HASSO-PLATTNER-INSTITUT
Fachgebiet Computergrafische Systeme

Multi-Scale Representations of Virtual 3D City Models

Dissertation
zur Erlangung des akademischen Grades
"doctor rerum naturalium"
(Dr. rer. nat.)
in der Wissenschaftsdisziplin Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam
von

Tassilo Glander

1. Gutachter: Prof. Dr. Jürgen Döllner
2. Gutachter: Prof. Dr. Monika Sester
3. Gutachter: Prof. Dr. Georg Gartner

eingereicht am 1.7.2012
verteidigt am 8.1.2013

Berlin, 31.1.2013

Published online at the
Institutional Repository of the University of Potsdam:
URL http://opus.kobv.de/ubp/volltexte/2013/6411/
URN urn:nbn:de:kobv:517-opus-64117
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus-64117

Abstract

Virtual 3D city and landscape models are the main subject investigated in this thesis. They digitally represent urban space and have many applications in different domains, e.g., simulation, cadastral management, and city planning. Visualization is an elementary component of these applications. Photo-realistic visualization with an increasingly high degree of detail leads to fundamental problems for comprehensible visualization. A large number of highly detailed and textured objects within a virtual 3D city model may create visual noise and overload the users with information. Objects are subject to perspective foreshortening and may be occluded or not displayed in a meaningful way, as they are too small.

In this thesis we present abstraction techniques that automatically process virtual 3D city and landscape models to derive abstracted representations. These have a reduced degree of detail, while essential characteristics are preserved. After introducing definitions for model, scale, and multi-scale representations, we discuss the fundamentals of map generalization as well as techniques for 3D generalization.

The first presented technique is a cell-based generalization of virtual 3D city models. It creates abstract representations that have a highly reduced level of detail while maintaining essential structures, e.g., the infrastructure network, landmark buildings, and free spaces. The technique automatically partitions the input virtual 3D city model into cells based on the infrastructure network. The single building models contained in each cell are aggregated to abstracted cell blocks. Using weighted infrastructure elements, cell blocks can be computed on different hierarchical levels, storing the hierarchy relation between the cell blocks. Furthermore, we identify initial landmark buildings within a cell by comparing the properties of individual buildings with the aggregated properties of the cell. The initial landmark building objects are then used to compute a landmark hierarchy, which is aligned with the cell block hierarchy. For each block, the identified landmark building models are subtracted using Boolean operations and integrated in a photo-realistic way. Finally, for the interactive 3D visualization we discuss the creation of the virtual 3D geometry and their appearance styling through colors, labeling, and transparency. We demonstrate the technique with example data sets. Additionally, we discuss applications of generalization lenses and transitions between abstract representations.

The second technique is a real-time-rendering technique for geometric enhancement of landmark objects within a virtual 3D city model. Depending on the virtual camera distance, landmark objects are scaled to ensure their visibility within a specific distance interval while deforming their environment. First, in a preprocessing step a landmark hierarchy is computed, this is then used to derive distance intervals for the interactive rendering. At runtime, using the virtual camera distance, a scaling factor is computed and applied to each landmark. The scaling factor is interpolated smoothly at the interval boundaries using cubic Bézier splines. Non-landmark geometry that is near landmark objects is deformed with respect to a limited number of landmarks. We demonstrate the technique by applying it to a highly detailed virtual 3D city model and a generalized 3D city model. In addition we discuss an adaptation of the technique for non-linear projections and mobile devices.

The third technique is a real-time rendering technique to create abstract 3D isocontour visualization of virtual 3D terrain models. The virtual 3D terrain model is visualized as a layered or stepped relief. The technique works without preprocessing and, as it is implemented using

programmable graphics hardware, can be integrated with minimal changes into common terrain rendering techniques. Consequently, the computation is done in the rendering pipeline for each vertex, primitive, i.e., triangle, and fragment. For each vertex, the height is quantized to the nearest isovalue. For each triangle, the vertex configuration with respect to their isovalues is determined first. Using the configuration, the triangle is then subdivided. The subdivision forms a partial step geometry aligned with the triangle. For each fragment, the surface appearance is determined, e.g., depending on the surface texture, shading, and height-color-mapping. Flexible usage of the technique is demonstrated with applications from focus+context visualization, out-of-core terrain rendering, and information visualization.

