cracks are under screen space error control, there is no need to avoid them, they only need to be filled with the correct color. This is achieved by attaching a triangle strip along the border that reaches down the equivalent of one pixel. In this way, the holes are shaded consistently with the borders.

2.5.3 Caching & Prefetching

Even in single-processor system, the CPU, GPU and IO subsystem can work more or less concurrently. To maintain and support such parallelism between the CPU and IO-subsystem, we employ caching and prefetching during the update and rendering stages. During the update stage, the caching thread receives requests from the rendering thread and fulfills them asynchronously. The threads are then synchronized to ensure completion of pending requests. While the terrain is rendered, which is a task independent of the IO subsystem, we can perform node prefetches based on the previously requested nodes and the estimated camera motion. These prefetched nodes are stored in a cache and will in many cases accelerate geometry requests in subsequent update stages.

2.5.4 Output Sensitivity

Since the rendering output always consists of a constant number of colored pixels, achieving output sensitivity is a very demanding task. With our basic LOD algorithm, we achieve, that the per frame complexity is within $O(\log n)$ (i.e. the number of visible tiles per LOD as well as the tile complexity is bounded by a constant). In order to be output sensitive in this theoretic sense, the number of visible tiles has to decrease with growing distance such that its series converges, which basically means that there are only finitely many visible LODs. In fact there are several real world effects, that suggest that this is a feasible demand. Occlusion, earth curvature, atmosphere (fog) and limited flight speed (distant features need not to be redrawn every frame) help to decrease complexity if taken into account. In the following, we will discuss practical aspects of techniques such as occlusion culling and impostors which make use of these effects. The tile granularity combined with the associated object space errors offers advantages for both methods.

2.5.5 Occlusion Culling

During quadtree traversal we can ensure a front-to-back ordering, which enables us to perform per-cell occlusion by conservatively testing tiles against potential occluders.

One can do so by rendering potentially occluding geometry into the depth-buffer (while disregarding texture & color information) during quadtree traversal. Visibility tests on potentially visible cells can be performed by rendering an appropriate enclosure of the geometry and then testing if any pixels passed the depth test. As noticed by Lloyd and Egbert [2002], bounding boxes give satisfying results.

We are able to render the occluders with a greater pixel error than the cells whose visibility is to be determined. This is due to the guaranteed error bounds on our geometry, as the bounding boxes can easily be scaled to compensate for the error introduced by using the coarse occluding geometry. If one accounts for discretization errors it is also possible to reduce the resolution of the depth buffer, thus minimizing fillrate requirements.

2.5.6 Impostors

For a flight speed v there always exists a distance $d(v)$ so that the 3 dimensional effects within a tile or region at this distance are no longer noticeable for several frames. This fact can be exploited by rendering these tiles or regions into textures and project these textures on a quad (impostor) which replaces the geometry. The error is tracked and the impostor is invalidated if the error exceeds a threshold.

If one wants to guarantee a screen space error of one pixel, one has to sum up the errors which are made on the different stages for the impostors. For example if one renders the impostor during setup a certain pixel error is made, but then again during rendering the texture a resampling error is added which also takes account for the resolution of the impostor texture. Both of these errors need to be added to the geometric error which reflects the parallax not represented in the flat geometry.

2.6 Implementation & results

For our experiments we implemented a simplifier, a renderer and a coding/decoding module as described in section 2.3. The simplifier performed edge-collapses, which were generated and scheduled using error quadrics [Garland and Heckbert, 1997b]. For each proposed collapse the Hausdorff error was computed by calculating the point-triangle as well as the edge-edge distances using the domain as an indicator, which elements need to be checked against each other. Since the z-projection used for finding correspondencies in this approach does not necessarily yield the closest elements, we establish upper bounds on the error. In order to guarantee linear time-complexity, each collapse is either performed or deleted from the queue.

The rendering was performed on a PC with a 1.8 GHz Pentium 4 processor, 512MB RAM running Linux and a GeForce3 graphics card.

Dataset	Gridsize	Spacing	Time	Filesize	Input
Puget Sound	16k $\times$ 16k	10m	1h25	9.1MB	256M
Turtmann valley	3 $\times$ 4k $\times$ 4k	2m	1h12	7.8MB	48M
Westbank	$\sim$6k $\times$ 15k	10m	0h40	3.8MB	$\sim$90M
Grand canyon	2k $\times$ 4k	60m	0h02	0.2MB	8M

#Vertices in approximation with relative error					
Dataset	0.5	2	4	8	16
Puget Sound	2,698,445	298,271	93,266	47,355	14,921
Turtmann valley	2,014,045	284,462	67,108	26,558	5,969
Westbank	1,135,903	127,713	34,181	14,497	3,266
Grand canyon	65,495	6,868	1,681	654	61

Table 2.1: Geometry statistics of tested models.

The largest dataset visualized so far with our approach shows the Puget Sound area in Washington, U.S. The input heightmap consists of 16,385 $\times$ 16,385 height samples, with 10m inter-pixel spacing. Additionally, matching texture and normal

maps were created. The presimplified dataset, which comprises geometry, S3TC-compressed textures and normal maps, uses 371MB of storage, as opposed to over 1GB needed by the uncompressed heightmap and texture data. Figure 2.4 depicts framerates of a high-speed (5,400km/h), low-altitude flight over the Puget Sound dataset. Rendering was performed on a 768 $\times$ 576 screen with an error threshold $\tau < 1$ and full resolution normal and texture mapping.

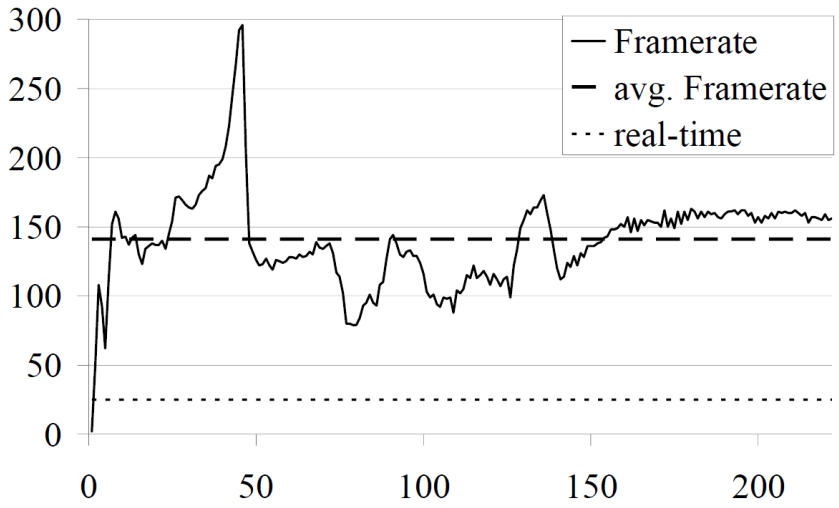

Figure 2.4: Frame rates for a Puget sound fly-over.

We were also able to visualize a complex dataset of the Turtmann valley in Switzerland (fig. 2.5) at high frame rates. The dataset features steep, mountainous parts of the alps at 2 meter resolution. It is actually a digital surface model, which means that even rocks, buildings and trees are present in the geometry. The data was cut into three slightly shifted 4k $\times$ 4k datasets and processed into three different tile trees. Note the flexibility of our approach, which easily integrated all three datasets into a single rendering process. We also implemented our terrain rendering engine on a 6-projector powerwall setup, where the isotropic error guarantee extends to accurate depth perception. Videos of the mentioned fly-overs can be downloaded at: `http://cg.cs.uni-bonn.de/project-pages/terrain`.

2.7 Conclusion & future work

We have seen that it pays to guarantee conservative Hausdorff error bounds. This enables us to render huge datasets with incredible detail which previous approaches would clearly fail to handle in real-time due to the high triangle complexity. We

Figure 2.5: Snapshot of Turtmann valley fly-over.

have shown that off-the-shelf hardware is powerful enough to render huge textured datasets, and are eager to explore the rendering capabilities of our new approach with even larger and more detailed datasets.

2.8 Acknowledgements

We like to thank the Jet Propulsion Laboratory for making their Landsat imagery available on the web for free as well as the Georgia Institute of Technology for the Puget Sound and Grand Canyon datasets. Special thanks to Prof. Dr. Richard Dikau from the Geomorphological and Environmental Research Group who made the Turtmann Valley data available to us.

CHAPTER 3

IDENTIFYING PLANES IN POINT CLOUDS FOR EFFICIENT HYBRID RENDERING

Abstract

We present a hybrid rendering technique for high-feature colored point clouds that achieves both, high performance and high quality. Planar subsets in the point cloud are identified to drastically reduce the number of vertices, thus saving transformation bandwidth at the cost of the much higher fill-rate. Moreover, when rendering the planes, the filtering is comparable to elaborate point-rendering methods but significantly faster since it is supported in hardware. This way we achieve at least a 5 times higher performance than simple point rendering and a 40 times higher than a splatting technique with comparable quality. The preprocessing performance is orders of magnitude faster than comparable high quality point cloud simplification techniques.

The plane detection is based on the random sample consensus (RANSAC) approach, which easily finds multiple structures without using the expensive Hough transform. Additionally, we use an octree in order to identify planar representations at different scales and accuracies for level-of-detail selection during rendering. The octree has the additional advantage of limiting the number of planar structures, thereby making their detection faster and more robust. Furthermore, the spatial subdivision facilitates handling out-of-core point clouds, both in preprocessing and rendering.

This chapter corresponds to the article [Wahl et al., 2005].

Keywords: hybrid rendering, point cloud, point rendering, plane detection, level-of-detail, out-of-core

3.1 Introduction

In the recent years, 3D scanners have become a common acquisition device. Since these scanners produce point clouds and rendering of point primitives is simple

Figure 3.1: Scanned Welfenschloss point cloud exhibiting high frequency material details.

and relatively fast, rendering of such point clouds has become an important area of research. Of course not only the geometry of an object is captured, but also color or other material properties. Furthermore, points are also a reasonable primitive for extremely detailed models. Whenever the triangles of a mesh mostly project to at most one pixel, rendering a point cloud is more efficient.

On current graphics hardware, the fill-rate is 10 to 20 times higher than the vertex transformation rate. Therefore, interactive rendering algorithms try to replace pixel-sized points by primitives covering multiple fragments. These can either be polygons or more complex point primitives like splats or surfels. However, if the scanned object is textured or features high frequency geometry, such a simplification of the model is not possible because it would remove important information. To preserve the appearance of a model for all levels of the hierarchy, a reduction operation (i.e. merging two close-by vertices) can only be performed if either the normal and color of the first vertex comply with those of the second one, or the vertices have a distance which is less than the allowed approximation

error. Since the first case is only true for smooth variations of normal and color on the object, the vertex distance roughly equals the approximation error for models with high frequency details. For pixel accurate rendering, this again leads to primitives which basically cover single pixels. Therefore, this approach, though successful for smooth objects, is not applicable for models with high frequency detail. In the latter case a higher rendering performance can only be achieved by using textured primitives as they can significantly reduce the number of vertices. However, existing simplification algorithms for textured models were designed for polygon meshes and cannot deal with point clouds.

In this paper, we present a novel method, where planar subsets of points are efficiently identified. Then the points are replaced by textured quadrilateral patches without establishing connectivity. This means that current texturing hardware can be exploited without performing a time-consuming meshing. Such a meshing is not only rather complex but also suffers from the need to triangulate small features, which again leads to many primitives. Instead of meshing, parts like ornaments or cavities which are not captured by the dominant planar surface are simply left empty (i.e. rendered as transparent) in the quad. The resulting holes are either filled by further quads that are coplanar to the features or with their original points.

To extract planes at different scales, we use an octree to decompose the model. This does not only improve the efficiency and robustness of the plane detection algorithm, but also allows a straightforward simplification even for out-of-core models. For octree generation, state-of-the-art out-of-core streaming compression file formats are well suited, since they decode the points locally [Isenburg and Gumhold, 2003]. After octree generation the effort per node is low and thus processing rates comparable to fast polygon simplification algorithms are achievable. Compared to recent high quality point cloud simplification techniques (e.g. [Wu and Kobbelt, 2004]) we achieve a reduction of the preprocessing time by a factor of about 50. During rendering the octree layout also supports efficient culling, LOD selection, and out-of-core rendering in a straightforward fashion.

3.2 Related Work

Recently much work focussed on point-based rendering and surface splatting and also the tradeoff between different rendering primitives was investigated and hybrid rendering techniques were developed. Due to the ever increasing size of acquired point clouds, exploiting out-of-core techniques for both model preparation and rendering becomes necessary.

Point-based Rendering. Points have first been proposed as universal rendering primitives by Levoy and Whitted [1985]. Instead of deriving a rendering algorithm

for each geometry representation, they propose to subdivide each representation into a sufficiently dense set of sample points. Since continuous (i.e. hole-free) images should be created by rendering a discrete set of surface samples, methods for closing these holes – e.g. by image-space reconstruction techniques [Grossman and Dally, 1998; Pfister et al., 2000] or by object-space re-sampling – were developed. Targeting at the efficient visualization of the models acquired during the Digital Michelangelo project [Levoy et al., 2000], Rusinkiewicz and Levoy [2000] proposed a hierarchical rendering method based on a pre-computed bounding-sphere hierarchy.

In contrast to this, surface splatting [Zwicker et al., 2001] renders splats, object-space disks or ellipses, instead of points only. In this case, the mutual overlap of splats in object-space guarantees a hole-free rendering in image space. Since rendering of inter-penetrating splats results in shading discontinuities, Zwicker et al. [2001] proposed a high quality anisotropic anti-aliasing method. Using pixel shaders allows the rasterization of elliptical splats by rendering just one vertex per splat. An implementation using circular object-space splats and two-pass Gaussian filtering was presented by Botsch and Kobbelt [2003], achieving a splat rate of 10M splats/sec. Since this necessitates level-of-detail methods for rendering, simplification algorithms for point clouds (e.g. [Pauly et al., 2002]) have been developed. In order to represent sharp features by point-sampled geometries, Pauly et al. [2003] proposed to clip splats against clipping lines defined in their local tangent frames. This representation can easily be rendered by integrating a per-pixel clipping test into the fragment shaders, as proposed by Zwicker et al. [2004] and Botsch et al. [2004]. Although the performance of splatting in combination with simplification is high for models containing larger smooth parts, it breaks down down at high frequency structure or color details.

Hybrid Rendering. A hybrid point/polygon-based representation of objects was first used by the POP rendering system [Chen and Nguyen, 2001], which uses polygons at the lowest level only and a point hierarchy on higher levels. Simultaneously a method for hybrid point polygon simplification based on edge collapse operations was introduced by Cohen et al. [2001]. In this approach points are generated according to the error metric and the size of the triangle. This algorithm however, allows a transition only from polygons to points and not vice versa, and therefore, the transition point has a high impact on the efficiency of the simplification. This was solved by Guthe et al. [2004], where the points are generated after hierarchical simplification and thus a transition from points to triangles was not necessary.

All of these methods have the drawback that they can only deal with triangle meshes which have to be generated from a given point cloud. A different approach

which starts with a point cloud representation of the model is PMR [Dey and Hudson, 2002]. The point cloud is simplified using a feature-based simplification algorithm and a triangulation of this point cloud is generated for display at higher resolutions afterwards. During rendering points or triangles are selected for display depending on their screen size. This approach adjusts the point/polygon balance to achieve maximum rendering performance, but the triangle are very small since they are generated from the simplified point cloud.

While these approaches allow a simple and flexible preprocessing and rendering of the models they all achieve a low performance for colored models with many small features, as they all attach information like color and normals at the vertices and thus hinders efficient simplification.

Impostors and Billboards. For high feature color models a drastic reduction of the number of rendering primitives can only be achieved by using textures. Among the earliest image-based approaches are static impostors, proposed by Maciel and Shirley [1995], which replace large parts of the geometry by a single textured polygon. The approaches of Schaufler and Stürzlinger [1996] and Shade et al. [1996] dynamically update the texture to match the current viewpoint. Later approaches aimed at improving parallax effects using layered impostors [Schaufler, 1998], layered depth images [Shade et al., 1998] or more complex, textured geometry [Sillion et al., 1997; Jeschke and Wimmer, 2002] which makes single impostors valid or acceptable for a larger set of viewpoints at the cost of increased texture, geometry and rendering complexity.

The recent approaches of Décoret et al. [2003] and Andújar et al. [2004] introduce and utilize the already described concept of billboard clouds. Unlike most previous methods this solution is view-independent making it very efficient for real-time rendering. Unfortunately, the presented methods generate billboard clouds which require much texture memory. Another view-independent approach is followed by Decaudin and Neyret [2004]: they sample objects into 3D textures which are efficiently rendered using volume visualization techniques. Although highly efficient rendering is possible, 3D textures require even larger amounts of texture memory.

Memory efficient billboard construction implies finding an optimal set of textured quads such that the appearance of the object is best preserved at minimal costs. In other words: efficient billboard clouds represent an optimized set of possibly overlapping clusters of geometry where each cluster is well approximated by a textured quad. Finding optimal clusters of geometry has been the topic of various publications, mainly based on simplification of connected triangular meshes (e.g. [Kalvin and Taylor, 1997; Sheffer et al., 1997]). A following publication of Inoue et al. [1999] introduced an ordering scheme for the merge operation based

on several criterions, but choosing appropriate weights is highly unintuitive. The memory problem becomes more critical, the more complex the texture data (e.g. for multi-texturing) is. Therefore, creating memory efficient billboard clouds were proposed by Meseth and Klein [2004] in the context of BTF-textured objects.

Out-of-core Rendering. For walkthrough applications an out-of-core rendering system [Varadhan and Manocha, 2002] was developed which combines level-of-detail and culling. An extension to hybrid point-polygon rendering was made to preserve the appearance of the model [Guthe et al., 2004].

For point clouds or meshes where the benefits of polygonal simplification are very minor, points with attached BRDF [Gobbetti and Marton, 2005] can also be used to represent coarser levels-of-detail. This however means that the rendering become transformation limited. Pajarola et al. [2004] proposed an out-of-core point rendering system based on a spatial subdivision hierarchy (e.g. an octree or median split). A LOD representation of the points is stored for each node of the hierarchy. But again, rendering is transformation limited due to the point primitives.

3.3 Preprocessing

To process out-of-core point clouds, we first stream through the entire model to construct the octree hierarchy which is later on also used for rendering. Inside this hierarchy we generate a level-of-detail representation for each node with a specific simplification tolerance ε. Depending on the desired granularity of culling and level-of-detail selection, the screen size s_{scr} of a node should be in a reasonable range. Therefore, ε has to be a constant fraction of the node size s. For a given screen space error tolerance ε_{scr} this constant is defined as $res = \frac{s_{scr}}{\varepsilon_{scr}}$. With this resolution we define a grid of $res \times res \times res$ cells inside each node and simplify the points using vertex clustering in each grid cell.

3.3.1 Plane Detection

The Random Sample Consensus (RANSAC) paradigm introduced by Fischler and Bolles [1981] is a general approach to fit a model to noisy data. The basic idea of this method is to compute the free parameters of the model from an adequate number of randomly selected samples. Then all samples vote whether they agree with the proposed hypothesis. This process is repeated until a sufficiently broad consensus is achieved.

Two major advantages of this approach are its ability to ignore outliers without explicit handling and the fact that it can be extended to extract multiple instances in a dataset. Especially in the domain of plane detection the methods based on

principal component analysis (PCA) fail in the presence of outliers or if multiple structures interfere.

As we want to apply RANSAC to detect multiple planes in point clouds, we roughly proceed as follows:

1. We choose a plane candidate by randomly drawing three samples from the point cloud.
2. The consensus on this candidate is measured.
3. The best candidate after a fixed number of iterations of the two above steps is taken.
4. If the consensus on the best candidate is high enough it is taken to be a plane and its conforming points are removed.
5. The whole process is repeated unless the best candidate fails. We then assume that no more planar structures are contained in the points.