This thesis presents components for the creation of abstract representations of virtual 3D city and landscape models. Re-using visual language from cartography, the techniques enable users to build on their experience with maps when interpreting these representations. Simultaneously, characteristics of 3D geovirtual environments are taken into account by addressing and discussing, e.g., continuous scale, interaction, and perspective.

Kurzfassung

Gegenstand der Arbeit sind virtuelle 3D-Stadt- und Landschaftsmodelle, die den städtischen Raum in digitalen Repräsentationen abbilden. Sie werden in vielfältigen Anwendungen und zu unterschiedlichen Zwecken eingesetzt, u.a. in den Bereichen Simulation, Stadtplanung und ortsbasierte Dienste. Dabei ist die Visualisierung ein elementarer Bestandteil dieser Anwendungen. Durch realitätsnahe Darstellung und hohen Detailgrad entstehen jedoch zunehmend fundamentale Probleme für eine verständliche Visualisierung. So führt die hohe Anzahl von detailliert ausmodellierten und texturierten Objekten eines virtuellen 3D-Stadtmodells zu Informationsüberflutung beim Betrachter. Dazu können weit entfernte Objekte durch perspektivische Darstellung nicht mehr in ausreichender Größe dargestellt werden, und es entstehen Probleme durch gegenseitige Verdeckung.

In dieser Arbeit werden Abstraktionsverfahren vorgestellt, die diese Probleme behandeln. Ziel der Verfahren ist die automatische Transformation virtueller 3D-Stadt- und Landschaftsmodelle in abstrakte Repräsentationen, die bei reduziertem Detailgrad wichtige Charakteristika erhalten. Nach der Einführung von Grundbegriffen zu Modell, Maßstab und Mehrfachrepräsentationen werden theoretische Grundlagen zur Generalisierung von Karten sowie Verfahren zur 3D-Generalisierung betrachtet.

Im Hauptteil der Arbeit werden die vorgeschlagenen Abstraktionsverfahren beschrieben:

Das erste vorgestellte Verfahren beschreibt die zellbasierte Generalisierung von virtuellen 3D-Stadtmodellen. Es erzeugt abstrakte Repräsentationen, die drastisch im Detailgrad reduziert sind, erhält dabei jedoch die wichtigsten Strukturen, z.B. das Infrastrukturnetz, Landmarkengebäude und Freiflächen. Dazu wird in einem vollautomatischen Verfahren das Eingabestadtmodell mithilfe des Infrastrukturnetzes in Zellen zerlegt. Für jede Zelle wird abstrakte Gebäudegeometrie erzeugt, indem die enthaltenen Einzelgebäude mit ihren Eigenschaften aggregiert werden. Durch Berücksichtigung gewichteter Elemente des Infrastrukturnetzes können Zellblöcke auf verschiedenen Hierarchieebenen berechnet werden. Dabei wird die entstehende Hierarchiebeziehung explizit den Zellblöcken zugeordnet und gespeichert. Weiterhin werden Landmarken gesondert berücksichtigt: Zunächst werden Eigenschaften der Einzelgebäudes statistisch mit den aggregierten Eigenschaften der Zelle verglichen. Einzelgebäude mit hoher statistischer Abweichung werden initial als Landmarken identifiziert Mithilfe der initialen Landmarken wird eine Landmarkenhierarchie berechnet und der Zellblockhierarchie gegenübergestellt. Schließlich werden die Landmarkengebäude je nach Hierarchieebene aus den generalisierten Blöcken mit Booleschen Operationen ausgeschnitten und realitätsnah dargestellt. Die Ergebnisse des Verfahrens lassen sich in interaktiver 3D-Darstellung einsetzen, dazu werden in der Arbeit die Erzeugung der virtuellen 3D-Geometrie sowie ihre Gestaltung durch Grundrissextrusion, Färbung, Beschriftung und Transparenz betrachtet. Das Verfahren wird beispielhaft an verschiedenen Datensätzen demonstriert. Zusätzlich werden Erweiterungen unter anderem durch Generalisierungslinsen sowie Übergänge zwischen Repräsentationen diskutiert.