Let us now analyse how well this algorithm performs in finding planarities: Consider a point cloud of n points sampled from k planes such that no point belongs to more than one plane. The probability p_s that 3 random samples belong to the same planar structure is then

$$p_s = \frac{\binom{n_1}{3} + \binom{n_2}{3} + \cdots + \binom{n_k}{3}}{\binom{n}{3}},$$

where n_i denotes the number of points belonging to the i-th plane. Obviously, the worst case is that all planes have an equal number of points, then the chance of finding a plane with 3 samples is:

$$p_s = \frac{k\binom{\frac{n}{k}}{3}}{\binom{n}{3}} = \frac{(\frac{n}{k}-1)(\frac{n}{k}-2)}{(n-1)(n-2)} = \frac{1}{k^2} \cdot \frac{(n-k)(n-2k)}{(n-1)(n-2)}$$

and therefore $p_s \approx 1/k^2$ is our likelihood of successfully detecting a plane in a single try whenever $k \ll n$. Now if we iterate the experiment l times the probability p_f of not finding any valid candidate plane evaluates to

$$p_f = (1 - p_s)^l$$

Taking logarithms and solving for l yields

$$l = \frac{\ln p_f}{\ln(1 - p_s)}.$$

Approximating the denominator using the Mercator series

$$\ln(1+x) = \sum_{i=1}^{\infty} \frac{-1^{i-1}}{i} x^i = x + O(x^2)$$

which holds for all $x \in (-1, 1]$ and plugging in the approximation of p_s tells us

$$l = \frac{\ln(p_f)}{-p_s - O(p_s^2)} \approx \ln(p_f)(-k^2) = \ln(1/p_f)k^2.$$

Therefore the number of necessary iterations depends logarithmically on the inverse failure rate and quadratically on the number of planes. If we want to find a plane with a failure rate of at most one in a million, we have to check $\ln(1\,000\,000) \cdot k^2 \approx 13.8k^2$ plane candidates. So obviously the success in finding planes is not governed by the imponderabilities of chance but dominated by the number of planar structures.

In the presence of outliers the number of useful candidates is reduced. If r_{out} denotes the rate of outliers in the data and assuming that an outlier in one of the three samples would make the candidate useless, p_s approximates $\frac{(1-r_{out})^3}{k^2}$. Although, the power of three in the numerator looks alarming, it only doubles the number of necessary steps even for 20% of outliers and has significantly less influence for smaller percentages.

Since we do not know the number of planes k in advance, the number l of reasonable iterations cannot be computed. But as we use an octree and look for planar structures on scales corresponding to the cell-size k is expected to vary only within a small range. Furthermore, the analysis above describes the worst-case in which all planes have the same amount of support. In practice you will often find a few dominating planes which are detected first and smaller structures are revealed when the large planes have been removed.

One key advantage of RANSAC over similar voting schemes like the Hough transform, which can also cope with outliers and noise, is that the measure of consensus for a candidate can be arbitrarily chosen. In our case we use this fact to directly establish a Hausdorff distance bound during detection combined with a compactness threshold.

$$c = \begin{cases} v & : \text{ if compact} \\ 0 & : \text{ else} \end{cases},$$

where v is the number of points within ε distance, i.e. the points that can be replaced by this plane.

To determine if a plane is sufficiently compact, we need to evaluate if replacing all close-by vertices with a textured quad improves the performance. Therefore,

we assume that a texel can be rendered φ times faster than a vertex. Since each quad requires four vertices to be transformed, this leads to the following condition:

$$t < \varphi(v - 4),$$

where t is the number of texel (i.e. the size of the quad). Note that this is a compactness condition and e.g. for $\varphi = 8$ already groups of five vertices can be replaced by a 2×3 texel quad while improving the performance.

3.3.2 Texture Generation

In the next step the texture for the found quads has to be generated. For this purpose, we use the approach of Décoret et al. [2003] and simply render the portion of the point cloud contained in the current node with appropriate clipping planes to project all points within ε-distance onto the plane. The resolution of the texture is chosen in such a way that the distance between two texels is the approximation error ε of the according octree node.

To maintain locality and facilitate out-of-core rendering, the textures for all nodes of the same grandparent are packed into a separate texture atlas. Since the packing problem is known to be NP-complete, we use a simple heuristic to pack the rectangular textures. First the bounding boxes are sorted by their height and then consecutively inserted row-by-row into the texture atlas (see Figure 3.2).

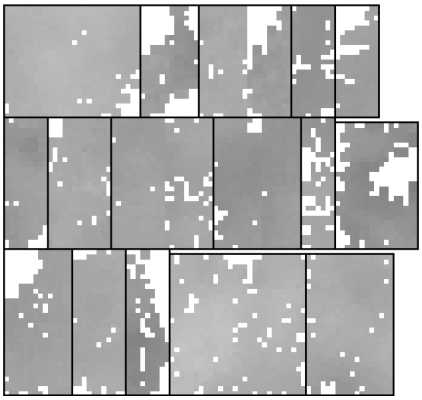

Figure 3.2: Texture packing.

3.3.3 Compression

For out-of-core models apart from efficient rendering representations, compression is also a desirable feature. For our hybrid representations we need to store the

remaining point cloud, the quad vertices with texture coordinates and the texture atlas. For the remaining points the position is quantized using the node's bounding box, which needs 6 bits in each dimension and the color takes 24 bits which sums up to a total of 42 bits per point. The corner vertices of the quads, however, should not be quantized with the same scheme to avoid accumulation of errors. As the number of quads is very low we can use 32 bit floats for the position, while 16 bit integers are sufficient for the min. and max. texture coordinates. In total this amounts to $4 \cdot 3 \cdot 32 + 4 \cdot 16 = 448$ bits per quad. For the texture we either have the possibility to use hardware supported lossy compression with the additional advantage of reducing transfer rate during upload or lossless compression formats (e.g. TIFF with Huffman compression).

3.4 Rendering

To render the scene we first determine the required level of detail and the visibility of cells. The octree is traversed and at each node the visibility is checked using view frustum culling. For each visible cell its approximation error is projected onto the screen and if this screen space error is too high (e.g. $> \frac{1}{2}$ pixel), the traversal is continued to finer levels.

Even with guaranteed screen-space accuracy, subpixel-cracks may appear as in other out-of-core rendering algorithms. Previous methods like fillets or fat borders however require the boundary of subparts to be known and thus cannot be used since it can become arbitrarily complex due to transparency in the texture. When viewing two adjacent quads along their normal direction, the crack can be filled by extending the size of the quads by the approximation error ε in each direction to ensure overlapping textures. Note that this implies that nearby points of neighboring nodes on the same level of the octree have to be rendered as well when generating the texture. To hide subpixel-cracks between octree cells introduced by not completely aligned planes, each plane is slightly tilted towards the viewer (see Figure 3.3). This rotation is performed by offsetting the vertices of the quad by at most ε along the plane normal and therefore changes in the appearance of the model are again in the subpixel range. The amount of displacement of a vertex can be calculated from it's relative position to the quad center and the direction from the viewer to the quad center. By passing the quad center position as an additional parameter, the tilting can be performed in the vertex shader and does not introduce a performance penalty.

Due to the texture resolution, the level-of-detail selection already performs the part of the texture filtering which normally is done by mip-mapping, since each texel projects to at most one pixel when using a screen space error of $\frac{1}{2}$ pixel. We do not need to take additional care to perform anisotropic filtering, since it is

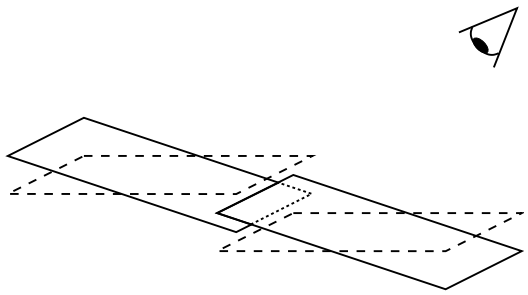

Figure 3.3: Rotation of planes to visually close cracks between octree cells (original planes are dashed).

supported in hardware for textures.

For prefetching, the priority based approach of [Guthe et al., 2004] is used in order to load data for subsequent frames. Here the loading priority of a cell's geometry depends on the viewer's movement that is necessary for the cell to become visible.

3.5 Implementation & Results

For the octree cell size we use 64 units, which means that the data contained in each leaf cell projects to 64×64 pixels on the screen at $\frac{1}{2}$ pixel screen space error. This value is a good choice for the granularity of the culling and LOD selection on the one hand and on the other hand fulfills the needs of robust and efficient plane detection. For this cell size a value of a few hundred candidates for the RANSAC proved to give very good results and nevertheless achieves an excellent performance. This number of candidates is enough to handle complex featured areas, where sometimes more than ten planes are detected and is able to fine-tune slightly curved areas without producing much overhead, since after two or three planes all points are accounted for. As texel to vertex performance gain factor we chose $\varphi = 8$, since firstly we expect texels to render between 10 and 20 times faster than points and secondly four texels need the same amount of graphics memory as a single point.

For evaluation we used two colored point clouds, the Welfenschloss shown in Figure 3.1 and the choir screen shown in Figure 3.4, both acquired with a 3D laser scanner. In order to perform a reasonable comparison with current point rendering approaches, we first resample the model in order to avoid unnecessary high point counts due to overlapping scans. The preprocessing times, recorded on a 3.4 GHz Pentium 4 with 1 GByte main memory and a GeForce 6800 Ultra, are listed in Table 3.1. Especially worth mentioning is that the preprocessing performance

Figure 3.4: Choir screen point cloud with 1mm accuracy (resolution of 3200 x 1200 pixels) rendered with our method.

of about 15k and 29k vertices per second to construct all LODs is comparable to state-of-the-art out-of-core mesh simplification algorithms. Considering the number of points in the resampled models, the generated hierarchy requires 13.11 bits per point for the Welfenschloss and 5.23 bits per point for the choir screen, or 7.88 and 3.76 bits per point respectively if we take all points in the octree hierarchy into account.

model	Welfenschloss	choir screen
input points	3,151,573	21,104,869
leaf resolution	5 cm	1 mm
octree construction	53 sec	317 sec
plane fitting	97 sec	252 sec
texture generation	63 sec	153 sec
disk size	4.18 MByte	4.06 MByte

Table 3.1: Preprocessing times for different point clouds.

Figure 3.5 shows the fitted quads and the remaining points for the Welfenschloss model at octree level 4. Note, how well planar structures were detected on the specific scale and the remaining points are mainly occurring at fine geometric details.

The number of detected planes per level of the octree hierarchy is shown in Tables 3.2 and 3.3. From these simplification rates a theoretical speedup can be calculated with

$$speedup = \frac{v_s/r_v}{(v_r + 4q)/r_v + t/r_f},$$

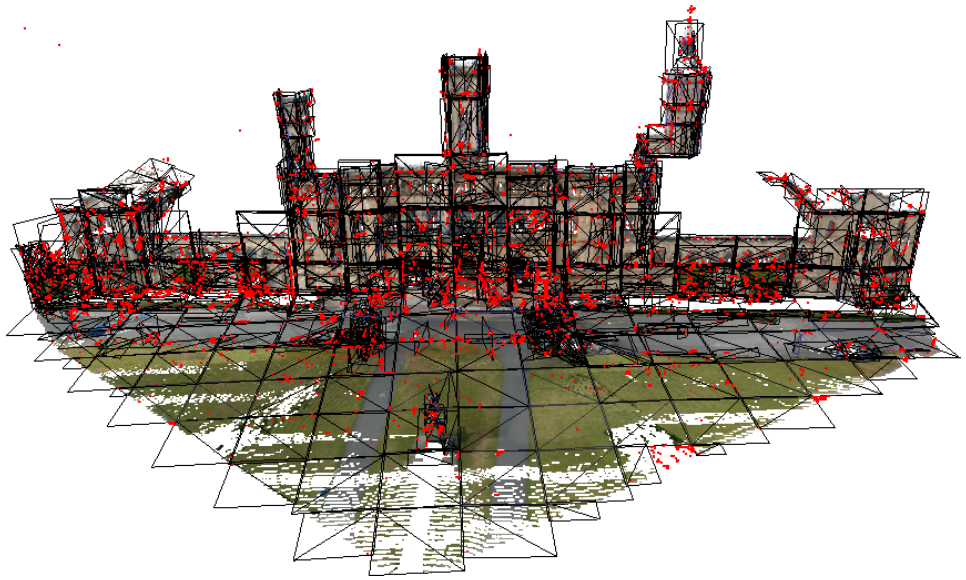

Figure 3.5: Fitted quads and remaining points for the Welfenschloss model at octree level 4.

where v_s is the number of simplified points, q the number of quads, t the number of texel and v_r the number of remaining points. The transformation rate r_v is given in vertices per second and the fill-rate r_f in fragments per second. Thus, if rendering simple GL_POINT primitives, the theoretical speedup factor should be about four to five for both models depending on the actual graphics card performance. Astonishingly, this expected speedup does practically not depend on the scale. When splats are used to render the point cloud to achieve comparable quality (anisotropic filtering) as our hybrid rendering method, the performance gain would even be about a factor of 50, since rendering a splat requires about as much time as rendering 10 GL_POINTS.

level	simpl. p.	quads	texel	remain. p.
0	1,707	6	4,646	85
1	6,504	19	15,082	362
2	26,020	65	56,467	1,209
3	102,522	224	230,996	4,583
4	386,248	815	866,993	18,696
5	1,255,646	2,669	2,921,690	76,462
6	2,673,675	5,907	6,996,920	416,901

Table 3.2: Simplified point cloud and generated planes with remaining points for the Welfenschloss.

level	simpl. p.	quads	texel	remain. p.
0	1,329	3	2,228	80
1	5,519	12	10,860	121
2	22,489	61	47,415	777
3	94,171	256	194,028	2,547
4	412,130	1,010	802,685	11,172
5	1,815,248	3,765	3,079,580	37,842
6	6,709,020	13,133	11,967,000	245,704

Table 3.3: Simplified point cloud and generated planes with remaining points for the choir screen.

Tables 3.4 and 3.5 show the measured rendering performance for each level of the two models on the same system ($r_f = 6.4$ billion texels/second, $r_v = 600$ million vertices/second) and a screen space error of $\frac{1}{2}$ pixel. Note that for a resolution of 800×600 (octree level 4) the measured speedup approximately matches the expected, i.e. a speedup factor of 4 and 5 is expected, while a factor of 5 and 6 is achieved. Note, that for higher resolutions and thus larger point clouds however, the performance of GL_POINTS significantly breaks down. This may be due to limited GPU memory reserved for geometry, or a driver issue.

resolution	points	splats	hybrid (points)	hybrid (splats)
800×600	345 fps	41 fps	1,780 fps	428 fps
1600×1200	22 fps	13 fps	583 fps	111 fps
3200×1200	11 fps	6 fps	123 fps	21 fps

Table 3.4: Frame rates for the Welfenschloss.

resolution	points	splats	hybrid (points)	hybrid (splats)
800×600	324 fps	39 fps	1,986 fps	639 fps
1600×1200	16 fps	9 fps	553 fps	185 fps
3200×1200	4 fps	2 fps	136 fps	33 fps

Table 3.5: Frame rates for the choir screen.

3.6 Conclusion

We have presented an efficient and robust plane detection algorithm for hybrid rendering of highly detailed point clouds. By exploiting an optimal balance between

vertex transformation and fill-rate, the rendering is significantly faster than low-quality point rendering (using simple GL_POINTS) and more than an order of magnitude faster than splatting with comparable quality. The balancing between points and textured quads leads to a smooth transition between rendering primitives, where points are used for structural details and the textured quads for planar regions. It can be observed that with decreasing scale details become parts of large-scale planar regions.

Our hybrid representation not only improves the rendering performance, but also reduces the memory requirements, since color and normal can be stored more efficiently in a texture than in vertex attributes. Furthermore, the planes at subsequent octree levels represent a hierarchical decomposition of the model which could be extended to detect semantic structures like buildings in scans of urban regions.

3.7 Acknowledgements

We thank Claus Brenner from the IKG Hannover for the Welfenschloss point cloud and Gerhard Bendels for scanning the choir screen. We especially thank Jan Meseth and Alexander Gress for helping us with the model preparation.

This work was partially funded by the German Science Foundation (DFG) as part of the bundle project “Abstraktion von Geoinformation bei der multiskaligen Erfassung, Verwaltung, Analyse und Visualisierung”.

CHAPTER 4

FROM DETAILED DIGITAL SURFACE MODELS TO CITY MODELS USING CONSTRAINED SIMPLIFICATION

Keywords: DSM, City Model, Geometry Simplification, Abstraction, Visualization.

Summary

We present a method to simplify high-detail full-featured digital surface models (DSM) of cities (i.e. containing the heights of trees, cars, buildings, etc.) geometrically in such a way that all relevant features are preserved, whereas noise and superfluous details collapse. The relevance of features is automatically evaluated using a semantically motivated shape detection and serves as constraint during the simplification. Our results show that we are able to preserve fine details of complex roof structures while all irrelevant features are effectively removed. Thus, we achieve an excellent abstraction of the city data without any interaction of the user, which is not only beneficial for visualization, but could also be used for GIS related applications.

This chapter corresponds to the article [Wahl et al., 2008].

Zusammenfassung

Von digitalen Oberflächenmodellen zu Stadtmodellen mittels eingeschränkter Simplifizierung. Wir stellen ein geometrisches Simplifizierungsverfahren für hoch detaillierte ungefilterte digitale Oberflächenmodelle (DOM) von Städten (inkl. Abtastwerten von Bäumen, Autos, Gebäuden, etc.) vor, welches alle relevanten Merkmale erhält, aber Rauschen und überflüssige Details verwirft. Die Relevanz der Merkmale wird automatisch mittels semantisch motivierter Formerkennung bewertet und dient als Einschränkung der Simplifizierung. Unsere Resultate zeigen, dass wir in der Lage sind, feine Details komplexer Dachstrukturen zu erhalten, während irrelevante Merkmale effektiv eliminiert werden. Auf diese Weise erreichen wir ohne jegliche Benutzerinteraktion eine ausgezeichnete Abstraktion der

Stadtdaten, die sich nicht nur für Visualisierungszwecke eignet, sondern auch in GIS-Applikationen benutzt werden könnte.

4.1 Introduction

Conventionally, the sole aim of geometry simplification in the context of real-time visualization of landscape or urban environments is to enable smooth, real-time navigation through the scene without disturbing interruptions for data loading or decoding. To this end, the simplification generates a suitable set of geometric levels of detail (LOD) of the terrain data. To sustain a satisfactory user experience, the blending in of additional detail without notable flickering or jumps while the user zooms in on objects should be supported by the underlying LOD structure. Hence, the LODs are usually generated with respect to a geometric error measure, e.g. Hausdorff error that guarantees pixel correct images at given viewing distances. However, often additional requirements arise when dealing with city-data:

- In the context of city visualization, a photorealistic visualization is not always desired, e.g. on a small PDA or cell phone display, the overwhelming amount of detail is difficult to grasp for the user and an abstracted view is usually preferred. The abstraction however should be semantically motivated and cannot be based on geometric error alone.

- A lot of existing GIS related software, e.g. for city-planning, operate on abstracted data in the form of CityGML or similar formats. To this day no automatic conversion of height-field data into this representation is available.

- For city visualization in a client-server setting over the internet, as it is available in a primitive form in Google Earth today, it is usually not possible to transmit all the necessary detail of geometry and texture in the short time available while the user navigates through the scene because of limited bandwidth. Therefore it is unavoidable that the user will frequently see coarser LODs from a distance where the simplifications therein become clearly visible (i.e. larger than a couple of pixels). Current simplification methods for high resolution height-field data however are only based on geometric error considerations, so that often façades of houses are askew or roofs have unnatural looking shapes, which results in views that are irritating to the user.

All of these requirements are not vital as long as we deal with city models derived from cadastral data or semi-automatic reconstruction, given that these models are generally reasonably abstract. However, considering the ongoing advances

in camera and reconstruction techniques and the consequently increasing detail and extent of full-featured digital surface models (DSM), automatic abstraction methods capable of handling out-of-core data are necessary.

In order to address this situation, in this work, we propose a novel form of constrained simplification that incorporates additional shape information together with geometric error considerations to generate LODs from highly detailed DSMs that respect both geometric as well as semantically motivated criteria. The incorporated shape information is low-level and very general. It is used to find edges and corners in the geometry that make up the important features of building geometry without resorting to more involved and specialized building models. The simplification is constrained to preserve these features even in coarse LOD. Due to the continuous nature of the LOD and the consideration of geometric error, this representation is still suitable for real-time pixel correct photorealistic terrain and city renderers. Moreover, due to the preservation of important edges and corners, it is applicable in client-server settings on the internet or visualization on mobile devices as well – all from the same data representation and generated fully automatically.

4.2 Previous Work

In Computer Graphics, LOD representations of objects and scenes have been extensively researched during the last 15 years. In combination with methods for efficient occlusion calculations, image based rendering as well as prediction and caching mechanisms they are employed for efficient visualization of large scenes.

4.2.1 Topology-Preserving Simplification

Even for triangulated height fields it is challenging to find an optimal approximating mesh with a given small number of faces in the sense of the L1-norm. Indeed Agarwal and Suri [1994] have proven this problem to be NP-complete. Therefore, iterative greedy algorithms have prevailed which in each simplification step either eliminate a vertex (vertex contraction) or an edge (edge collapse) from the triangulation [Schroeder et al., 1992; Hoppe et al., 1993] Several different error measures have been proposed and evaluated in the literature. Compared to other distance measures, the Hausdorff metric has the advantage that the projection of the 3D approximation tolerance onto screen space can be used to select a corresponding LOD automatically for pixel correct rendering [Klein et al., 1996]. The quadric error metric introduced later by Garland and Heckbert [1997b] has the advantage of a simpler and more efficient computation. Therefore, it has become very widespread, although it does not guarantee any bounds on the screen space

error. Since then, there were also improvements in computing fast Hausdorff distance approximations [Cignoni et al., 1998; Guthe et al., 2005].

4.2.2 Topology-Changing Simplification

The family of *vertex clustering* methods has been introduced by Rossignac and Borrel [1993] and has been refined in numerous more recent works, see e.g. [Low and Tan, 1997]. The algorithms of this family essentially apply a 3D grid to the object and for each cell contract all the vertices inside the cell. This way holes in objects are closed or objects in close proximity are merged. Although the degenerate faces are subsequently removed, it is difficult to influence the fidelity of the result due to lack of control over the induced topological changes. The already mentioned *vertex contraction* operator [Garland and Heckbert, 1997b; Popović and Hoppe, 1997] offers more control over the topological modifications. However, without further processing it possibly generates non-manifold meshes.

4.2.3 Out-of-Core Simplification

To simplify models of ever increasing size, a number of out-of-core simplification algorithms have been developed. El-Sana and Chiang [2000] sort all edges according to their lengths and use this ordering as decimation sequence. Lindstrom [2000] uses vertex clustering to reduce the number of vertices. As the representing position of each vertex cluster is computed from an accumulated quadric error metric, the memory requirement of the algorithm is proportional to the size of the output model. For cases where neither input nor output model fit into main memory, an out-of-core vertex clustering [Lindstrom and Silva, 2001] was developed. The multiphase algorithm [Garland and Shaffer, 2002] uses vertex clustering to reduce the complexity of the input model followed by a greedy simplification approach and achieves high quality results. Another way for out-of-core simplification is to split the model into smaller blocks, simplify these blocks and stitch them together for further simplification. In [Hoppe, 1998] this approach is applied to terrain and in [Cignoni et al., 2003b] to arbitrary meshes. The approach has the problem that special care has to be taken at patch boundaries. Recently, stream decimation algorithms [Wu and Kobbelt, 2003; Isenburg et al., 2003] for out-of-core simplification have been developed, but the resulting model is not optimal with respect to mesh size and Hausdorff distance of the simplified model to the original.

4.2.4 Remeshing

Another area related to our approach is remeshing of triangulated geometry. Remeshing algorithms take a triangle mesh and resample it such that some quality

requirements are satisfied but the original geometric shape is retained. In this sense, mesh simplification can be seen as a special case of remeshing. Other remeshing techniques include surface fairing [Taubin, 1995; Desbrun et al., 1999], where connectivity is preserved but vertex positions are optimized in order to remove noise or to evenly distribute vertex positions. Hildebrandt and Polthier [2004] presented a bilateral mesh smoothing algorithm that is able to preserve edges and corners in the geometry. A similar approach is given by Vorsatz et al. [2001] who describe a remeshing algorithm that is feature sensitive. Both approaches however are not combined with simplification and are not robust to outliers.

4.3 Overview

Given a high-resolution height-field model, it is converted into a 3D point-cloud and decomposed by our recently proposed efficient RANSAC shape detection [Schnabel, Wahl, and Klein, 2007a] into areas that correspond to primitive shapes such as planes, spheres, cylinders etc. and a set of remaining points. The points of the original DSM are then tagged with the indices of shapes detected in their proximity. These index sets then implicitly define the shape, edge or corner property of the points, which is subsequently used to constrain the geometric simplification. Only those simplification operations are allowed that respect the detected primitives on the corresponding LOD. This way it is asserted that coarse building models are generated which obey the abstract structure defined by the segmentation into primitives. Depending on the chosen size and approximation fidelity of the detected primitives, the resulting coarse polygonal models adhere to different semantically motivated levels of detail. In areas where no primitives could be detected (e.g. areas of natural cover such as in parks), the simplification is guided by geometric error alone, which has been proven to give good results for terrain in general.

4.4 Geometric Simplification

We build our simplification framework around the edge-collapse operation with tight upper bounds on the Hausdorff distance against the original mesh. Each edge of the original mesh generates three collapse candidates, which are either of the two corresponding halfedge-collapses or an edge-collapse with vertex placement. As optimizing the new vertex' position with regard to the Hausdorff distance, which includes evaluating the maximum, does not make sense, we use the quadric error metric [Garland and Heckbert, 1997b] for candidate generation. This metric is fast and easy to compute, and directly yields the optimal vertex for the edge collapse operation in general cases. For degenerate cases the distance to the original edge is

used as an additional criterion. Each collapse candidate is then checked for validity, that is whether it introduces flipping of orientations or degeneration of neighboring triangles, and scheduled in a priority queue keyed to its approximation error. For the sake of speed we again use the quadric error metric for computing priorities.

After these preparational steps, iteratively the best collapse candidate is evaluated, this time using the actual distance metric and if it does not surpass the current error threshold, it will be applied to the mesh. As it changes the appearance of its 1-ring, all conflicting candidates are rescheduled or deleted from the priority queue. This process comes to an end when each remaining valid collapse operation surpasses the threshold and therefore the bottom-up simplification scheme is in a local optimum. Although this approach is greedy, it is able to collapse a mesh completely, if the distance threshold allows it (i.e. it does not get stuck in a local minimum).

4.4.1 Distance Metric

For pixel-true rendering, the Hausdorff distance is almost the perfect choice, since it guarantees two crucial properties, directly linked to its definition:

Firstly, for every feature of the original mesh, there exists a part of the proxy mesh which represents that feature within a distance of at most the predefined threshold. And secondly, as also the inverse Hausdorff hemimetric is accounted for, the resulting approximation does not introduce artifacts which have no justification from the original mesh.

The arguments against using Hausdorff distance are that it is very difficult to compute and that in many cases, simpler approximations well serve their purpose.

Especially, in the domain of terrain rendering, measuring only along the z-axis is a popular alternative. Its main advantage is that opposed to strict Hausdorff distance, the counterpart on the other mesh is implicitly given and therefore, we get two piecewise linear distance functions parameterized over the plane. It has been observed that evaluating this metric only at the vertices of the two corresponding meshes does not yield tight bounds on the Hausdorff distance, but also the edges need to be considered, as otherwise the error can become arbitrary large. It can be shown that for small maximum steepness angles α the overestimation of the distance is bounded by $\cos^{-1}(\alpha)$. So this does not lead to significant overheads for coarsely sampled terrain datasets.

However, in the presence of high-frequency signal, which is very common in high-resolution digital surface models, this approximation is no longer effective. Therefore, we only use the implicit correspondence between the meshes as given by the z-projection and evaluate the Hausdorff distance locally between the corresponding parts. That way, the distance computations remain local (i.e. in the 1-ring

of the edge in question) and still we get tighter bounds and effective simplification of steep geometry.

4.5 Semantic Constraints

As mentioned in the introduction, LOD generation based on purely geometric simplification often leads to unwanted results, since it does not consider the overall shape, but only local geometric features. For terrain datasets, the resulting approximation is generally good enough, but especially for man-made objects as buildings, where the shape is often dominated by recurring patterns, geometric simplification fails to maintain symmetries and structures and is therefore not well suited as an abstraction method. Nevertheless, it has the big advantage that it always yields a complete representation of the underlying scene automatically, irrespective of whether it can interpret the scene or not. Therefore, it is desirable to combine its strengths with global semantic analysis which is able to identify important feature edges and corners in order to get the best of both worlds.

One way to have simplification respect the overall shape is via accordingly designed constraints. For this approach care must be taken that the constraints achieve the desired feature preservation and that they do not limit the effectiveness of the simplification.

In the following we will first discuss a very general method to automatically add semantically motivated metainformation to the input data. Then, we deal with how these data are used as constraints during simplification.

4.5.1 Edges & Corners

In this work, we propose to use primitive shapes to detect important edges in the height data. The reason a shape-based detection is preferred over more traditional methods such as Laplace edge detection is that the shape detection can handle outliers and noise in a robust fashion and has a more global notion of structure (i.e. based on connected components of parts with equal curvature), which enables it to detect edges reliably comprising a wide angle between two primitives, e.g. on top of a shallow roof.

As a first step, we employ the shape detection described in [Schnabel, Wahl, and Klein, 2007a]. As it operates on 3D point-clouds, the input height-field is first converted to 3D by insertion of additional points at discontinuities in the 2.5D data (e.g. for façades). We use the same sampling density for this vertical upsampling as in the planar domain, in order to maintain a close relation between the number of samples and surface area. The resulting point-cloud $P = \{p_1, \ldots, p_N\}$ is partitioned into subsets S_i associated to shape primitives Φ_i (i.e. planes, spheres,

cylinders, cones and tori) as well as a single subset R containing any remaining points that could not be assigned to a shape for the given parameters. In order to ensure heuristically that only parameterizable patches are created, a point is considered compatible if its Euclidean distance to the shape is within a given distance threshold and its normal does not deviate from the respective shape normal by more than a given angle threshold. After removing the compatible points, the algorithm is restarted on the remaining points until no more shapes can be found for the given set of parameters.

For details on the efficient probabilistic RANSAC-based algorithm we refer to the original work, we only want to emphasize here that there are parameters which allow us to select what kind of shapes are considered valid and therefore define a low-level interface to the interpretation of the data (e.g. surface area). If wanted also more complex parameters (e.g. neighboring shapes, shape orientation) can be used to decide whether the shape is important or not (cf. [Schnabel et al., 2008b]) or the results can be cross-validated against cadastral data. But as we aim at a high level of automatism and generality we will work with the inherent data and few parameters if possible.

In our setting, we define vertices of the DSM to be edge points if they are close to two different shape primitives. Points that are close to even more primitives are classified as corner points. For closeness again we use a distance threshold ε, but this time we do not measure to the ideal shape but its points. That is, a point is close to Shape j if it is within ε distance of any of the points from S_j. In order to identify all edge and corner points efficiently, the point-cloud P is sorted into an axis aligned 3D grid. Then for all grid cells that contain points belonging to

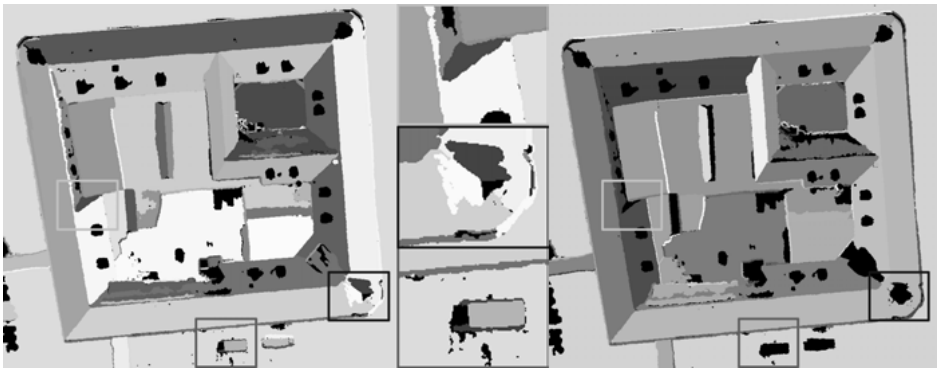

Figure 4.1: Shape detection results with 4m^2 (left) or 16m^2 (right) size thresholds. Intensities are random, black means no shape detected. The middle column shows close-ups of a small part of the roof, a large dormer and a truck, which are no longer present in the 16m^2 detection result.

different shapes, the contained points' distances are compared to ε and a counter is increased for each potentially different assignment. In order to avoid discretization dependencies due to the location of the grid cells, we use eight translated versions of the grid, corresponding to the eight corners of a cube. Given the distance threshold ε, the width of the cells is set to ε and shifted versions of the grid are created with an offset of $\varepsilon/2$ along the respective axes. Cells are stored in a hash table, so that memory is only allocated for occupied cells. However, in order to get most out of the semantic constraints it is valuable not only to classify edges and corners, but to keep the whole information to which shape each point corresponds. This additional information will be used to not restrict simplification in the presence of features blindly, but to guide which of the possible combinations of features are allowed. This information is stored in an additional raster of shape-IDs, which is read along with the height field during simplification.

4.5.2 Constrained Simplification

In order to respect and maintain the shape information of the vertices, we pose an additional constraint to each collapse candidate during validity check (see sec. 4.4): The vertex which is about to collapse must ensure that its set of shape-IDs is a subset of the shape-IDs of its collapse partner.

That this simple rule maintains the vertex' shape-IDs is obvious, but how does it help in maintaining features? The principle is that a vertex, once it collapsed to a corner or edge, cannot move away from there, as it can only move along the feature. So, as the IDs are globally unique and each two planes share only one line (see sec. 4.6 for discussion of non-planar shape primitives), this approach guarantees that every corner and every edge as defined by the shape-map is maintained.

But whenever a feature edge is not detected along its whole extent, or is not enclosed by two corner features, it might collapse to a single point, which is of course not the desired representation. This case occurs very often due to the presence of noise and occluders and because of the incomplete shape segmentation. In digital surface models, this is probably the default case. In order to cope with that situation, we suggest the use of additional topological constraints. We define border vertices of a shape as those vertices which have one incident edge pointing to a vertex that is not in the same shape. Such vertices are not allowed to move inside the shape, but may only collapse to neighboring border vertices. This constraint can be checked by looking at the shape-IDs of the two tip vertices of the incident triangles. One of these must be outside of the shape if the collapse takes place at the border. As opposed to labeled edge vertices, this criterion does not allow finding a low-error approximation within a defined small range in the proximity of the hypothetical intersection, but the purpose of maintaining the border is served and still effective complexity reduction along the border is possible.

Now there remains one situation in which detected features still might degenerate, namely if two edges of the same shape do not meet in a common corner but are connected via a series of border vertices. As the collapses along each of the two edges are legitimate, they may again collapse to their next corners respectively introducing an unwanted shortcut edge. We deal with this problem by detecting the implicit corners defined as those edge vertices which only have one neighboring border vertex with respect to one of their shapes. Implicit corners are then treated as corners and may not be collapsed to other vertices unless they are of the same corner type.

4.6 Results

We tested the proposed methods on a 256m $\times$ 256m part of a highly detailed digital surface model of downtown Berlin, featuring complex buildings at an input resolution of 12.5cm (resampled from the 7cm resolution dataset courtesy of DLR). The original heightmap therefore contained 4.2 million vertices. After adding the façade points, the point-cloud had more than 11 million points. The shape detection took 197sec. and resulted in 1,658 planes larger than 4m^2 and 695 planes larger than 16m^2.

The resulting segmentations are depicted in Fig. 4.1. The small borders around the buildings are no artifacts but belong to the façades which are not represented in 2.5D. Now we performed geometric simplification with exactly the same algorithm but either using a map of shape-IDs or not. Without shape-IDs the simplification of the 4.2 million vertices took 266sec. and resulted in 2,172 vertices, with the constraints it took 291sec. and resulted in 7,493 vertices. Fig. 4.2 shows the resulting models. The leftmost column of Fig. 4.2 shows the results at an error threshold of 4m without additional constraints. Even the huge gable roofs look already scrambled, the small chimney in front turned into a strange looking peak and also on the flat roofs in the background we see some disadvantageous collapse artifacts. Texturing this model (lower row) reveals the spatial inaccuracy of the feature edges. That is definitely not what we would expect from an abstracted model, even though from a distance where a pixel projects to about 4m it will be almost indistinguishable from the original.

In the middle column we see which difference the constraints make here. Geometric error is the same, but all collapses trying to demolish feature edges were inhibited. A lot of features, which are significantly smaller than 4m and hence missing in the leftmost mesh are still present in the data, e.g. the glass roof in the courtyard or the chimney are reasonably represented. The textured rendering reveals the high positional accuracy of the feature edges which is due to the small ε threshold used during point classification. This effect becomes even clearer when

we look at the rightmost column of fig. 4.2. As the whole patch is only little more than 30m high, distance threshold 32m means collapse everything you can. So, every feature which is still there is there due to the shape-IDs it has.

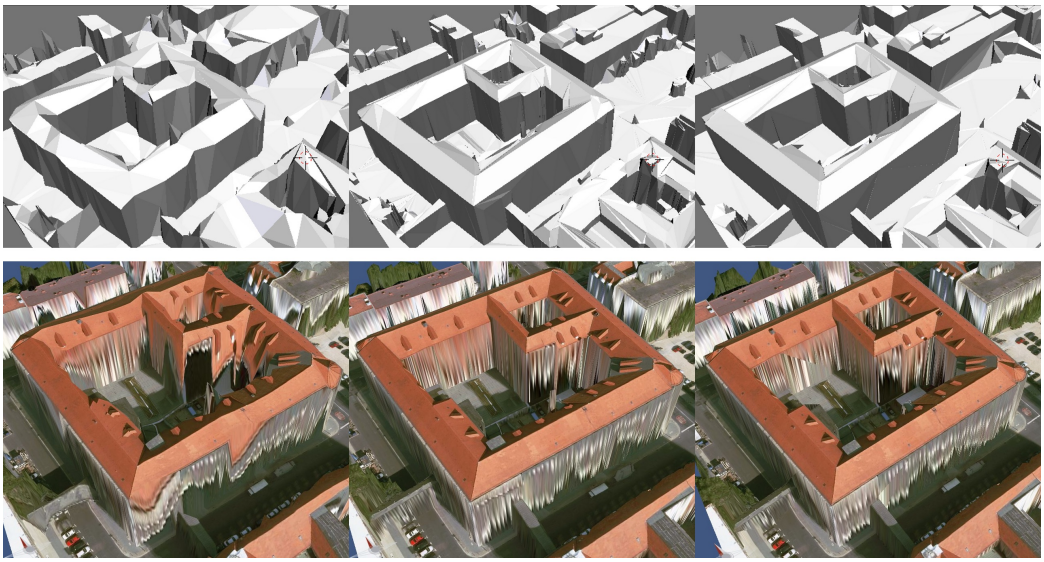

Figure 4.2: Simplification results. Left column: unconstrained, 4m threshold. Middle column: constrained, 4m threshold. Right column: constrained, 32m threshold. The upper row shows the shaded TIN whereas the lower row shows a rendering textured with full 12.5cm resolution.

4.6.1 Conclusion

We proposed a robust way to derive feature edges and corners from highly detailed digital surface models. Such constraints can be easily integrated into a Hausdorff distance simplification framework. Adding topological shape constraints and inhibiting collapse-vertex placement makes the resulting mesh strictly following the prescribed edge features, while still simplifying along these edges. Since the features are defined using a low-level shape detection, we are able to preserve the shape of very complex roof structures and buildings without having a specialized model of them. If a semantic annotation was added, the resulting geometry could be directly exported into a high-level format as CityGML.

Directions of future work will include a better support for curved features, such that there also the positional accuracy is independent of the global error threshold and also refining the definition of shapes, such that they approximately match existing concepts of semantic LODs. As the results in figure 4.2 revealed that, the

resulting triangulation is in some places even less complex than the pure geometric simplification, we will also try to improve the concept of shape such that vegetation and point-cloud artifacts do not lead to additional constraints.

4.6.2 Acknowledgements

We thank DLR – Institute of Robotics and Mechatronics for providing us the high-detailed Berlin dataset. This work was funded by the German Research Foundation DFG as a part of the bundle project “Abstraction of Geographic Information within the Multi-Scale Acquisition, Administration, Analysis and Visualization”.

CHAPTER 5

OUT-OF-CORE TOPOLOGICALLY CONSTRAINED SIMPLIFICATION FOR CITY MODELING FROM DIGITAL SURFACE MODELS

Abstract

We present a framework for rapid reconstruction of building models from very large, high-detail digital surface models (DSM) of urban areas. Our method is based on a geometric mesh simplification approach augmented with shape constraints [Wahl et al., 2008, chapter 4]. This approach allows to abstract the full-featured DSM in such a way that important structural elements are maintained irrespective of the approximation accuracy.

In this paper we present two major extensions. Firstly, we deal with situations, where the original approach may generate artefacts due to incomplete or inconsistent structural information, mainly caused by vegetation close to the facades. We present refined topological constraints, which handle these problems and a filtering which neglects structural information that contradicts the mesh topology. Secondly, we extend the computational framework to be fully out-of-core capable and present a way to parallelize computations on multiple cores.

We demonstrate the efficiency of our method by showing results for downtown Berlin a dataset containing more than 1 billion height samples processed in less than 30 hours.

This chapter corresponds to the article [Möser, Wahl, and Klein, 2009].

Keywords: DSM, City Model, Geometry Simplification, Abstraction, Visualization

5.1 Introduction

The amount of digital data available for cities has drastically increased in the recent years. The advances in sensor technology such as airborne LiDAR or stereo

Figure 5.1: Rendering of the automatically reconstructed city model of downtown Berlin on top of a corresponding DTM.

cameras combined with post-processing technologies have led to raw city data with point densities of about 200 samples per square meter.