Das zweite vorgestellte Verfahren ist ein Echtzeit-Rendering-Verfahren für geometrische Hervorhebung von Landmarken innerhalb eines virtuellen 3D-Stadtmodells: Landmarkenmodelle werden abhängig von der virtuellen Kameradistanz vergrößert, so dass sie innerhalb eines spezifischen Entfernungsintervalls sichtbar bleiben; dabei wird ihre Umgebung deformiert. In einem Vorverarbeitungsschritt wird eine Landmarkenhierarchie bestimmt, aus der die Entfernungsintervalle für die in-

teraktive Darstellung abgeleitet werden. Zur Laufzeit wird anhand der virtuellen Kameraentfernung je Landmarke ein dynamischer Skalierungsfaktor bestimmt, der das Landmarkenmodell auf eine sichtbare Größe skaliert. Dabei wird der Skalierungsfaktor an den Intervallgrenzen durch kubische Bézier-Splines interpoliert. Für Nicht-Landmarkengeometrie in der Umgebung wird die Deformation bezüglich einer begrenzten Menge von Landmarken berechnet. Die Eignung des Verfahrens wird anhand eines detaillierten virtuellen 3D-Stadtmodells sowie eines generalisierten 3D-Stadtmodells demonstriert. Weiterhin wird eine Adaption des Verfahrens für eine 3D-Kartenanwendung für mobile Geräte in Kombination mit einem Ansatz für nicht-lineare Projektionen demonstriert.

Das dritte vorgestellte Verfahren ist ein Echtzeit-Rendering-Verfahren, das eine abstrakte 3D-Isokonturen-Darstellung von virtuellen 3D-Geländemodellen erzeugt. Für das Geländemodell wird eine Stufenreliefdarstellung erzeugt, die für eine Menge von nutzergewählten Höhenintervallen das Gelände repräsentiert. Das Verfahren arbeitet ohne Vorverarbeitung und lässt sich mit minimalen Anpassungen mit üblichen Gelände-Rendering-Verfahren kombinieren, da es mittels programmierbarer Grafikkarten-Hardware implementiert ist. Entsprechend erfolgt die Verarbeitung in der Prozesskette pro Geometrieknoten, pro Geometrieprimitiv, d.h. Dreieck, und pro Bildfragment. Pro Geometrieknoten wird zunächst die Höhe auf den nächstliegenden Isowert quantisiert. Pro Dreieck wird dann die Konfiguration bezüglich der Isowerte der drei Geometrieknoten bestimmt. Anhand der Konfiguration wird eine geometrische Unterteilung vorgenommen, so dass ein Stufenausschnitt entsteht, der dem aktuellen Dreieck entspricht. Pro Bildfragment wird schließlich die finale Erscheinung definiert, z.B. anhand von Oberflächentextur, durch Schattierung und Höheneinfärbung. Die vielfältigen Einsatzmöglichkeiten werden mit Anwendungen aus den Bereichen Fokus+Kontext-Darstellungen, Out-Of-Core-Geländedarstellungen und der Informationsvisualisierung demonstriert.

Die Arbeit stellt Bausteine für die Erzeugung abstrakter Darstellungen von virtuellen 3D-Stadt- und Landschaftsmodellen vor. Durch die Orientierung an kartographischer Bildsprache können die Nutzer auf bestehende Erfahrungen bei der Interpretation zurückgreifen. Gleichzeitig werden die charakteristischen Eigenschaften 3D geovirtueller Umgebungen berücksichtigt, indem z.B. kontinuierlicher Maßstab, Interaktion und Perspektive behandelt und diskutiert werden.