Automatic reconstruction of abstracted city models from such data is a challenging task. On the one hand the detailed signal is demanding, as it allows to model finer structures, on the other hand the enormous size requires efficient, scalable, out-of-core methods (i.e. methods which do not rely on the whole dataset residing in main memory).

Approaches that reconstruct building models from raw data can be roughly categorized into two classes. Model-based approaches try to fit a model to the data, and are therefore limited to a predefined set of building types, specific characteristics are not maintained. Data-driven approaches characterize a building

by a set of primitive shapes (mostly planes) which can build almost arbitrary complex constellations. However, even the generation of the final model based on the detected primitives can be intricate for complex roof structures.

Wahl et al. [2008, chapter 4] proposed a new data-driven method motivated by DSM simplification which circumvents the problem of model generation. Instead of defining a polyhedron for a given set of planes, they constrain a classical geometric mesh simplification approach to respect the previously detected planes. Although this approach achieves a good abstraction of complex roof structures while maintaining a high positional accuracy of roof edges, problems arise if the shape borders are not well represented in the data. The proposed solution to this problem was the introduction of topological constraints.

In this work we analyse how this approach can be extended to achieve good results also in the presence of severe noise and occlusion artefacts as those caused by vegetation at the borders. To this end we analyse situations that lead to artefacts and extend the concept of topological constraints to deal with such situations. Moreover, we show how this approach can be efficiently extended to handle out-of-core datasets efficiently and demonstrate this with a full-featured dataset of downtown Berlin.

5.2 Related Work

While digital surface models based on laser point clouds usually reveal a higher positional accuracy and due to multiple returns generally allow an easier filtering of vegetation, DSMs based on stereo reconstruction Hirschmüller [2008] offer a matching photometric signal and therefore can be used directly in a visualization. Although many publications regarding building reconstruction explicitly mention Lidar data in their title most of the presented techniques can be directly adapted to stereo reconstruction results.

5.2.1 Automatic City modeling

Starting with the availability of airborne lidar data a lot of research was devoted to automatic building reconstruction based on such data. While the model-driven approach [Weidner and Förstner, 1995; Maas and Vosselmann, 1999] is better suited for simple models. As it easily allows to put constraints on the models it is especially suitable for coarsely sampled buildings. Subsequent model-based approaches tried to enlarge the class of possible buildings, recently Arefi et al. [2008] proposed a projection along dominant axes determined by morphology. Although such enhancements increase the set of possible models, the main restriction remains. The data-driven approach, based on segmentation as in [Rottensteiner

and Briese, 2002; Dorninger and Pfeifer, 2008] is generally able to maintain all sorts of features, as it does not need to understand all parts of a model. However, as Tarsha-Kurdi et al. [2007] points out the main disadvantages of the data-driven approach are sensitivity to detail features and noise as well as high computational costs.

5.3 Overview

As mentioned above the presented framework extends the approach by Wahl et al. [2008, chapter 4] who combine geometric simplification with semantic constraints, derived from detected primitive shapes, to reconstruct building models from a full featured DSM. Therefore, we first review the basic approach and afterwards roughly analyse the problems and outline proposed solutions in section 5.3.3.

Generally the processing pipeline from a full-featured DSM to the final building models consists of two stages. In the first stage, primitive shapes of a desired fidelity are detected in the height-field to identify important features, namely prominent edges and corners, and how they are connected. Subsequently, an abstracted city model is computed by simplifying the triangulated DSM with respect to the detected shapes and geometric error. This combination allows high abstraction in a geometric sense while details like roof structures are accurately preserved. Finally, the models' connected components above the ground are automatically extracted and textured by projecting the image mosaic.

5.3.1 Shape Detection

Although our framework would support any type of shape information, we use primitive shapes to detect important features in the height-field, as they can handle noise and outliers in a robust way and directly provide additional topological information, which is later exploited in the simplification process. We employ the RANSAC based shape detection proposed by Schnabel et al. [2007b] as it allows us to efficiently process huge point-clouds. The height-field is converted into a 3D point-cloud with additional points at discontinuities. For these we use the same sampling density, as in the planar domain, to maintain a constant ratio between number of points and surface area. As facades are badly represented in a DSM and thus often suffer from noise and reconstruction errors, coarse detection parameters are needed to avoid the decomposition whole facades into many small segments. On the other hand, fine parameters are needed to accurately capture fine roof structures. We account for this issue by splitting the point-cloud into roofs and facades, which allows us to adjust the used parameter sets for both domains independently. The shape detection partitions the point-cloud P into

disjoint subsets S_i, which are associated to a plane Φ_i, and a subset R containing all points not assigned to a shape. Points are compatible to a shape candidate if their Euclidean distance and normal deviation to the shape is smaller than a defined threshold. Additionally, shapes need to have a minimal number of compatible points to be accepted. After removing the compatible points, the algorithm is restarted on the remaining points until no more shapes can be found for the given set of parameters. With the low-level constraints described above, we are able to control the size and accuracy of the detected shapes and thus the maximal possible degree of abstraction. For further details we refer to the original publication.

To identify edges and corners in the DSM, we create an additional raster layer (shape map, see figure 5.2) which stores the globally unique indices of associated shapes for each vertex of the height-field. The shape map for a vertex v is defined as $\varsigma(v) = \{i | \exists w \in S_i : d(v, w) < \epsilon\}$, where $d(x, y)$ is the Euclidean distance of both vertices and ϵ a predefined distance threshold. Vertices v are defined as edge vertices if $|\varsigma(v)| = 2$ or as corner vertices if $|\varsigma(v)| > 2$. Now we can define the edge of two shapes Φ_i and Φ_j as $Edge(\Phi_i, \Phi_j) = \{v \in DSM | i \in \varsigma(v) \wedge j \in \varsigma(v)\}$. As can be easily observed in figure 5.2 these shape edges will generally not form simple chains of vertices but elongated sub-meshes. The same holds for corners. This is the main reason for the definition of topological constraints which deal with situations along the borders of such edge or corner meshes.

The shape map is used in the subsequent simplification and allows us not only to identify important features, but to also determine the involved shapes.

5.3.2 Constrained Simplification

Our simplification framework is build around the edge-collapse operation with tight upper bounds on the Hausdorff distance against the original mesh. Each edge of the original mesh generates two halfedge-collapse candidates. These collapse candidate are then checked for validity, that is whether it introduces flipping of orientations or degeneration of neighboring triangles, and scheduled in a priority queue keyed to its approximation error. For the sake of speed we use the quadric error metric [Garland and Heckbert, 1997a] for computing priorities as it is fast and easy to compute. After that, iteratively the best collapse candidate is evaluated, this time using the actual distance metric. If the Hausdorff distance does not surpass the current error threshold, the halfedge-collapse will be applied to the mesh. As it changes the appearance of its 1-ring, all conflicting candidates are rescheduled or deleted from the priority queue. This process comes to an end when each remaining valid collapse operation surpasses the threshold and therefore the bottom-up simplification scheme is in a local optimum. Although this approach is greedy, it is able to collapse a mesh completely, if the distance threshold allows it (i.e. it does not get stuck in a local minimum).

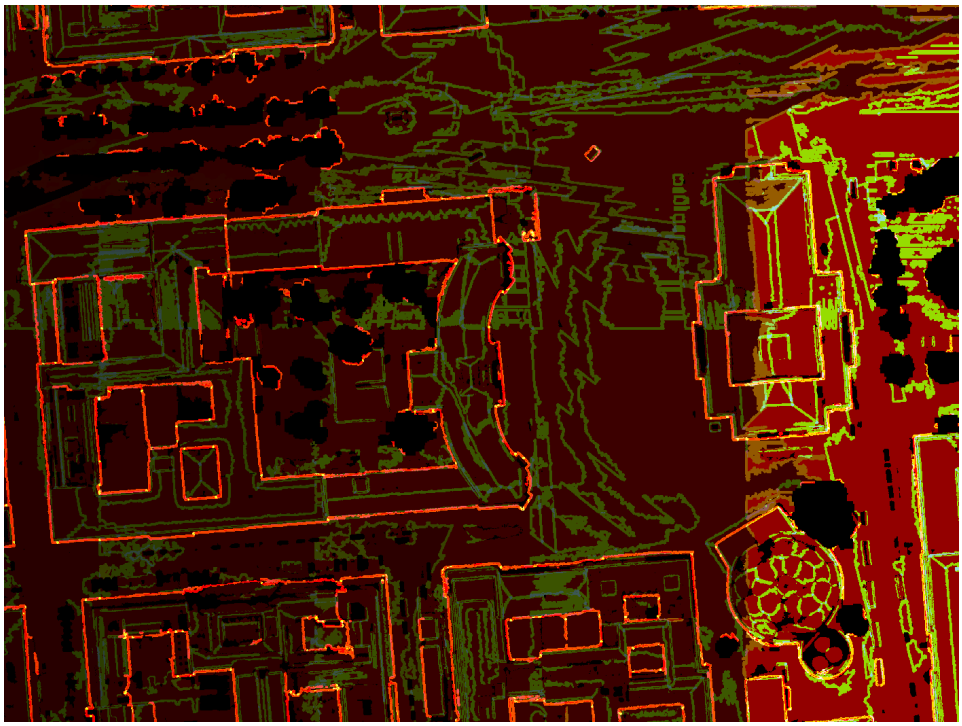

Figure 5.2: Example of a Shape Map depicted as RGB values: intensities correspond to shape-IDs, reddish color means one associated shape, green and orange regions belong to two shapes, and bright yellow or bluish colors denote regions with three or more shapes.

The basic rule in the constrained simplification is, that a vertex v may only collapse to another vertex w if the shape-IDs $\varsigma(v)$ are a subset of the shape-IDs $\varsigma(w)$. Assuming we have a perfect segmentation of our DSM into planes, meaning all planes are bounded by edges which end in corners, this constraint ensures that vertices stay inside their associated planar domains. Edge vertices have two domains and to stay in both, they may only collapse along the edge, until they reach a corner. Since corners have three or more shapes to respect, they are only allowed to collapse to other vertices of the same corner, which may exist as we propagated the shape information in an ϵ region around the shape. Problems arise if the shape information is not perfect, e.g. edges are discontinuous or corners are missing. If allowed by the geometric error, edges and whole shapes can shrink to a single vertex if they are not constrained in all directions, as collapses along the edges and inside the shapes are legitimate. Figure 5.3 depicts two critical cases.

In case of incomplete edges Wahl et al. [2008, chapter 4] propose to restrict the topological shape borders. These are defined as those vertices with one incident

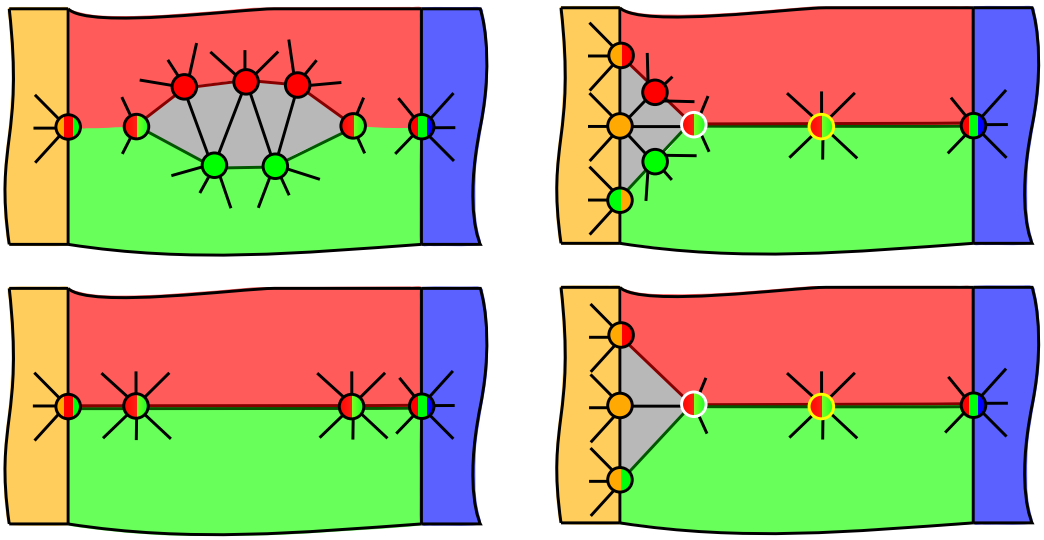

Figure 5.3: Two cases with incomplete shape constraints. Small circles represent vertices with color coded shape assignments. Left: discontinuous edge, collapsing of red or green vertices could lead to visible artefacts, Right: the implicit corner, marked as the dot with the white outline must not collapse to the incident border vertex (yellow outline). Upper row shows the initial constellation, lower row shows the allowed solution with topological constraints.

edge pointing to a vertex not in the same shape. To simplify along the shape border, vertices may only collapse to neighboring border vertices. This effectively allows to reduce the complexity even of defective edges if the both restricting corners are present. But if a corner is missing, the involved edges are only connected via a series of border vertices, the shapes still degenerate. To cope with this issue, additional implicit corners are defined as those edge vertices which only have one neighboring border vertex in the same edge with respect to one of their shapes and treated as regular corners (see figure 5.3).

5.3.3 Problem Analysis

The basic method provides two orthogonal ways of defining the result. The first is to steer the segmentation result and the second is to set the simplification accuracy. However, if the segmentation results are intended to give a coarse aproximation, a lot of constraints disappear and therefore a high geometric error threshold can easily lead to visible artefacts. This circumstance makes it difficult to filter out large unwanted regions, especially vegetation and still maintain a high-quality building model. Therefore, we analysed the situations in the presence of incomplete shape

information and extended the concept of topological constraints to avoid potential sources of artefacts.

When dealing with whole cities instead of small regions, we run into serious problems considering the quality of the shape map. Firstly, huge surfaces, e.g. ground surface, are segmented into several shapes. These slightly different planes introduce additional edges, subsequently called pseudo edges, which can be very broad (see the greenish areas in Fig. 5.2). Merging both planes to avoid such edges violates the assumption, that the maximal possible error of inner shape collapses is bounded by the error threshold used to detect the planes. While this might not pose a problem for two shapes, the systematic error may accumulate and cause artefacts if many such planes are merged.

Secondly, we often have vegetation close to buildings, which can cause holes in the shape map and inhibit the robust detection of edges between ground and buildings. In this case, but especially in combination with the aforementioned pseudo edges, the proposed constraints are too weak to avoid undetected facade vertices to be pulled away. To deal with these issues, we define the semantic constraints in a more general and stricter form in section 5.4, which also includes a more general notion of borders and corners. As these only restricts vertices associated to a primitive shape, artefacts caused by collapsing vertices without shape information are still possible. As shown in figure 5.5, these can be caused by collapse operations, that establish a connection between non neighboring shapes, subsequently called tunnel. The prevention of such tunnels is discussed in section 5.4.2.

Additionally, the accuracy of the height-field differs between buildings or even within building parts. While the representation of roof-structures in a digital surface model is usually quite accurate, facade edges are often jagged or even appear twice. The first case can be easily handled with appropriate shape detection parameters, but the second is only covered by the original approach if the facade edges are continuous. To remove these double facades, which are usually very short and thus appear as spikes, we propose an additional filtering in section 5.4.3.

Another drawback of the original approach is its limitation to small datasets that completely fit into main memory, which inhibits the application to whole cities. To remedy this drawback, we modified all stages of the processing pipeline to efficiently run out-of-core with a small memory footprint. Moreover, for practical purposes we introduce a high-level parallelization scheme which gains a speed-up in the order of the employed number of cores.

The contributions of this paper are

- more general topological constraints to avoid artefacts
- a tunnel constraint for better results at unclassified vertices
- a topological filtering, leading to simpler models

- a processing schema, allowing for whole cities to be processed
- a parallelization schema, improving the efficiency

5.4 Topological Constraints

5.4.1 Topological Corners and Edges

Although we obtain satisfying results using the original definition of edges and corners (implicit and explicit) for a moderate geometric error, high error bounds which are needed for a strong abstraction often cause severe artefacts in combination with missing shape information. Especially in regions with glancing intersections or multiple borders of the same shape meeting in a common vertex, vertices of undetected edges can be pulled away from facades. We illustrated these effects in figures 5.5 and 5.6. Both are very common situations in a full-featured DSM. The first is caused by intersections of nearly coplanar planes, which often happens in the segmented ground surface, resulting in large areas that are classified as edges. If these edges are not completely bounded, implicit corners are only detected were the borders of both involved shapes diverge, while vertices in between may move inside. The second is usually a result of missing shape information and may also result in the demolishing of not explicitly detected edges.

In both cases the shape topology gives additional hints about the form of shapes and the building structure to prevent unwanted collapses. Therefore, we introduce a more general notion of corners and edges from a topological point of view. Instead of considering planes as regions bounded by edges and corners, we use the dual view in which edges and corners are implicitly defined by the intersection of bounded planar regions. Using this notion, it is quite obvious that only simplifications should be applied which respect the borders of all shapes simultaneously. The border of a shape is always respected by collapses of vertices inside the shape and for border vertices if they collapse to a neighbor vertex of the same border. In this sense, we define topological edges as the intersection area of two shapes, which is similar to the previous definition of edges. In constrast, we define topological corners as those vertices which can not collapse to any neighbor vertex without disrespecting a shape border or which connect multiple borders of the same shape. This means that two criterions have to be checked when collapsing border vertices. Firstly, the collapse operations have to be valid for all involved shapes, otherwise those shapes become demolished. And secondly, all borders meeting in the vertex which is to collapse, must enter and leave the one-ring in the same two neighbors. Otherwise, borders are diverging or ambiguous in the considered vertex, which classifies it as a topological corner .

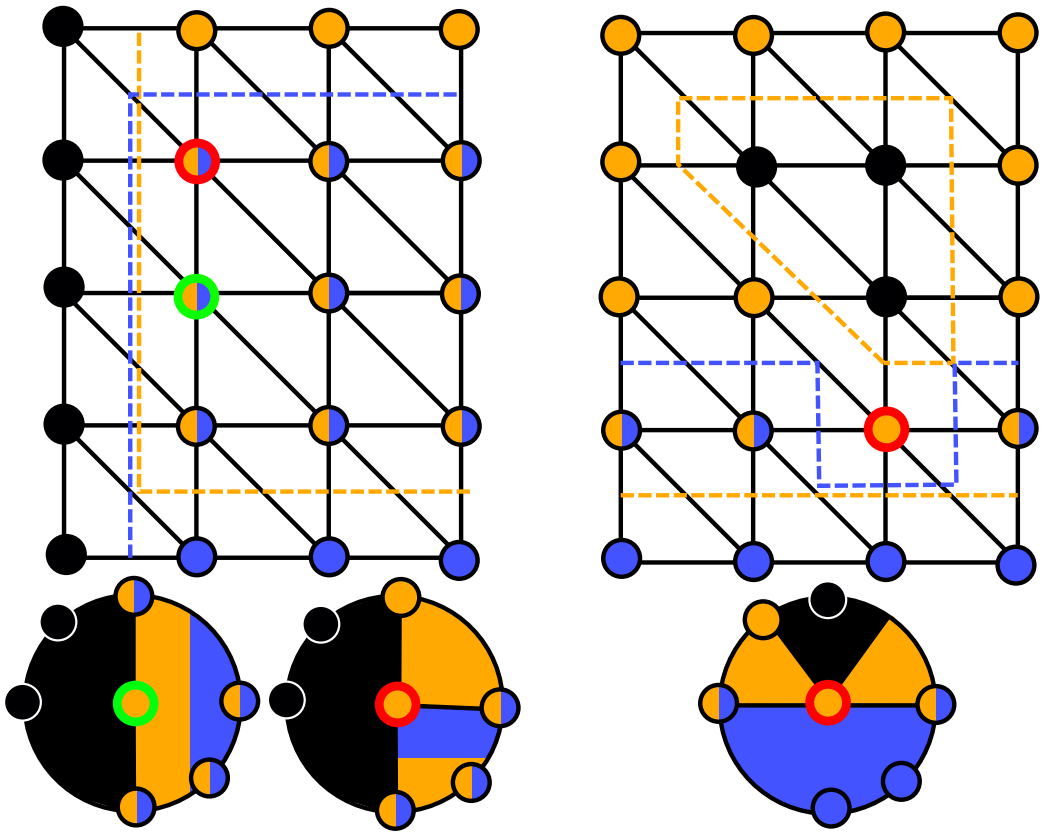

Figure 5.4: Schematic depiction of critical shape constellations. Left: broad edge, green vertex may collapse along border, red vertex not (implicit corner), Right: red vertex may not collapse because of multiple incident orange borders, lower row: shows the corresponding shape assignement of the highlighted vertices in the one-ring

5.4.2 Treatment of Points without Shape Information - Tunnel avoiding

To obtain a strong abstraction, we inevitably introduce holes in the shape map, as fine building details are discarded. The same is true for vegetation or other unwanted objects like cars, which may be either not detected or masked out. Vertices without any shape information have to be treated with great care. Interpreting such vertices as jokers, which can collapse to any other neighbor, even if surpassing the geometric error, might lead to severe artefacts (see figure 5.6). Especially when dealing with vegetation this can cause a connection between buildings and nearby trees, which is not the desired result. Although we have no direct assignment to a