Acknowledgements

I would like to thank Prof. Dr. Jürgen Döllner for his encouragement, advice, and enlightening discussions throughout my work at the department of computer graphics systems (CGS) at the Hasso-Plattner-Institut (Universität Potsdam). Only through his support I had the opportunity to start working at Vicomtech (San Sebastian, Spain) while finishing this thesis. I also would like to thank Prof. Dr. Georg Gartner, Prof. Dr. Monika Sester, and Prof. Dr. Hartmut Asche for agreeing to review this thesis. I am thankful to the Bundesministerium für Bildung und Forschung (BMBF), which funded my research at the HPI as part of the research project "3D Geoinformation".

Through the years of my studies and research, I enjoyed working at the CGS department very much. Thanks are due to my colleagues for inspiring discussions, research collaborations, and lots of fun. To name a few, I want to thank Matthias Trapp, Benjamin Hagedorn, and Henrik Buchholz for fruitful collaborations. In addition, I am thankful to Rico Richter for being my eyes and ears at HPI after I left, and for commenting my thesis.

During the years I had the opportunity to work with many talented students. I want to thank them for providing feedback to my work. I am also grateful to Sabine Biewendt for always offering help, incredibly caring organization, and providing coffee supplies. I owe to Lyn Wilson for proofreading this thesis.

Finally, I am grateful for the support of my family and friends during my studies and beyond, through caring questions, for cheering me up, and for giving me chocolate. My research and this document would have been impossible without my wife Marie-Luise. I am deeply grateful for her continuing support, enduring patience, and love. Thanks for sharing this adventure! Finally, I thank my precious daughter Mathilde for reminding me of a life beyond the thesis.

Copyright Notices

This thesis contains material that I have published previously in journal articles and conference proceedings. Publishers grant authors personal use of this material as part of their thesis, if the author includes a copyright notice and points to the document where the material appeared originally. In this thesis, images that have been published before are marked with a footnote pointing to the original document. The required copyright notices are listed as follows.

ACM Copyright Notice

IEEE Copyright Notice

Springer Copyright Notice

Elsevier Copyright Notice

Eurographics Copyright Notice

Contents

CHAPTER 1

Introduction

Virtual 3D city models as digital representations of urban space represent a major topic in the discipline of geovisualization. This discipline is aiming to "provide theory, methods, and tools for visual exploration, analysis, synthesis, and presentation of geospatial data (any data having geospatial referencing)" (MacEachren and Kraak, 2001). One of the challenges of geovisualization is how to effectively present and communicate information by adapting information density to an appropriate level of abstraction taking into account the context of user, task, and medium. Drawing inspiration from cartographic methods, this thesis explores how abstract representations can be derived from virtual 3D city models. In the introduction we define basic terms, present the research objectives and the contributions of this work to the field of geovisualization.

1.1 Background and Motivation

A virtual 3D city model (3DCM) is the digital representation of urban space, e.g., cities or metropolitan regions, that describes geometrical, topological, semantical, and appearance properties of its components. The components include terrain surface, buildings, vegetation objects, transportation objects, land use classification, and city furniture, as defined by the international CityGML standard published by the Open Geospatial Consortium (OGC) (Gröger et al., 2008). CityGML defines an open data model and an Extensible Markup Language (XML) based format for storing and exchanging 3DCMs. In general, a 3DCM serves as an integration platform for multiple facets of an urban information space, as pointed out by Batty (2006):

> "In short, the new models are not simply the digital geometry of traditional models but large scale data bases which can be viewed in 3D. As such, they already represent a way of merging more abstract symbolic or thematic data, even symbolic models, into this mode of representation."

Their versatility and increasing importance are demonstrated by application examples from various fields where 3DCMs are used for different purposes:

Simulation based on a mathematical model of a 3DCM enables one to make predictions about phenomena such as noise emission and propagation (Czerwinski et al., 2007) or solar radiation for environmental analysis (Carneiro et al., 2009).

Mapping / Visualization uses the 3DCM to associate it with thematic data to give spatial context (Wolff and Asche, 2009; Döllner et al., 2006a). Furthermore, components of the 3DCM can be used as visual variables, for example, by adapting color or height of buildings according to the data to be mapped (Buchholz and Döllner, 2005b).

Management of a city with respect to fiscal and legal purposes, land use, or infrastructure can be done in a 3D cadastre (Stoter, 2004). Another topic is the management of single buildings

during their lifecycle including design, construction, and maintenance, as addressed by building information modeling (BIM) (Hagedorn and Döllner, 2007).

Planning in 3DCMs enables urban planners, architects, and urban landscape designers to visualize and analyze planned changes in context. Moreover, interactive tools allow them to do ad-hoc planning and collaboration (Dangermond, 2010; Klimke and Döllner, 2010; Chen et al., 2008)

Presentation of 3DCMs facilitates communication in different contexts such as cultural heritage sites (Trapp et al., 2010), business and tourism marketing (Döllner et al., 2006c). Also, the entertainment industry relies on highly detailed 3DCMs for computer games, movies and commercials (Kelly and McCabe, 2006).

Visualization is an important part of many applications of 3DCMs. *To visualize* means to bring something to the mind, hence it includes the human reception in the process. More specifically, *geovisualization* is the discipline of visualizing geospatial data with the aim to communicate information, explore data, and build and test hypotheses (MacEachren and Kraak, 2001). It draws from different fields, e.g., cartography, computer graphics, geographic information science, and cognition science, to effectively present geospatial data. In addition to technical issues of rendering, aesthetic and design aspects have to be taken into consideration (Häberling, 2003).

1.2 Challenges for Visualizing Virtual 3D City Models

In the last decade, research related to 3DCMs focused on high resolution terrain and building models and photo-realistic rendering. Advances were made along the creation and processing pipeline of 3DCMs, in acquisition of geometry and visual models, e.g., (Brenner, 2000; Karner et al., 2001; Remondino and El-Hakim, 2006), building and façade reconstruction, e.g., (Kada, 2009; Müller et al., 2007), and visualization. Advances in visualization addressed, for example, the handling of large amounts of data (Losasso and Hoppe, 2004; Buchholz and Döllner, 2005a) and photo-realism (Döllner et al., 2006b). Today, 3DCMs can be created with a high degree of detail and quality. For example, the 3DCM of Berlin has app. 470.000 buildings on an area of approximately $775\ \mathrm{km}^2$ with approximately $150\ \mathrm{GB}$ of façade texture data (Lorenz, 2010).

With increasingly detailed models and their extended usage, fundamental problems for the visualization arise:

- 3DCMs contain many single objects such as building models, city furniture, or vegetation objects. The high quantity of heterogeneous objects is difficult to perceive and process, potentially leading to information flooding of users. Information density needs to be adapted to an appropriate level depending on the application.

- Creating a mapping between virtual 3D environment and reality can be difficult due to missing visual landmarks (Vinson, 1999). The integration of visual landmarks into the visualization of 3DCMs is crucial to facilitate user orientation within the model and within reality using the model.

- Classical rendering follows the pinhole camera model (Foley et al., 1995). Due to perspective foreshortening objects are rendered smaller with increasing distance to the virtual camera. In consequence, for example, distant buildings are hard to differentiate. At some point they may even drop to sub pixel size, leading to disturbing flickering during animation. Hence, small objects have to be either enhanced or eliminated.

- Occlusion of background objects by foreground objects can hide essential information, such as a route to follow in a navigation scenario. Essential information must be enhanced to ensure their visibility.
- Computational complexity for rendering increases, because massive geometry and texture memory is required for highly detailed 3DCMs. Efficient rendering techniques are needed for interactive visualization on common consumer hardware.