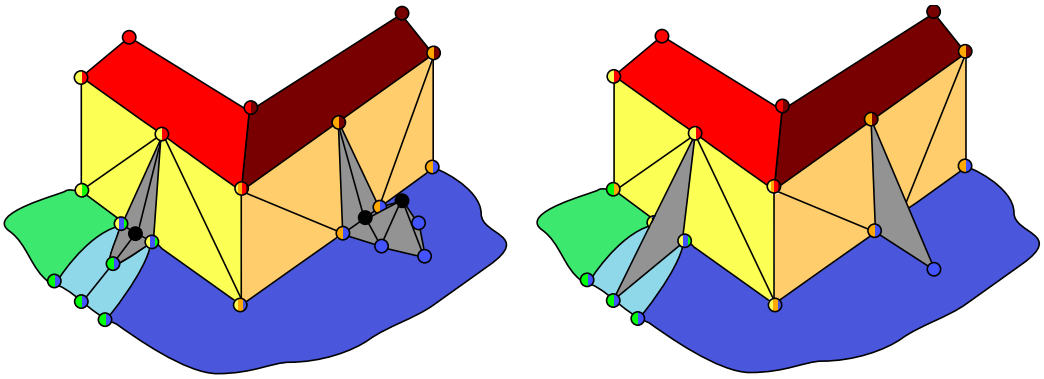

Figure 5.5: Left: initial situation, pseudo edge connected to the left facade, the lower edge of the right facade is discontinuous due to vegetation. Right: possible result with the original approach, black dots mark vertices without shape assignment

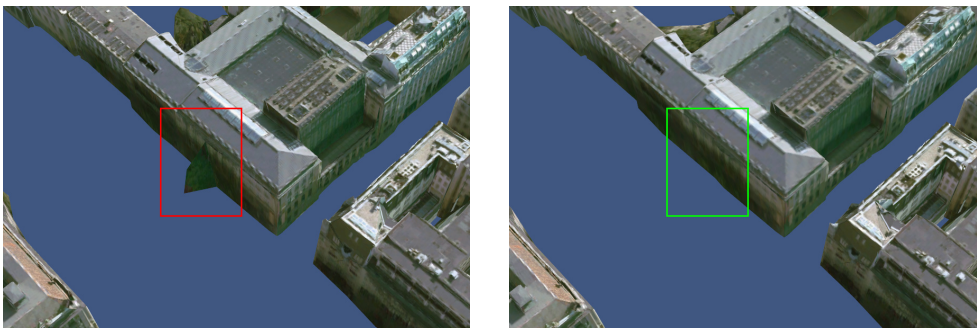

Figure 5.6: Simplification results. Left: without tunnel constraint. Right: with active tunnel constraint. Less artefacts are caused by vegetation.

specific shape, we can at least discard certain collapses that conflict with the shape topology in the neighborhood. When collapsing such a vertex, we connect vertices that were previously separated. As these may be associated to a shape, we may also connect shapes, that are not directly connected in the dataset. In the case of vegetation near facades, such connections often cause a direct connection between a ground vertex and a roof vertex. We call such a connection tunnel. The solution is to find a legitimate connection between the corresponding shapes and let the unassigned vertex only collapse to such legitimate connection vertices. Technically, a collapse is valid if the destination vertex has a common shape with each neighbor of the source vertex. In situations as illustrated in figure 5.5 this effectively avoids artefacts.

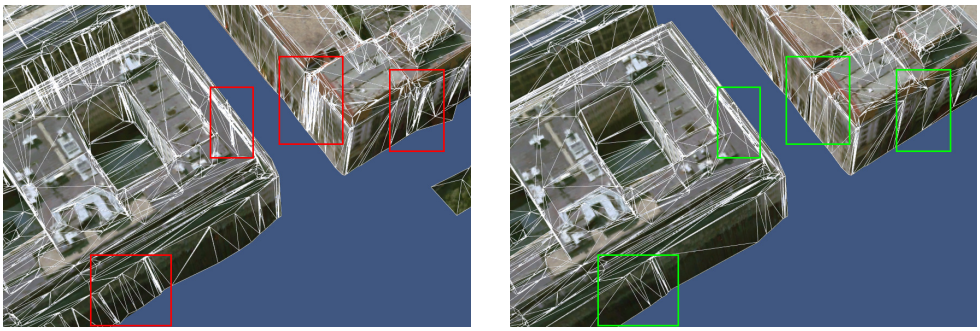

Figure 5.7: Simplification results. Left: without topological filtering. Right: with topological filtering. Removing spikes drastically decreases the triangles count.

5.4.3 Topological filtering

Double facade segments, usually visible as spikes (see figure 5.7), are hard to remove. Often they wrongly belong to a roof shape, which prevents a collapse to ground shapes. Even worse they can also provoke simplification deadlocks in parts nearby, because geometric errors such as flipping of orientations or the degeneration of neighboring triangle may be introduced. To remove these double facades, we propose a topological filter, which deletes shape-IDs if they are not supported by a neighboring triangle. For regular shapes this is always the case, whereas isolated, unbounded parts shrink during the simplification process until only a single line or vertex remains, which will then be removed.

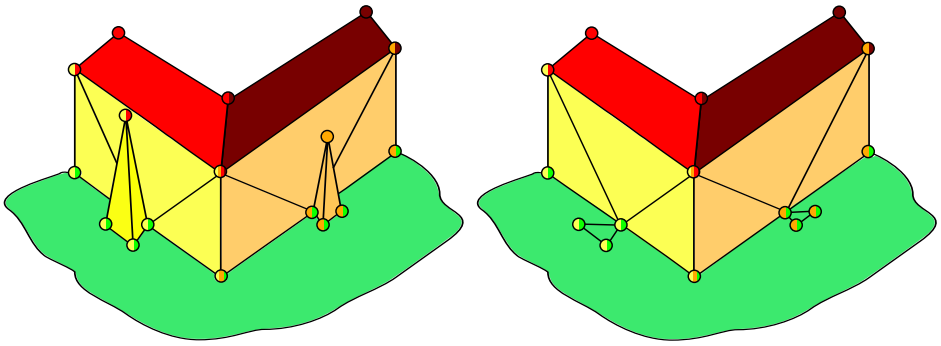

Figure 5.8: Left: Two spikes at roof-edges, the left is protected by shape constraints and the right one only by its size, Right: spikes are removed using the topological filtering

But even if allowed by all constraints, such collapses would cause a huge Hausdorff error and thus are forbidden if only a moderate error is allowed. In this

case we could increase the possible geometric error for such vertices, but since we also have similar spikes at vertices without erroneous shape information. Since we detected planes with a given accuracy, we assume that the error introduced by inner shape collapses is tightly bounded. As the overall shape is preserved by the detected edges and corners, we can safely increase the possible simplification error for non border vertices until all artifacts are removed.

5.5 Out-of-Core, Parallel Computation

We achieve both out-of-core and parallel computation by cutting the dataset into smaller pieces either using a regular grid or a quadtree hierarchy. Since all steps in the pipeline depend on neighborhood information, e.g. shapes in the proximity of a point, or need consistent transition between pieces, we can not treat them independently. Generally to cope with this issue, cells are processed consecutively line by line from west to east, while data from neighboring cells in the east and south is included in the computation. Changes that influence unprocessed cells are propagated to ensure consistent transitions. Applied to shape detection, we use a regular grid whose cell size has to be at least as big as the maximal shape size to detect. To guarantee such shapes to be completely inside the considered point-cloud, points from the three neighbors in east and south direction are included in the detection. Shapes that intersect the cell borders are propagated to the corresponding cells while the shape connection points in the intersection region are preserved to allow a correct continuation. In consecutive cells, existing shapes are continued and the algorithm restarts.

The same approach is applicable to the constrained simplification. We first partition the domain of the DSM using a quadtree. In the finest level, each cell constains a quadtree triangulation of that part of the height-field. The simplification works from level to level from fine to coarse. First, the finest level is simplified in the above mentioned order under the constraint, that north and west borders are left untouched whereas south and east borders have to apply the same collapses in both cells. In the next level, the simplified meshes of the previous level are stitched together and likewise simplified. This is continued until the coarsest level is completed. While this greedy approach allows us to process huge datasets, it complicates the computation of cells in parallel. In order to obtain correct results, we have to ensure correct data propagation and that no two threads access the same cells. As we have only forward dependencies in two directions, this can be achieved if each thread processes a different row and only moves east if the previous thread has a column distance of two cells (see figure 5.9). Once the two cell distance is established it does no longer cause synchronisation overhead if we assume equal processing time for each cell. Thus, synchronization overhead is

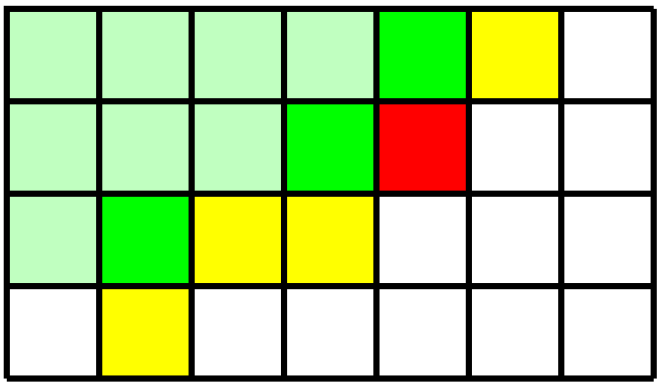

Figure 5.9: Example of parallel computation: Three cells are computed in parallel(green). The thread associated with the second row is in conflict with the first row as it tries to access the same cell (red). It will wait on the first thread to finish that cell. Needed neighbor cells without a conflict are marked yellow.

negligible if the number of processors is small compared to the edge length of the dataset.

5.6 Results

With the above mentioned techniques, we are able to rapidly reconstruct entire cities from full-featured DSMs. To evaluate our method, we used an infinite geometric error in the subsequent simplification, which leaves only the influence of our topological constraints. The experiment was performed on an intel Core 2 Quad processor with 8GB of RAM. However, the actual memory footprint only depends on the number of vertices in the current active cells as well as the cache size and cell size. For the example the memory footprint was consistently below 200MB. However, these parameters can be adapted if memory cost is an issue.

The test dataset is a $6.2km^2$ digital surface model with a resolution of $7cm$ per pixel of downtown Berlin, Germany. Figure 5.10 gives an overview of the reconstruction result of the complete dataset. The height-field, in combination with additional points for the facades, contained 1.6 billion points (35.86 GB), in which we detected 11,945 facade and 26,802 roof planes. This took 777 minutes and additional 257 minutes for the computation of the shape map. The simplification took 680 minutes, which equals 30,000 vertices per second and is thus approximately 2.15 times faster than the processing of the 4 million points dataset with the original in-core implementation which had less constraints and no parallelization. Consequently, we can state that the proposed out-of-core approach is truly scalable.

Figure 5.10: Simplification results: Whole dataset

Counting all steps together, the whole dataset was processed in 28.6 hours, which is approximately 4.6 hours per square kilometer.

Figure 5.11 illustrates the high quality of our results even for complex buildings. The left image shows the results without topological constraints. Although all buildings are recognizable using pure geometric simplification, the result is far from satisfying as important edges and corners are demolished, which causes severe visual artifacts especially when textured. In the right image we see that our proposed constraints effectively preserve feature edges with high positional accuracy.

Figure 5.7 depicts the effects of our topological filtering. The white lines visualize the underlying triangle mesh, which is due to artifacts in the height-field very fine in the left image. As the right image shows, the topological filter is able to significantly coarsen the mesh without introducing artifacts. In Figure 5.6 we see a situation where a tree grows close to a building causing a discontinuity in the detected edge of facade and ground. Additionally, a glancing intersection prevents the tree from being absorbed by the facade. As shown in the right image, the tunnel

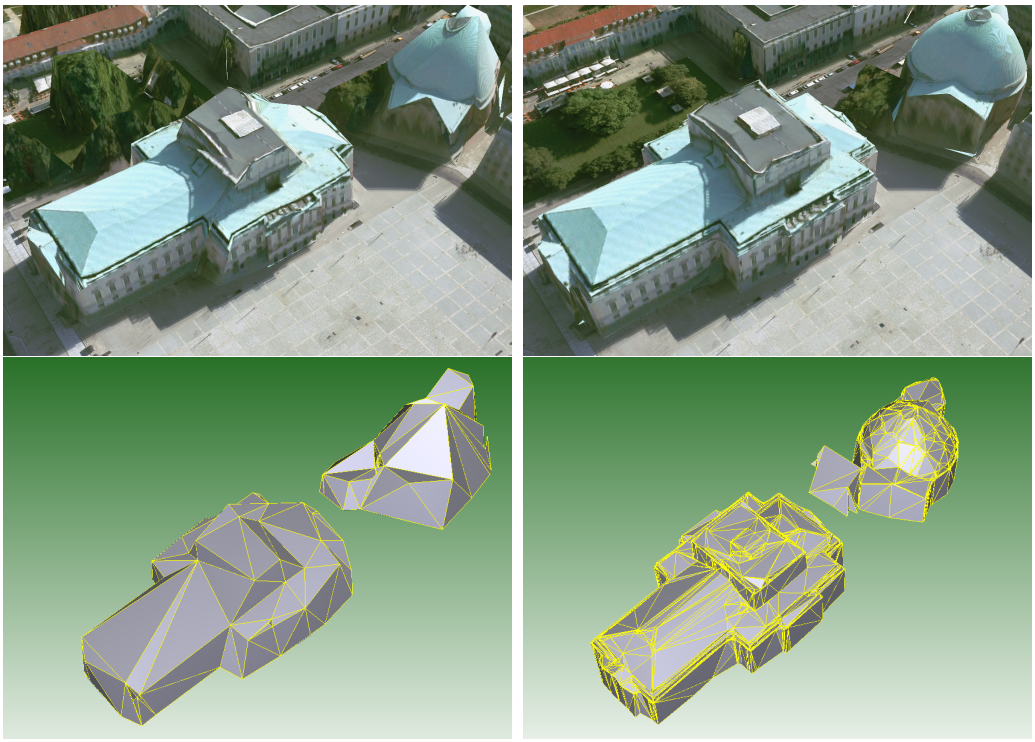

Figure 5.11: Simplification results. Left: unconstrained. Right: With topological constraints. Important roof-structures are preserved.

constraint inhibits the collapse of the undetected part of the edge causing the tree to stay separated. Unfortunately, if in such a situation the vegetation is directly connected with a buildings, as we can see at the cathedral in Figure 5.11, we are not able to remove these vegetation parts, as there is no edge to preserve.

5.7 Conclusions

We proposed a framework for rapid reconstruction of city models from full-featured DSMs. We extended the original approach with efficient out-of-core algorithms to handle huge datasets and in combination with parallel computation on multi-core CPUs are able to process a complete city in a matter of hours.

The refined topological constraints improve the already good results of the previously proposed constrained simplification especially in the presence of severe noise and errors in the height-field or only partially detected features. Nevertheless, since we not only reconstruct roof-structures but also the facades, our approach may still produce artefacts if these facades are partially not represented in the DSM.

Although the number of possible constellations in the 1-ring is finite if we consider the vertex degree bounded and only deal with up to 4 different shapes per vertex. It contradicts the simplicity of the chosen approach if we would define rules for very complex situations. After all, we rely on a greedy simplification mechanism, which will be disturbed in its effectivenes if too many interdependencies of collapses appear.

In the future we plan to extend our approach to other primitive shapes, such as spheres or cylinders, to achieve more accurate results for buildings with curved features. Another research direction will be the specification of LODs for primitive shapes and thus important features.

Acknowledgements

We thank the DLR - Institute of Robotics and Mechatronics and RSS - Remote Sensing Solutions GmbH for providing us the high-detailed dataset of the city center of Berlin. This work was partially funded by the German Research Foundation DFG as a part of the bundle project “Abstraction of Geographic Information within the Multi-Scale Acquisition, Administration, Analysis and Visualization”.

CHAPTER 6

TOWARDS SEMANTIC INTERACTION IN HIGH-DETAIL REALTIME TERRAIN AND CITY VISUALIZATION

Abstract

On the one hand, the extent, modeled detail and accuracy of virtual landscapes, cities and geospecific content is rapidly increasing. Realtime visualizations based on geometric levels-of-detail (LODs) allow the user to explore such data, but up to now, the methods of interaction are very low-level. On the other hand, we have semantic categories for the objects which are modeled in ontologies. We propose an approach which allows to combine the advantages of both, realtime visualization techniques and semantic hierarchies, in a single application without establishing an explicit link. That way, we can achieve semantic interaction without interfering with the rendering techniques which is crucial for performance on large datasets. Moreover, we are able to exchange geometric and semantic models independently of each other.

This chapter corresponds to the article [Wahl and Klein, 2007].

Keywords: DTM, DSM, city model, LOD, multiscale methods, realtime visualization, semantic interaction, ontology

6.1 Introduction

Recent advances in digitization technology and reconstruction methods have lead to the availability of huge high-resolution 2.5d digital surface models [Hirschmüller, 2005]. As sensors also record views at slightly tilted angles, to a certain degree also 3d reconstruction from aerial data is possible. Reconstructions from these aerial data will help to match and integrate data obtained from terrestrial sensors [Früh and Zakhor, 2003], so that in future we will face captured data sets of cities

which bear high detail in full 3d, i.e. huge raw point clouds with spatial resolution in the range of single centimeters. These advances in data acquisition and fusion go side by side with progress in realtime rendering methods which become capable of visualizing the ever growing data sets in full detail.

Concepts in our mind tell us that something is a house, a church, a balcony or something else depending on its appearance and our experience. We are used to perceive things with high visual detail and at the same time think about them in the compressed form of semantic categories. For efficient human computer interaction a system must match these abstract concepts of our mind, i.e. we have to close the semantic gap. To this end data corresponding to different semantic entities are labeled accordingly, which results in a semantic model.

Currently, coming along with the increasing detail of the captured data we also observe an increasing demand for semantic models. Developments in 3d GIS lead to domain-specific ontologies which are also rapidly growing in that they add more and finer semantic categories and metadata [Kolbe et al., 2005]. Semantic models based on such ontologies represent the underlying geometry in different simplified versions depending on the semantic level of detail (LOD). Details which are not semantically relevant for the model are neglected and at most represented by textures. This way the reconstruction and semantic modeling of parts of the scene that are not required for a specific ontology can be avoided which saves reconstruction time and costs. Therefore, much information contained inherently in the captured data sets is not mapped into the semantic domain and therefore not available in the semantic model. However, these details, although irrelevant from a semantic interaction point of view might still carry important cues. Examples would be the natural cover and topography in front gardens which are important for the rating of real estates, whereas on the semantic level cadastral data and building data would suffice.

Unfortunately, current terrain and city visualization systems that allow for 3d semantic interaction build on the geometry of the corresponding semantic models only and omit the additional information contained in the captured data. To achieve interactive performance, only parts of the terrain or city-models are rendered and in addition only coarse representations like extruded ground polygons on LOD1 or extruded ground polygons with roof structures on LOD2 are used [Gröger et al., 2004]. Due to the rising amount of data higher LODs like LOD3 and LOD4 would require additional multi-resolution techniques to achieve interactive performance and are therefore currently used only selectively. Combining these semantic LODs with view-dependent geometric LODs is a non-trivial problem that is currently not solved, since the geometric and semantic hierarchy must be intertwined. While from a rendering point of view representing planar facades of several neighboring buildings as a single polygon with texture is appropriate, the semantic model requires the geometric representation to respect the borders of the houses.

If semantics and geometry were not kept separate, one possible way to handle this situation would be to choose a representation based on the semantic category. The problems with this approach are firstly, that the optimal representation is not necessarily consistent throughout a semantic class and secondly, that we have to decide for every element of the raw data which category it belongs to. Also semantic categories often overlap, are ambiguous or decompose geometric entities into non-trivial subparts.

Therefore, we suggest to use different geometric representations for the semantic and geometric hierarchies of terrain and city models. The geometric representation, in the following called *rendering model*, which is actually visualized is based directly on the captured data. The representation of the semantic entities, in the following called *interaction model*, which is only used for interaction purposes is built on reconstructed and modeled data, like the above mentioned semantic LODs. These separate representations of geometric and semantic data are joined on-the-fly in interactive rendering systems. This approach enables us to implement semantically based interaction within high-resolution virtual worlds. Furthermore, it allows to combine interaction models for different ontologies with the same rendering model. Even on-the-fly exchange of ontologies can be achieved by loading different interaction models. This is especially useful as the ontology, the semantic LOD as well as the spatial extent of the semantic model can be selected dependent on the specific task. In addition, geometry data and semantic information are usually obtained and created by very different and separate processes as well as different people. Therefore, separate representations fit naturally in the corresponding graphics and GIS workflows and modeling of geometry as well as semantic information becomes substantially easier as none of them has to consider the intricacies hidden in the other, separate system. Last but not least, separate representations allow independent modification of either geometry or semantics at any time. This is especially of interest for update purposes where either the interaction model is further refined or the rendering model is updated, e.g. by adding new data.

In the following, we first concentrate on the rendering aspect. We discuss the problems arising during interactive visualization of high-definition terrain and city models, as well as the consequences for the rendering model in the next section. Then we discuss the interaction aspect in section 6.3 which describes several computer graphical methods that can be employed to connect the semantic data with the rendering model. This in turn leads to the definition of the interaction model. Some results are presented in section 6.4 before we come to a brief conclusion in section 6.5.

6.2 Realtime Terrain Rendering

6.2.1 High detail terrain and city models

For the purpose of photorealistic rendering we need a model which captures as much of the photometric and geometric details as could be perceived in the real world. Obviously, the perceivable detail depends on where the camera is placed in the virtual world. As long as the application shows the terrain from high altitude aerial views, an orthotextured digital terrain model (DTM) is appropriate. Closer to the ground, we need a digital surface model (DSM) in order to perceive the correct parallax and occlusion. Even high-detail textures mapped on a DTM will spoil the realism, due to the contradicting depth-cues (experience tells you that roofs are above ground, but the identical parallax suggests that the roof is at the same height as its surroundings).[1] For terrestrial or almost terrestrial views the 2.5d modeling approach suffers from systematic problems as facades and other steep parts of the geometry are not represented in orthotextures. Moreover, relevant geometry may be occluded from above. Thus, simply increasing the resolution of the 2.5d model will not suffice. Instead, we have to switch to a 3d model to be able to represent the scene with high precision from terrestrial perspectives as well.

6.2.2 Insufficiencies of terrain rendering

Multiresolution algorithms for fast rendering of large terrain data sets with viewpoint adaptive resolution have been an active area of research in the field of computer graphics for many years. Since giving a complete overview is beyond the scope of this paper, we refer to the surveys [Lindstrom et al., 1996; Pajarola, 2002].

Although there is a wide variety of methods regarding the details, the basic principles are always:

1. Choose a reasonable granularity for LODs. View-dependent simplification of individual primitives on-the-fly is more expensive than rendering a larger number of primitives. Batches containing a part of the model at the same LOD can be rendered more efficiently.

2. Use these batches for the choice of parts of the model which are visible for the camera (view-frustum culling) and the choice of required LOD.

3. As we deal with out-of-core datasets (i.e. data of such size that it breaks the bounds of physical memory of a computer) we need to implement a

[1] This effect becomes even more striking if seen on stereo displays, where it is noticeable even in still images.

preprocessing stage where the data is processed into a hierarchy of batches, which can then be loaded on demand.

4. Choose effective compression algorithms for the data that lessen storage and bandwidth requirements for the model without introducing performance penalties during decompression.

We base our terrain and city rendering algorithm on the terrain rendering system presented by Wahl et al. [2004, chapter 2] which is based on a quadtree data structure (see Fig. 6.1). The system has proven to be able to visualize very large terrain data sets efficiently and with high quality, e.g. data sets with a resolution up to a few centimeters for the aerial photography together with elevation models of about 1m, covering areas of hundreds of square kilometers, have already been visualized with realtime frame rates.

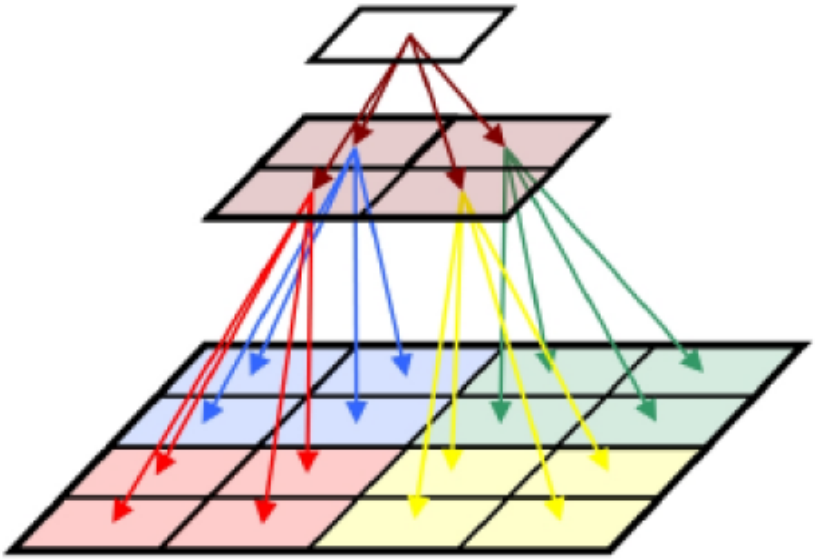

Figure 6.1: Quadtree layout of terrain rendering. Each part of the model belonging to a quadtree cell (called tile) has the same raster size independent of LOD.

In order to evaluate the different character of landscapes and cities over the scales, let us compare the number of primitives needed for a tile. In a city we may have about 20 buildings per block of size 100m $\times$ 100m which is 2,000 buildings per km^2. If a tile of a square kilometer should be pixel correct we need $\sim$8m per pixel accuracy (1km per 128pixels) which means that every building needs to be present in the data. Even if each building is modeled by a hemicube with 5 faces

only this leads to 20,000 triangles/tile just due to the buildings itself, not even accounting for the triangulation overhead at the floor and other features, like trees.

Let us now take a closer look at the root of the problem: In the presence of buildings and trees, the complexity in small tiles is comparable to that in alpine regions ($\sim$1,000),[2] but as opposed to the latter reducing the approximation accuracy does not lead to a smooth diminution of complexity. Instead, the complexity is rising to a maximum and finally breaks down when building heights fall below the necessary accuracy. As due to the LOD scheme, the screen-size of a tile remains constant, this means we get more than 1 triangle/pixel, which indicates clearly that such a mesh is the wrong representation of the data.

This example demonstrates that the assumptions stating that the complexity density of representation is roughly independent of scale, which was made for terrain models, do not hold for city data and the scalability of classical terrain models is violated.

Moreover, in 2.5d models the representation of steep geometry most prominently at facades is inappropriate. That is even if terrain visualization did work, it would need to adapt to the incorporation of 3d data.

In summary we can state that there is already a fundamental difference between rendering huge terrain data sets which are modeled as DTM and rendering even comparatively small DSMs. Given a data accuracy and density in the range below meters yields a lot of complexity in otherwise harmless (i.e. flat) data sets. The reason for this is the disproportionate distribution of features to scales. On a 20m scale only hills, rivers, shores and mountains dominate the complexity, whereas on a single meter DSM even in flat regions every tree, bush, car, building etc. leaves its high-frequency fingerprint. As a result of such details a visualization of a single city on a meter scale can be more demanding regarding the level-of-detail scheme than visualizing planet earth on a 20–30 meter basis. Moreover, the representation of the surface as 2.5d is less appropriate for high detail scales than it was for classical terrain models.

6.2.3 The rendering model

Nevertheless, on coarser levels the geometry still keeps its DTM characteristics and therefore, current highly optimized terrain rendering techniques remain the first choice. In addition, most of the earth's surface is not yet modeled other than with a DSM. The question therefore is how to represent the detailed geometry. We want to decompose the data set into a terrain model and highly detailed 3d geometry like buildings and natural cover which are added in the visualization.

[2] As measured by [Wahl et al., 2004, chapter 2] using Hausdorff-distance based mesh approximations.

This decomposition does not need to respect any kind of semantics. That is, for mere visualization performance we will not necessarily distinguish between what belongs to terrain and what not, but we want to automatically generate representations which consist of hybrid mesh, points or volume data, or any other representation which is applicable.

One way to represent such non-2.5d details is to represent them as point clouds. The rationale behind modeling with points is simple. Point clouds can be trivially extracted from any other boundary representation by surface sampling. Laser scanners, other range finders and stereo reconstruction generate points natively and the raw point cloud thus contains all details present in the data without any interpretation. An additional advantage of point clouds is that they easily represent high-frequency features. A small bump on a plane is modeled using a small pyramid in case of a triangle mesh. This however takes 4 points and 8 additional triangles (3 for the pyramid and 5 to maintain the mesh topology in the plane), whereas with points we could use a single point. Therefore point clouds have become an important means of modeling and as a consequence a lot of rendering techniques and LOD-schemes for points have been developed [Sainz and Pajarola, 2004].

Despite the apparent advantages of the point cloud representation, it leads to problems due to the high depth complexity of the city model (i.e. a high number of intersections of a half ray emanating from the view point with the model) if viewed at acute angles. In this case the number of point rendering primitives remains high although the projected screen size of the model is small, i.e. multiple points are drawn into the same pixel. In such a situation billboards, where the geometry is approximated by textured planes are advantageous [Décoret et al., 2003; Meseth and Klein, 2004]. We therefore build our rendering model on a combination of point rendering with billboards as described in [Wahl et al., 2005, chapter 3].

6.3 Interactive Visualization

In this section we will discuss how semantic information can be integrated into a visualization. Of course, the mere fact whether semantic information is available or not does not affect the visual appearance. So, the degree of semantic enrichment of a model is measured by how the user is able to interact with the model. We want to achieve an interactive application which gives the user the impression that the machine has a similar concept of the objects which are in the scene.

Apart from standard interaction features:

- the application reacts on user input through navigation
- the application renders metainformation of selected objects onto the screen

we want to be able to perform *semantic* interaction:

- the application activates objects or visualizes metadata it has linked with objects on demand (picking)
- the application emphasizes objects in the visualization based on semantic information (highlighting)
- semantics-based navigation
- the application is able to temporarily delete objects from the scene, such that the camera looks through them
- the application synthesizes objects on demand

For semantic modes of interaction, the application needs a concept to translate user queries from a semantic level to a geometric level (used for rendering images) and vice versa.

6.3.1 Implicit modeling of semantics

As we do not want to interfere with the rendering model, we cannot store semantic information directly with the original geometry. We store the missing link between the 3d rendering model and the semantic categories and metadata in the interaction model. The interaction model is linked with the rendering model implicitly by means of its spatial reference. Although this is a common technique in the 2d case (e.g. mapping thematic layers on orthophotos or topographic maps), the extension to 3d is not trivial. In the 2d case, the mapping is easily achieved by reprojecting the data into the target's coordinate system and highlighting or picking can be performed by using image overlays. In the 3d case, a 2d overlay will not suffice, since it does not discriminate along the vertical axis (e.g. windows on different floors). However, generating a full 3d overlay is no option, as it requires exactly the semantic 3d model we wanted to get rid of. Our solution to this dilemma is to use proxy objects in the interaction model. For example, in the semantic model a house can be modeled by a cube although it is a detailed point cloud in the rendering model. The link is established by performing a 3d (i.e. volume) intersection between the proxy geometry and the rendering model.

This approach effectively implements the relation between data objects and semantic entities and has the following advantages:

- There is no explicit link between the rendering model and the semantic model.
- There is no necessity to deal with the different LODs of the rendering model.

- The rendering model can be changed or replaced with no influence on the interaction capabilities.
- The interaction model can be updated dynamically without affecting the rendering, i.e. interaction models for different ontologies can be loaded on-the-fly.
- Direct compatibility with traditional 2d or 2.5d GIS data. Proxy geometry for such data can be trivially generated on-the-fly by extrusion.
- The intersection test with the proxy geometry can be evaluated on the graphics hardware (see sec. 6.3.3) which is very fast and easy to implement.
- Output sensitivity. The overhead is solely determined by the query complexity irrespective of the complexity of the rendered model. Especially, this means that without any semantic interaction we have no additional costs at all.

However, there are also some limitations:

- The rendering model must use accurate 3d coordinates, so that the volume intersection tests leads to the expected result. This constraint is obviously fulfilled by most geometry representations (triangles, points, voxels) but it does not hold for image based approaches.
- Skipping parts of the model based on semantic information (see sec. 6.3.4) is substantially more difficult to implement than with explicit links.
- As the user can interchange models, depending on the accuracy of these models and temporal changes in the meantime they can be inconsistent. This is unlikely for models derived from the same raw data sets and therefore rather a side-effect of interchanging models as a real disadvantage.

Alternatively to the implicit linking via volume intersection, we could also explicitly store the inverse relation between geometry and semantic entities. Instead of telling the geometry which semantic object it belongs to one would establish a mapping of semantic objects to geometric primitives. If this mapping is implemented with the ability to address sub-parts of a geometry (e.g. the part of a triangle belonging to a window), the model could be employed without constrainingthe simplification. The task during semantic interaction then consists of identifying the object by querying the inverse relation which is admittedly less efficient. Although the direct influence of the semantics on the preparation of the model is remedied, there is still a close explicit link between semantic and rendering model: In order to address the geometry of an object we need detailed information on how the model

is represented. Moreover, a representation which lends itself to efficient rendering might still be very complex when we have to address parts for example in point clouds or billboards.

6.3.2 Interaction model

As mentioned earlier, the interaction model represents the semantic hierarchy and therefore its underlying ontology within the visualization application. It is composed of the actual semantic information i.e. categories and metadata as well as the proxy geometry. Apart from the difference in the geometric representation any type of semantic model can be directly used as interaction model, e.g. CityGML [Kolbe et al., 2005]. Therefore, we will not discuss the semantic features here, but focus on the requirements for the proxy geometry.

Regarding the representation, the proxy geometry should consist of well-defined solids, ideally in form of an oriented boundary representation as these can be used directly for rendering. Apart from that there are three observations which influence the design of such proxies:

1. Interaction may take place on a coarser level of detail than visualization. Inaccuracies in the range of pixels would compromise the visual quality, but human interface devices are rarely placed with pixel accuracy.

2. Interaction models remain hidden from the user. As highlighting is not implemented as a 2d overlay, we can change the appearance of the model by evaluating the 3d proximity to the semantic model, thus the proxy geometry carrying the semantic information is an invisible layer.

3. Picking results are predictable even with coarse interaction models. The predictability of the picking is owed to the accuracy and simplicity of the proxies. Buildings and related objects in cities can mostly be well described by simple features like a small number of boxes or polyhedra. Moreover, an immediate feedback to the user can be used to verify whether the active object in the interaction model matches the intention of the user.

As a consequence, we do not need to model the proxies with the same detail as a representation intended for rendering. Actually, we just need to ensure that the object under consideration is within the volume and no other object intersects the volume. As we are dealing with models of bounded accuracy, this implies that we should inflate the proxy solids a bit in order to avoid that parts of the rendering model protrude the volume.

For performance reasons, the proxy geometries should also be organized in a spatial hierarchy like the quadtree mentioned earlier. This will improve performance when a large number of highlighted objects is not within view and can

therefore be skipped. Activating only parts of the interaction model, based on the specific application, will further accelerate interaction in the case of very complex models.

6.3.3 Implementation issues

The implementation of semantic interaction based on implicit mapping using proxy geometry is straightforward. For highlighting, however, there is an optimized implementation which exploits graphics hardware.

The highlighting works as follows:

1. Render the whole scene. This sets the colors of the frame-buffer (the buffer which is displayed on the screen) but also the corresponding z-buffer (an off-screen buffer which records the distance of each drawn pixel to the camera, used for correct occlusion).

2. Render the active object into the stencil buffer (a special-purpose auxiliary buffer which can be efficiently queried and updated during rendering):

   a) Disable depth-buffer and frame-buffer writes. Set stencil operation to decrease stencil for pixels less distant than the z-buffer (z-pass) and render back-facing polygons of the proxy.

   b) Set stencil operation to increase stencil on z-pass and render front-facing polygons of the proxy.

3. Now the stencil is positive in all pixels which have depth values within the proxy geometry. Activate frame-buffer write, blending for transparency and stencil test. Render screen-sized quad with the highlight color.

As only the colors are changed, it is clear that this process can be applied to any number of objects simply by iterating phases 2 and 3. In fact it suffices to simultaneously perform step 2 for all objects with the same highlight color.

Picking can either be implemented by using the highlighting feature to render unique IDs into an auxiliary buffer, which then tells which objects cover which part of the screen or by performing a z-readback (transferal of the depth-buffer from graphics hardware to main memory) which yields the 3d point at the given screen coordinates. This point can then be transformed into a spatial query to the semantic model.

6.3.4 Advanced interaction

While the picking and highlighting features are easy to implement because they only rely on information gathered in the hardware buffers of the graphics processing

unit (GPU), tasks such as looking through (i.e. deleting) objects are more intricate. However, in many scenarios of future 3d-GIS applications such features may be especially worthwhile. Although omitting objects from rendering is a lot easier when we have direct access to the model, there are still possibilities to achieve the same result without touching the rendering model.

The problem with the transparency feature is that we need to know what would have been seen if the occluding (deleted) object was not there. Of course, we can apply our highlighting framework to determine which fragments[3] of the image are the false occluders, but as our output consists only of 2d (color and depth) buffers there is no way to access information behind these. The obvious way to circumvent this problem is to modify the rendering such that only fragments not within the proximity of deleted objects are rendered. This can be achieved using the hardware-support on modern GPUs by either performing clipping of the rendering primitives in the geometry shader or by discarding fragments in the fragment shader based on an implicit representation of the proxy geometry. Clipping can be very efficient for convex polytopes as the in/out status can be established using a few halfspace inclusion tests. The disadvantages of this approach are that it relies on computing capabilities only available on very recent hardware, the computation needs to be done every frame and the number of halfspace intersections will introduce a performance penalty, if many or complex objects are to be hidden. Preferably, the fragments should be discarded based on a single lookup. The stencil lookup however is insufficient, as it has no depth value. We can render the hidden proxies into an auxiliary z-buffer, but there we could only store one depth per fragment. In a 3d scene, however a high depth complexity (many primitives project to the same pixel) can be present especially when rendering terrain from almost terrestrial views. Therefore we would need to compute a list of depth intervals for each pixel which correspond to the different parts of the hidden proxies. Generating such lists is not feasible on current hardware.

Therefore we propose to use a technique similar to shadow mapping [William, 1978]. We make use of the fact, that city models still have a 2.5d character. Thus, the complexity in vertical direction will generally be bounded by a small constant. So instead of storing the visibility information in the screen domain we will use a map-projection and store height values instead. The remaining z-complexity of the proxies' volume can be accounted for by storing multiple upper and lower contours. This approach has the additional advantage, that the visibility information remains the same for many frames and does not have to be recomputed. Recomputation is only necessary when and where objects change their invisibility status. In order to exploit this locality we will bind these auxiliary invisibility maps to the tiles of the

[3] A fragment denotes the data necessary to generate a pixel in the frame buffer. This includes raster position, depth, color, stencil, etc.

geometric LOD-hierarchy. Technically, the algorithm for hiding objects works in two parts:
Initialization of new invisible objects (if applicable):

- identify tiles of new objects
- initialize contour buffers
- set map projection
- render front/back faces in upper/lower contour buffer respectively

Scene rendering (every frame):

- reproject each fragment into map coordinates (gives height above ground)
- compare height to contour heights
- if height is within an upper/lower contour pair kill fragment

Note that again the additional complexity correlates with the query complexity (complexity of invisible objects) and does not depend on the scene, such that true output-sensitivity is maintained.

6.4 Results

We implemented the proposed scheme for semantic interaction within a realtime 2.5d terrain visualization framework to demonstrate the applicability. The rendering model is a $28cm$ resolution DSM colored with orthophotos of $7cm$ resolution. This model is courtesy of German Aerospace Center (DLR) – Institute of Robotics and Mechatronics and was derived by semi-global matching [Hirschmüller, 2005] from Vexcel Ultracam™ imagery. The semantic model was derived from "CityGML reference data on Pariser Platz" freely available for download[4]. The boundary representation of the solids modeling the semantic entities was used as the above mentioned proxy geometry.

Figure 6.2 shows a rendering of a part of Berlin with the highlighting feature. The extension of Hotel Adlon was selected by the user and is thus highlighted. In figure 6.3 the picking feature is demonstrated: As the mouse hovers over the image during realtime exploration the semi-transparent highlighting provides an instant feedback which objects are present in the active semantic category. With these objects we can interact by accessing meta-information or highlighting them as seen above. The glass roof of this building (Eugen-Gutmann Haus) is not highlighted because it is not within the volume defined by the semantic model which demonstrates that this mode of interaction is truly 3d.

[4] http://www.3d-stadtmodell-berlin.de/

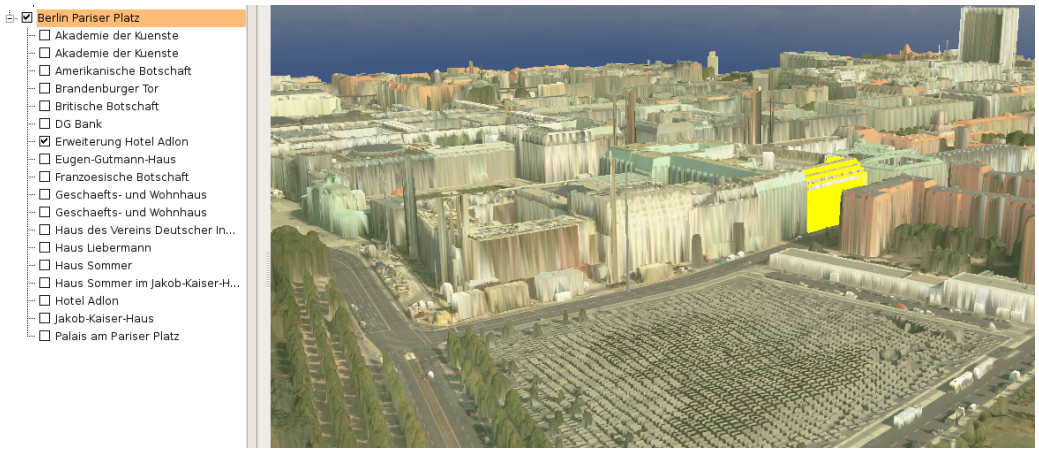

Figure 6.2: Screenshot from the rendering application showing a highlighted feature of the semantic dataset within a visualization of the Berlin DSM.

6.5 Conclusion

In this work we introduced methods to allow semantic based interaction in highly detailed 3d terrain- and city models. In the proposed approach different ontologies can be used without changing the geometric LOD-hierarchy used for photorealistic rendering. In our opinion, the ability to switch the ontologies and thus the categories in which we think about the data is more important than the exact correspondence between geometric features and semantic entities. We have presented methods which allow interaction with photorealistic visualizations, so that the missing explicit link becomes unnoticable to the user.

6.5.1 Future Work

The ability to access detailed semantic information within a real-time photorealistic rendering framework is just a first step towards semantic interaction. With the combined strength of detailed semantic models and visually detailed instances there is a lot of place for new interaction paradigms especially concerning the modeling or synthetization of such scenes but also the exploration and visualization techniques.

Another direction of research will concern the flexibility of models. By now, all successful realtime rendering methods have to preprocess the data. Apart from the delay which is introduced by doing so, there is a conceptual difference whether the data needs to be static or it can be adapted on-the-fly.

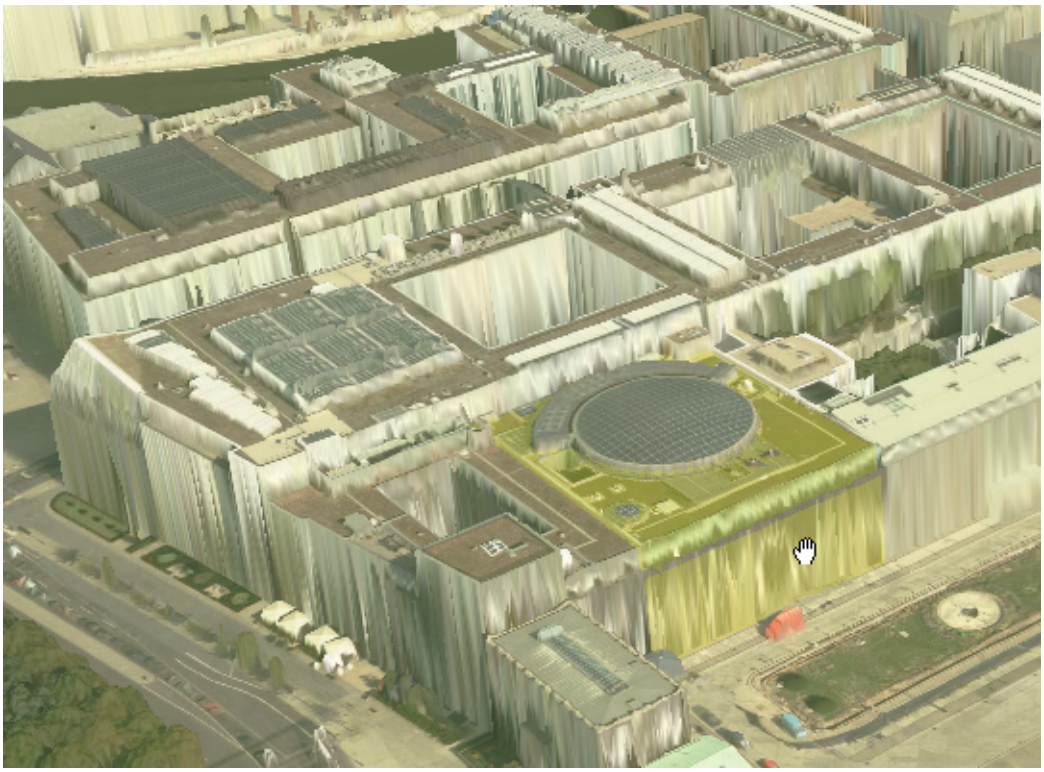

Figure 6.3: Screenshot from the rendering application. In picking mode semantic entities are highlighted as the mouse hovers over them in the image. The user thus gets an instant feedback with which objects and on which semantic LOD he is about to interact.

Acknowledgments

This work was funded by the German Science Foundation (DFG) as part of the bundle project “Abstraction of Geographic Information within the Multi-Scale Acquisition, Administration, Analysis and Visualization”. We thank the Berlin Senate and Berlin Partner GmbH for making the CityGML model available. Finally, we also want to thank the reviewers for helpful comments.

CHAPTER 7

CONCLUSION AND FUTURE DIRECTIONS

This work focussed on the topic of efficient rendering and interaction with terrain and city models. The challenges in this area are the immense size and complexity of the raw input data and to enable an intuitive, efficient user experience during interaction.

In this direction, I have examined the following aspects:

- scalable LOD methods for photorealistic realtime rendering resulting in a hierarchical structure of models tuned for rendering
- in the field of reconstruction: abstraction of raw input models, to reflect their inherent structure better.
- basic techniques to augment the purely geometric/radiometric visual models with arbitrary additional semantic data.

I will now shortly recall the main findings of the preceding chapters, mention their further development and indicate future directions.

7.1 Rendering of Digital Surface Models

The research presented in chapter 2 has proven that it is possible to render huge out-of-core terrain datasets in realtime with photorealistic detail. The quadtree hierarchy equipped with precomputed mesh approximations of known accuracy and corresponding texture images is the basis of the realtime rendering method. The overheads of traversing the quadtree and auxilliary computations are so small that the costs of rendering a frame are dominated by the visible part of the model and thus scalability is guaranteed. At the same time, the strict approximation bounds based on the Hausdorff distance render the LOD switches almost unnoticable, which greatly enhances realism. The individual tiles of the quadtree can be

compressed very efficiently, which is a bonus in low-bandwidth scenarios such as internet based streaming.

While back in 2003, the largest available model consisted of about 256 million height samples and the maximal ground resolution was 2m, in the meantime I was able to process digital surface models with up to 10cm resolution and up to 20 billion height samples. Moreover, the rendering application based on the presented approach is still capable to render such datasets at realtime framerates.

The approach has been transferred to the enterprise 3D RealityMaps, where high resolution data of most of the Eastern Alps was acquired and preprocessed using my proposed method. Together with the corresponding viewer application, those regions are available as streaming terrain datasets at `http://www.outdoor-guides.de/tourenplaner`. The scalability of my method has even been established on thin clients, culminating in the release of the first public smartphone app with a high-detail 3d map of the Alpine Ski World Championships at Schladming in 2013.

Scientifically, there are three directions which seem fruitful:

Improved models. The most prominent defect in 2.5d modeling is the lack of texture information on steep parts. Although in terrain such features are very rare, they often represent very interesting parts or even famous landmarks. Schneider and Klein [2008] show how to semi-automatically add photos to the terrain surface. They are able to texturize also steep parts by using a reparameterization of the corresponding tiles.

Another issue of purely geometric simplification is the lack of structure-awareness. I address this in chapters 4 and 5, where I refine the LOD approach regarding the representation of models at coarser LODs in order to get semantically improved models.

Improved rendering. There are several ways to increase the photorealism of the rendered images. Rendering with textures is based on the assumption that the whole surface is lambertian diffuse, i.e. takes the same color from all perspectives. Especially for cities, where the roofs dominate the orthogonal perspective but facades become visible in slanted perspectives, this assumption can not be transferred to the simplified models. Ruiters [2008] presents a method to model coarser LODs with a compact BTF model instead of diffuse textures.

Further possible improvements in this direction are models for atmospheric scattering or rendering with high-dynamic range and tone-mapping.

Towards 3d GIS. The high accuracy of my method maintains georeferences exactly and is therefore well suited for GIS applications. Consequently, it has

been used in [Goncharova et al., 2007] for the visualization of flows and seismic profiles in the Baltic Sea, and in [Moder et al., 2008; Taubenböck et al., 2009] for the visualization of Tsunami hazard and disaster management.

7.2 Planes in Point Clouds

In chapter 3 I have presented a hierarchical LOD scheme for point clouds which renders high-detailed datasets much more efficiently than classical point rendering methods, without sacrificing details. I use a hybrid point polygon representation, combining the advantages of the two. While points are well suited for salient details, polygons are very efficient for planar structures. In order to arrive at such a hybrid representation, I have developed a RANSAC based plane detection which identifies large, compact and densly covered point subsets and represents them as textured rectangles.

I combine the hybrid representation with an octree hierarchy which defines the LODs of the model along with a required accuracy. Moreover, it directly subdivides the out-of-core input point cloud into manageable parts with bounded complexity, as I use vertex clustering and quantization in coarser nodes to presimplify the data. For each node of the octree, a hybrid representation is generated independently. The LOD of the node, which matches its depth in the octree, defines the necessary accuracy.

As experiments with two large point clouds demonstrate, the proposed method outperforms purely point-based rendering by almost an order of magnitude during rendering. Apart from that the RANSAC plane detection is very fast, taking only as much time as the straight-forward octree construction. It proved also very robust with respect to noise and outliers, so that it can be applied to raw Lidar data. The texture-based representation very well suits standard image compression techniques, which is a further advantage.

The RANSAC based plane detection was later on extended to a wider range of primitive shapes in [Schnabel, Wahl, and Klein, 2007a]. Apart from planes which are linear surfaces also a range of quadratic surfaces as balls, cylinders and cones and even tori were fitted to the point data. An adaption of the extended algorithm to an out-of-core city dataset was presented in [Schnabel et al., 2007b]. Based on these additional primitives, Schnabel et al. [2008a] have also refined the compression and rendering aspects of the shape-detection approach.

Heading in the direction of semantic models we also presented a method for finding shape configurations in point clouds [Schnabel, Wessel, Wahl, and Klein, 2008b]. Thus, the low-level shape detection in point clouds is indeed a valuable basis for both, efficient scene representation on the one hand and scene analysis on the other hand.

7.3 Semantically improved modeling

In chapters 4 and 5 I introduced a constrained simplification approach which aims at preserving relevant structures. The semantically relevant structures like walls, roofs, and dormers are mostly represented in their corresponding planes. The proposed constraints are based on the local constellations of planes and thus implicitly carry the structural information. I showed that a simple greedy simplification approach augmented with these constraints effectively reduces model complexity while maintaining important feature edges and corners. Although in real data sets there are many problems related to partial occlusion, noisy data etc., the robust plane detection and the carefully defined constraints yielded pleasant results for a large high-detail dataset of Berlin.

The parameters of the plane detection algorithm greatly influence the resulting structures. So without modification of the general approach, one direction of further research is to experiment with the shape information.

In this respect I already mentioned the possibility of introducing further primitive shapes. As architecture makes use of curved surfaces e.g. in domes, cylindrical roofs, or conic steepletops, it is desirable to recognize them. However, as curved surfaces are not perfectly representable by polygons, either more complex representations are required or approximating triangulations need to be defined. While the latter would be very similar to the result of plane detections, the more complex surface models are not directly suited for many applications. In a combination with LOD techniques though, they could be instantiated at any desired resolution, which would be a a great advantage.

So, another direction of research focusses on the LOD aspect of the semantic structure. If applied in the context of my approach, an important task consists in finding the optimal parameters for the shape detection so that the result reflects the city data at a specific meaningful LOD. Such meaningful LODs can not be determined objectively. Although there have been attempts at standardizing semantically motivated LODs for city models [Kolbe et al., 2005], these discrete LODs generally fail to cover the full range of useful representations of complex architecture. A more fine-grained range of LODs and corresponding geometric representations would be desirable.

The just recently presented approach of Verdie, Lafarge, and Alliez [2015] is an example for this direction of semantic LODs and shows that the topic of finding semantically improved models for cities is still an active area of research.

7.4 Semantic Interaction

The previous topics focussed on the efficient design, construction and use of LOD models. Chapter 6 deals with the fundamental problem of adding semantic information to such LOD models aiming at semantic interaction.

I showed that it is useful to separate the interaction model from the rendering model. Instead of reorganizing the rendering model in semantic sub-parts, I implicitly link semantic information to the rendering model via their common spatial reference. In analogy to layered maps in 2d, I use spatially referenced 3d proxy objects, which are used to establish the link to the 3d model at rendering time.

Apart from the obvious advantage that there is no need to label the parts of the rendering model according to their semantic categories, the proposed approach offers a range of further opportunities. It becomes possible to exchange either the interaction model or the rendering model without touching the other. The additional complexity of the interaction model does not influence the efficiency of the rendering model unless when it is used to query semantics. In that case, the costs for the queries are determined only by the interaction model and not by the more complex rendering model.

Once the semantic categories are available, interaction methods can be based on them. The most basic methods to use semantics in interactive rendering are highlighting and picking, which correspond to the forward and backward mapping from semantic entities to pixels, respectively. I implemented the highlighting feature using hardware-accelerated volume intersection based on stencil buffers and demonstrated that real-time picking and highlighting is indeed feasible within a complex terrain dataset.

In future 3d-GIS applications, many further semantic interaction techniques could be useful. Such more refined methods of semantic interaction include navigation, deformation and model editing:

Navigation. A very common interaction technique within complex scenes is the fly-to feature, where the user clicks on the screen and is steered towards the chosen coordinate. With picking, the application can understand the clicked object and better match the expectation of the user (e.g. fly in view of the whole mountain instead of close to the summit). Furthermore, experts could define a standard way to present certain objects (e.g. always steer the camera to the entrance of a house).

Space deformation. To achieve better orientation of the user, important parts of the scene can not only be highlighted, but also geometrically emphasized. Möser, Degener, Wahl, and Klein [2008] introduced several methods of space

deformation which target at improved user navigation in a city. In that work we present the combination of different perspectives, as worm's-eye view and bird's-eye view, the slanting of building facades, which are more useful for orientation than roofs, and importance-based scaling imitating the effect of a looking glass. These techniques can be implemented by space deformation and as such need not change the rendering model directly. Instead we can use vertex shader programs to achieve hardware assisted on-the-fly deformations.

Model editing. While in general, new or modified objects can easily be added to empty parts of a scene, difficulties arise when they should replace existing objects. So, a fundamental feature of model editing is the omission of objects. Unfortunately, this is not easily achievable without modifying the rendering model, as occluded parts of the scene cannot be reconstructed. I sketched a shader-based approach in sec. 6.3.4 which uses a map lookup similar to 2d layers.

But actually, for efficient model editing, we need more flexible rendering models. Ideally, they should support operations like the replacement of objects directly. For example to exchange abandoned industrial sites by freshly planned residential blocks would be an important application in city planning. This leads us back to the topics of the first two chapters, where I described efficient LOD schemes for large datasets. Thus, a further direction of research is the design of schemes that efficiently incorporate model changes. Thinking about change, however, leads to the general concept of a fourth dimension and the modeling of 3d space depending on its position on the timeline.

BIBLIOGRAPHY

PANKAJ K. AGARWAL AND SUBHASH SURI. Surface approximation and geometric partitions. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1994. 43

C. ANDÚJAR, P. BRUNET, A. CHICA, J. ROSSIGNAC, I. NAVAZO, AND A. VINACUA. Computing maximal tiles and application to impostor-based simplification. *Computer Graphics Forum*, 23(3):401–410, 2004. 29

H. AREFI, J. ENGELS, M. HAHN, AND H. MAYER. Level of detail in 3d building reconstruction from lidar data. *ISPRS Congress Beijing 2008, Proceedings of Commission III*, 37:485–490, 2008. 55

M. BOTSCH AND L. KOBBELT. High-quality point-based rendering on modern GPUs. In *Proceedings of Pacific Graphics 03*, 2003. 28

M. BOTSCH, M. SPERNAT, AND L. KOBBELT. Phong splatting. In *Proceedings of Symposium on Point-Based Graphics 04*, 2004. 28

BAOQUAN CHEN AND MINH XUAN NGUYEN. POP: A hybrid point and polygon rendering system for large data. In *IEEE Visualization*, 2001. 28

P. CIGNONI, C. ROCCHINI, AND R. SCOPIGNO. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998. 44

PAOLO CIGNONI, ENRICO PUPPO, AND ROBERTO SCOPIGNO. Representation and visualization of terrain surfaces at variable resolution. *The Visual Computer*, 13(5):199–217, 1997. 13

PAOLO CIGNONI, FABIO GANOVELLI, ENRICO GOBBETTI, FABIO MARTON, FEDERICO PONCHIO, AND ROBERTO SCOPIGNO. BDAM – batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum*, 22(3):505–514, September 2003a. URL `http://www.crs4.it/vic/cgi-bin/bib-page.cgi?id='Cignoni:2003:BBD'`. 14

PAOLO CIGNONI, CLAUDIO MONTANI, CLAUDIO ROCCHINI, AND ROBERTO SCOPIGNO. External memory management and simplification of huge meshes. *IEEE Transactions on Visualization and Computer Graphics*, 9:525–537, 2003b. ISSN 1077-2626. doi: http://doi.ieeecomputersociety.org/10.1109/TVCG.2003.1260746. 44

DAVID CLINE AND PARRIS K. EGBERT. Terrain decimation through quadtree morphing. *IEEE Transactions on Visualization and Computer Graphics*, 7(1): 62–69, 2001. 13

J. COHEN, D. ALIAGA, AND W. ZHANG. Hybrid simplification: Combining multi-resolution polygon and point rendering. In *IEEE Visualization 2001*, pages 37–44, 2001. 28

DANIEL COHEN-OR AND YISHAY LEVANONI. Temporal continuity of levels of detail in delaunay triangulated terrain. In *Proceedings of the 7th conference on Visualization '96*, pages 37–42. IEEE Computer Society Press, 1996. ISBN 0-89791-864-9. 14

LEILA DE FLORIANI AND ENRICO PUPPO. A hierarchical triangle-based model for terrain description. In A. U. FRANK, I. CAMPARI, AND U. FORMENTINI, editors, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, pages 236–251, Berlin, 1992. Springer. 14

P. DECAUDIN AND F. NEYRET. Rendering forest scenes in real-time. In *Proceedings of EG Symposium on Rendering*, pages 93–102, 2004. 29

X. DÉCORET, F. DURAND, F. SILLION, AND J. DORSEY. Billboard clouds for extreme model simplification. In *SIGGRAPH 2003, Computer Graphics Proceedings*, pages 689–696, 2003. 29, 33, 77

MATHIEU DESBRUN, MARK MEYER, PETER SCHRÖDER, AND ALAN H. BARR. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-48560-5. doi: 10.1145/311535.311576. URL http://dx.doi.org/10.1145/311535.311576. 45

TAMAL K. DEY AND JAMES HUDSON. PMR: Point to mesh rendering, a feature-based approach. In *IEEE Visualization 2002*, pages 155–162, 2002. 29

JÜRGEN DÖLLNER, KONSTANTIN BAUMANN, AND KLAUS HINRICHS. Texturing techniques for terrain visualization. In *IEEE Visualization*, pages 227–234, 2000. 14

PETER DORNINGER AND NORBERT PFEIFER. A comprehensive automated 3d approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds. *Sensors*, 8(11):7323–7343, 2008. ISSN 1424-8220. doi: 10.3390/s8117323. URL http://www.mdpi.com/1424-8220/8/11/7323. 56

MARK A. DUCHAINEAU, MURRAY WOLINSKY, DAVID E. SIGETI, MARK C. MILLER, CHARLES ALDRICH, AND MARK B. MINEEV-WEINSTEIN. ROAMing terrain: real-time optimally adapting meshes. In *IEEE Visualization*, pages 81–88, 1997. 13, 14, 19

JIHAD EL-SANA AND YI-JEN CHIANG. External memory view-dependent simplification. *Computer Graphics Forum*, 19(3):139–150, 2000. 44

WILLIAM S. EVANS, DAVID G. KIRKPATRICK, AND G. TOWNSEND. Right-triangulated irregular networks. *Algorithmica*, 30(2):264–286, 2001. 13

MARTIN A. FISCHLER AND ROBERT C. BOLLES. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 30

CHRISTIAN FRÜH AND AVIDEH ZAKHOR. Constructing 3d city models by merging aerial and ground views. *IEEE Computer Graphics & Applications*, 23(6):52–61, 2003. ISSN 0272-1716. doi: http://dx.doi.org/10.1109/MCG.2003.1242382. 71

MICHAEL GARLAND AND PAUL S. HECKBERT. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, New York, NY, USA, 1997a. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7. doi: http://doi.acm.org/10.1145/258734.258849. 57

MICHAEL GARLAND AND PAUL S. HECKBERT. Surface simplification using quadric error metrics. *Computer Graphics*, 31(Annual Conference Series): 209–216, 1997b. 22, 43, 44, 45

MICHAEL GARLAND AND ERIC SHAFFER. A multiphase approach to efficient surface simplification. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 117–124, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7803-7498-3. 44

T. GERSTNER. Multiresolution compression and visualization of global topographic data. *GeoInformatica*, 7(1):7–32, 2003. URL http://wissrech.iam.uni-bonn.de/research/pub/gerstner/globe.pdf. (shortened version in Proc. Spatial Data Handling 2000, P. Forer, A.G.O. Yeh, J. He (eds.), pp. 14-27, IGU/GISc, 2000, also as SFB 256 report 29, Univ. Bonn, 1999). 13

ENRICO GOBBETTI AND FABIO MARTON. Far voxels - a multiresolution framework for interactive rendering of huge complex 3d models on commodity graphics platforms. *ACM Transactions on Graphics*, 24(3):878–885, 2005. 30

NATALIA GONCHAROVA, PAVEL BORODIN, AND ALEXANDER GRESS. Gis for planning, navigation acquisition and visualization of results for the study of chemical munition dumpsites in the baltic sea. In Y. CHEN, I. CLUCKIE, AND K. TAKARA, editors, *2nd International Conference of GIS/RS in Hydrology, Water Resources and Environment (ICGRHWE '07)*, September 2007. 89

G. GRÖGER, T. H. KOLBE, R. DREES, A. KOHLHAAS, H. MÜLLER, F. KNOSPE, U. GRUBER, AND U. KRAUSE. Das interoperable 3d-stadtmodell der SIG 3d der GDI NRW. http://www.ikg.uni-bonn.de/fileadmin/sig3d/pdf/Handout_04_05_10.pdf (15.07.2007), 2004. 72

M. H. GROSS, R. GATTI, AND O. STAADT. Fast multiresolution surface meshing. In *Proceedings of the IEEE Visualization '95*, pages 135–142. IEEE Computer Society Press, 1995. URL ftp://ftp.inf.ethz.ch/pub/publications/tech-reports/2xx/230.ps. 13

J. P. GROSSMAN AND W. J. DALLY. Point sample rendering. In *Proceedings of the 9th EG Workshop on Rendering*, pages 181–192, 1998. 28

STEFAN GUMHOLD AND WOLFGANG STRASSER. Real time compression of triangle mesh connectivity. In MICHAEL COHEN, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 133–140. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8. 19

MICHAEL GUTHE, PAVEL BORODIN, ÁKOS BALÁZS, AND REINHARD KLEIN. Real-time appearance preserving out-of-core rendering with shadows. In A. KELLER AND H. W. JENSEN, editors, *Rendering Techniques 2004 (Proceedings of Eurographics Symposium on Rendering)*, pages 69–79 + 409. Eurographics Association, June 2004. ISBN 3-905673-12-6. 28, 30, 35

MICHAEL GUTHE, PAVEL BORODIN, AND REINHARD KLEIN. Fast and accurate hausdorff distance calculation between meshes. *Journal of WSCG*, 13(2): 41–48, February 2005. ISSN 1213-6972. 44

KLAUS HILDEBRANDT AND KONRAD POLTHIER. Anisotropic filtering of non-linear surface features. *Computer Graphics Forum*, 23:391–400, 2004. 45

HEIKO HIRSCHMÜLLER. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 807–814, 06 2005. ISBN 0-7695-2372-2. doi: http://dx.doi.org/10.1109/CVPR.2005.56. 71, 83

HEIKO HIRSCHMÜLLER. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30 (2):328–341, Feb. 2008. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1166. 55

HUGUES HOPPE. Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 99–108, New York, NY, USA, 1996. ACM. ISBN 0-89791-746-4. URL http://doi.acm.org/10.1145/237170.237216. 13

HUGUES HOPPE. 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., 1997. ISBN 0-89791-896-7. doi: http://doi.acm.org/10.1145/258734.258843. 14

HUGUES HOPPE, TONY DEROSE, TOM DUCHAMP, JOHN MCDONALD, AND WERNER STUETZLE. Mesh optimization. *Computer Graphics*, 27(Annual Conference Series):19–26, 1993. 43

HUGUES H. HOPPE. Smooth view-dependent level-of-detail control and its application to terrain rendering. In DAVID EBERT, HANS HAGEN, AND HOLLY RUSHMEIER, editors, *IEEE Visualization '98*, pages 35–42, 1998. 13, 14, 16, 44

K. INOUE, T. ITOH, A. YAMADA, T. FURUHATA, AND K. SHIMADA. Clustering a large number of faces for 2-dimensional mesh generation. In *Proceedings of 8th International Meshing Roundtable*, pages 281–292, 1999. 29

MARTIN ISENBURG AND STEFAN GUMHOLD. Out-of-core compression for gigantic polygon meshes. In *SIGGRAPH 2003, Computer Graphics Proceedings*, pages 935–942, 2003. 27

MARTIN ISENBURG, PETER LINDSTROM, STEFAN GUMHOLD, AND JACK SNOEYINK. Large mesh simplification using processing sequences. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 61,

Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-2030-8. doi: http://dx.doi.org/10.1109/VISUAL.2003.1250408. 44

S. JESCHKE AND M. WIMMER. Textured depth meshes for real-time rendering of arbitrary scenes. In *Proceedings of the 13th EG Workshop on Rendering*, pages 181–190, 2002. 29

A. KALVIN AND R. TAYLOR. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications*, 16(3):65–77, 1997. 29

REINHARD KLEIN AND ANDREAS SCHILLING. Efficient multiresolution models. In ANDREAS SCHILLING, editor, *Festschrift zum 60. Geburtstag von Wolfgang Straßer*, pages 109–130. Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 2001. 14

REINHARD KLEIN, GUNTHER LIEBICH, AND WOLFGANG STRASSER. Mesh reduction with error control. In RONI YAGEL AND GREGORY M. NIELSON, editors, *IEEE Visualization '96*, pages 311–318, 1996. 16, 43

THOMAS H. KOLBE, GERHARD GRÖGER, AND LUTZ PLÜMER. CityGML – interoperable access to 3d city models. In PETER VAN OOSTEROM, SISI ZLATANOVA, AND E. M. FENDEL, editors, *Proc. of the 1st International Symposium on Geo-information for Disaster Management*, 2005. 72, 80, 90

JOSHUA LEVENBERG. Fast view-dependent level-of-detail rendering using cached geometry. In *IEEE Visualization 2002*, pages 259–266, 2002. 14

M. LEVOY AND T. WHITTED. The use of points as display primitives. Technical report, CS Departement, University of North Carolina at Chapel Hill, 1985. 27

M. LEVOY, K. PULLI, B. CURLESS, S. RUSINKIEWICZ, D. KOLLER, L. PEREIRA, M. GINZTON, S. ANDERSON, J. DAVIS, J. GINSBERG, J. SHADE, AND D. FULK. The digital michelangelo project: 3d scanning of large statues. In *Siggraph 2000, Computer Graphics Proceedings*, pages 131–144, 2000. 28

P. LINDSTROM, D. KOLLER, W. RIBARSKY, L. HODGES, N. FAUST, AND G. TURNER. Real-time continuous level of detail rendering of height fields. *Proceedings of SIGGRAPH'96*, pages 109–118, 1996. URL `citeseer.ist.psu.edu/lindstrom96realtime.html`. 13, 14, 19, 74

PETER LINDSTROM. Out-of-core simplification of large polygonal models. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 259–262, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. ISBN 1-58113-208-5. doi: http://doi.acm.org/10.1145/344779.344912. 44

PETER LINDSTROM AND VALERIO PASCUCCI. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):239–254, July–September 2002. 13

PETER LINDSTROM AND CLÁUDIO T. SILVA. A memory insensitive technique for large model simplification. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 121–126, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7803-7200-X. 44

BRANDON LLOYD AND PARRIS K. EGBERT. Horizon occlusion culling for real-time rendering of hierarchical terrains. In *IEEE Visualization 2002*, pages 403–409, 2002. 21

KOK-LIM LOW AND TIOW-SENG TAN. Model simplification using vertex-clustering. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 75–ff., New York, NY, USA, 1997. ACM. ISBN 0-89791-884-3. doi: http://doi.acm.org/10.1145/253284.253310. 44

HANS-GERD MAAS AND GEORGE VOSSELMANN. Two algorithms for extracting building models from raw laser altimetry data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2/3):153–163, 1999. 55

P. MACIEL AND P. SHIRLEY. Visual navigation of large environments using textured clusters. In *ACM Symposium on Interactive 3D Grapics*, pages 95–102, 1995. 29

JAN MESETH AND REINHARD KLEIN. Memory efficient billboard clouds for BTF textured objects. In *Vision, Modeling, and Visualization 2004*, pages 167–174, 2004. 30, 77

F. MODER, M. OCZIPKA, F. SIEGERT, F. LEHMANN, Y. S. DJAJADIHARDAJA, REINHARD KLEIN, AND ROLAND WAHL. Last-mile—large-scale topographic mapping of densely populated coasts in support of risk assessment of tsunami hazards. In *International conference on tsunami warning systems (ICTW), Bali, Indonesia*, November 2008. 89

SEBASTIAN MÖSER, PATRICK DEGENER, ROLAND WAHL, AND REINHARD KLEIN. Context aware terrain visualization for wayfinding and navigation. *Computer Graphics Forum*, 27(7):1853–1860, October 2008. 91

SEBASTIAN MÖSER, ROLAND WAHL, AND REINHARD KLEIN. Out-of-core topologically constrained simplification for city modeling from digital surface models. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-5/W1, February 2009. ISSN 1682-1777. 7, 53

RENATO PAJAROLA. Overview of quadtree-based terrain triangulation and visualization. Technical Report UCI-ICS-02-01, Information & Computer Science, University of California Irvine, 2002. 13, 74

RENATO PAJAROLA, MIGUEL SAINZ, AND ROBERTO LARIO. EXTreME splatting: External memory multiresolution point visualization. Technical Report 04-14, Department of Computer Science, University of California, Irvine, 2004. 30

RENATO B. PAJAROLA. Large scale terrain visualization using the restricted quadtree triangulation. In DAVID EBERT, HANS HAGEN, AND HOLLY RUSHMEIER, editors, *IEEE Visualization '98*, pages 19–26, 1998. 13

RENATO B. PAJAROLA, MARC ANTONIJUAN, AND ROBERTO LARIO. QuadTIN: Quadtree based triangulated irregular networks. In *IEEE Visualization 2002*, pages 395–402, 2002. 14

M. PAULY, M. GROSS, AND L. KOBBELT. Efficient simplification of point-sampled surfaces. In *IEEE Visualization*, 2002. 28

M. PAULY, R. KEISER, L. KOBBELT, AND M. GROSS. Shape modeling with point-sampled geometry. In *SIGGRAPH 2003, Computer Graphics Proceedings*, 2003. 28

H. PFISTER, M. ZWICKER, J. VAN BAAR, AND M. GROSS. Surfels: Surface elements as rendering primitives. In *Siggraph 2000, Computer Graphics Proceedings*, pages 335–342, 2000. 28

ALEX A. POMERANZ. Roam using surface triangle clusters (rustic). Master's thesis, Center for Image Processing and Integrated Computing, University of California, Davis, 2000. 14

JOVAN POPOVIĆ AND HUGUES HOPPE. Progressive simplicial complexes. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer*

graphics and interactive techniques, pages 217–224, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7. doi: http://doi.acm.org/10.1145/258734.258852. 44

ENRICO PUPPO. Variable resolution terrain surfaces. In *Proceedings of the 8th Canadian Conference on Computational Geometry*, pages 202–210, 1996. 13

BORIS RABINOVICH AND CRAIG GOTSMAN. Visualization of large terrains in resource-limited computing environments. In *Proceedings of the conference on Visualization '97*, pages 95–102. ACM Press, 1997. ISBN 1-58113-011-2. 14

JAREK ROSSIGNAC. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, 1999. 19

JAREK ROSSIGNAC AND PAUL BORREL. Multi-resolution 3D approximations for rendering complex scenes. In B. FALCIDIENO AND T. KUNII, editors, *Modeling in Computer Graphics: Methods and Applications*, pages 455–465, Berlin, 1993. Springer-Verlag. Proc. of Conf., Genoa, Italy, June 1993. (Also available as IBM Research Report RC 17697, Feb. 1992, Yorktown Heights, NY 10598). 44

F. ROTTENSTEINER AND C. BRIESE. A new method for building extraction in urban areas from high-resolution lidar data. *IAPRS International Archives of Photogrammetry and Remote Sensing and Spatial Information Sciences*, 34 (3A):295–301, 2002. 55

ROLAND RUITERS. View-dependent far-field level of detail rendering for urban models. *Computer Graphics & Geometry*, 10(3), 2008. 88

SZYMON RUSINKIEWICZ AND MARC LEVOY. QSplat: A multiresolution point rendering system for large meshes. In *Siggraph 2000, Computer Graphics Proceedings*, pages 343–352, 2000. 28

MIGUEL SAINZ AND RENATO PAJAROLA. Point-based rendering techniques. *Computers & Graphics*, 28(6):869–879, 2004. 77

G. SCHAUFLER. Per-object image warping with layered impostors. In *Proceedings of the 9th EG Workshop on Rendering*, pages 145–156, 1998. 29

G. SCHAUFLER AND W. STÜRZLINGER. A three-dimensional image cache for virtual reality. *Computer Graphics Forum*, 15(3):227–236, 1996. 29

RUWEN SCHNABEL, ROLAND WAHL, AND REINHARD KLEIN. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007a. 45, 47, 89

RUWEN SCHNABEL, ROLAND WAHL, AND REINHARD KLEIN. RANSAC based out-of-core point-cloud shape detection for city-modeling. *Schriftenreihe des DVW, Terrestrisches Laser-Scanning (TLS 2007)*, December 2007b. 56, 89

RUWEN SCHNABEL, SEBASTIAN MÖSER, AND REINHARD KLEIN. Fast vector quantization for efficient rendering of compressed point-clouds. *Computers and Graphics*, 32(2):246–259, April 2008a. 89

RUWEN SCHNABEL, RAOUL WESSEL, ROLAND WAHL, AND REINHARD KLEIN. Shape recognition in 3d point-clouds. In V. SKALA, editor, *The 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2008*. UNION Agency-Science Press, February 2008b. ISBN 978-80-86943-15-2. 48, 89

MARTIN SCHNEIDER AND REINHARD KLEIN. Enhancing textured digital elevation models using photographs. In *The Fourth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'08)*, pages 261–268, June 2008. 88

WILLIAM J. SCHROEDER, JONATHAN A. ZARGE, AND WILLIAM E. LORENSEN. Decimation of triangle meshes. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, pages 65–70, 1992. 43

J. SHADE, D. LESCHINSKI, D. SALESIN, T. DEROSE, AND J. SNYDER. Hierarchical image caching for accelerated walkthroughs of complex environments. *Computer Graphics (Proceedings of Siggraph 96)*, 30:75–82, 1996. 29

J. SHADE, S. GORTLER, L.-W. HE, AND R. SZELISKI. Layered depth images. In *Siggraph 1998, Computer Graphics Proceeding*, pages 231–242, 1998. 29

A. SHEFFER, T. BLACKER, AND M. BERCOVIER. Clustering: Automated detail suppression using virtual topology. In *Trends in Unstructured Mesh Generation*, pages 57–64, 1997. 29

F. SILLION, G. DRETTAKIS, AND B. BODELET. Efficient impostor manipulation for real-time visualization of urban scenery. *Computer Graphics Forum*, 16 (3):207–218, 1997. 29

F. TARSHA-KURDI, T. LANDES, P. GRUSSENMEYER, AND M. KOEHL. Model-driven and data-driven approaches using lidar data: Analysis and comparison.

In U. STILLA, H. MAYER, F. ROTTENSTEINER, C. HEIPKE, AND S. HINZ, editors, *PIA07: Photogrammetric Image Analysis*, pages 87–92. International Society for Photogrammetry and Remote Sensing, September 2007. ISBN 978-80-223-2292-8. 56

H. TAUBENBÖCK, N. GOSEBERG, N. SETIADI, G. LÄMMEL, F. MODER, M. OCZIPKA, H. KLÜPFEL, ROLAND WAHL, T. SCHLURMANN, G. STRUNZ, J. BIRKMANN, K. NAGEL, F. SIEGERT, F. LEHMANN, S. DECH, ALEXANDER GRESS, AND REINHARD KLEIN. "Last-mile" preparation for a potential disaster – interdisciplinary approach towards tsunami early warning and an evacuation information system for the coastal city of Padang, Indonesia. *Natural Hazards and Earth System Science*, 9(4):1509–1528, August 2009. ISSN 1561-8633. URL http://www.nat-hazards-earth-syst-sci.net/9/1509/2009/. 89

GABRIEL TAUBIN. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 351–358, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: 10.1145/218380.218473. URL http://doi.acm.org/10.1145/218380.218473. 45

GOKUL VARADHAN AND DINESH MANOCHA. Out-of-core rendering of massive geometric environments. In *IEEE Visualization 2002*, 2002. 30

YANNICK VERDIE, FLORENT LAFARGE, AND PIERRE ALLIEZ. Lod generation for urban scenes. *ACM Trans. Graph.*, 34(3):30:1–30:14, May 2015. ISSN 0730-0301. doi: 10.1145/2732527. URL http://doi.acm.org/10.1145/2732527. 90

JENS VORSATZ, CHRISTIAN RÖSSL, LEIF KOBBELT, AND HANS-PETER SEIDEL. Feature sensitive remeshing. *Computer Graphics Forum*, 20(3):393–401, 2001. 45

ROLAND WAHL AND REINHARD KLEIN. Towards semantic interaction in high-detail realtime terrain and city visualization. In U. STILLA, H. MAYER, F. ROTTENSTEINER, C. HEIPKE, AND S. HINZ, editors, *PIA07: Photogrammetric Image Analysis*, number XXXVI (3/W49A) in International Archives of Photogrammetry and Remote Sensing, pages 179–184, September 2007. ISBN 978-80-223-2292-8. 7, 71

ROLAND WAHL, MANUEL MASSING, PATRICK DEGENER, MICHAEL GUTHE, AND REINHARD KLEIN. Scalable compression and rendering of textured

terrain data. *Journal of WSCG*, 12(3):521–528, February 2004. ISSN 1213-6972. 5, 6, 11, 75, 76

ROLAND WAHL, MICHAEL GUTHE, AND REINHARD KLEIN. Identifying planes in point-clouds for efficient hybrid rendering. In *The 13th Pacific Conference on Computer Graphics and Applications*, October 2005. 6, 25, 77

ROLAND WAHL, RUWEN SCHNABEL, AND REINHARD KLEIN. From detailed digital surface models to city models using constrained simplification. *Photogrammetrie, Fernerkundung, Geoinformation (PFG)*, 3:207–215, July 2008. 6, 41, 53, 55, 56, 58

U. WEIDNER AND WOLFGANG FÖRSTNER. Towards automatic building extraction from high resolution digital elevation models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 50:38–49, 1995. 55

LANCE WILLIAM. Casting curved shadows on curved surfaces. *Computer Graphics (Proceedings of SIGGRAPH '78)*, pages 270–274, 1978. 82

JIANHUA WU AND LEIF KOBBELT. A stream algorithm for the decimation of massive meshes. In *Graphics Interface*, pages 185–192, 2003. 44

JIANHUA WU AND LEIF KOBBELT. Optimized sub-sampling of point sets for surface splatting. *Computer Graphics Forum*, 23(3):643–652, 2004. 27

JULIE C. XIA AND AMITABH VARSHNEY. Dynamic view-dependent simplification for polygonal models. In *Visualization '96. Proceedings.*, pages 327–334, 1996. doi: 10.1109/VISUAL.1996.568126. 13

M. ZWICKER, H. PFISTER, J. VAN BAAR, AND M. GROSS. Surface splatting. In *Siggraph 2001, Computer Graphics Proceedings*, pages 371–378, 2001. 28

M. ZWICKER, J. RÄSÄNEN, M. BOTSCH, C. DACHSBACHER, AND M. PAULY. Perspective accurate splatting. In *Proceedings of Graphics Interface 04*, 2004. 28