To address these fundamental problems, visualization of 3DCMs requires *abstraction techniques* which adapt information density to the user, task, and medium. Moreover, effective use of 3D visualization beyond photo-realistic imagery requires abstraction, as "realism can be a distraction" (MacEachren et al., 1999). Abstraction techniques create representations for different scales with different levels of abstraction. They should aim to reproduce characteristics of classical maps, which make use of well-known design principles (Häberling, 2003). Map generalization provides the methodology to derive abstract, legible maps in cartography (MacEachren, 1995; Hake et al., 2002). However, it is less commonly applied to 3DCMs. Abstraction techniques must also take into account the specific characteristics of 3D visualization; for example, perspective views depict a continuous range of scales within a single image. In addition interactive, dynamic visualization allows one adaptation in real-time.

1.3 Research Objectives

We derive research questions that address the need for abstraction techniques. This need represents a key challenge for effective visualization of 3DCMs. The first research question of this thesis is: *Given a detailed 3DCM, how can we derive abstracted variants in multiple scales?* This question involves a number of requirements:

Reduced information density: Abstract representations need to have reduced information density while maintaining sufficient similarity to the given 3DCM by resembling its major structure and properties.

Hierarchical relations: Multi-scale representations need to support hierarchical relations. This means hierarchies known from the city, e.g., city, district, neighborhood, street, building, are to be taken into account in the multi-scale representations to enable recognition at all levels of abstraction. Furthermore, the generalization relation must be modeled explicitly to support (further) computational processing.

Automatic processing: Abstraction techniques should have a high degree of automation to handle massive data sets and yield consistent results.

Parameterized processing: Abstraction techniques should provide sufficient parameters to give designers sufficient control over the outcome.

The second research question occurs in the context of interactive 3D visualization: *How can we visualize multi-scale representations of a 3DCM?* This question involves the following requirements:

Dynamic adaptation: In an interactive application users have many degrees of freedom to change the visualization, for example, by switching between displayed data layers, adapting virtual camera parameters, and applying focus+context techniques. Hence, if the user interacts with and changes the visualization, also displayed representation has to be adapted to a level of abstraction that is appropriate for the new situation. Simple switching is perceived as

distracting. To avoid this, a visualization technique has to perform smooth transitions between representations.

Handling occlusion: Important elements of a 3DCM should have enhanced visibility. Occlusion needs to be handled to ensure the visibility of important elements.

Expressing abstractness: 3D rendering is commonly perceived as being close to reality because it matches real world experience. Hence, an abstract 3D visualization needs to carefully communicate and point out its abstractness to the users to avoid wrong hypotheses.

1.4 Contributions and Structure of this Thesis

This thesis explores approaches to derive multiple abstract representations of 3DCMs with the aim to adapt information density, enhance important structures, and reduce "visual noise". In the following chapter, we review fundamental concepts and principles (Chapter 2). In the central part of this thesis, we present concepts and techniques addressing the research questions in separate chapters together with a review of their related work, implementation, and discussion.

Cell-based generalization of 3DCMs is a concept and implementation of an automatic preprocessing technique to create multi-scale representations of 3DCMs with different levels of abstraction (Chapter 3).

Geometric landmark enhancement in 3DCMs is a concept and implementation of a real-time technique to adapt the size of landmark objects dynamically to ensure their visibility (Chapter 4).

3D isocontour terrain visualization is a concept and implementation of a real-time technique to obtain a 3D stepped terrain visualization without preprocessing (Chapter 5).

A summary and an outlook for open problems finish the thesis (Chapter 6).

1.4.1 Cell-Based Generalization of Virtual 3D City Models

One problem on the way to improve visualizations of 3DCM beyond photo-realistic presentation is the visual complexity in terms of number of objects, geometric shape complexity, and lack of distinguishable orientation marks. We present a method to create abstract representations for 3DCMs by deriving abstracted building blocks from a given infrastructure network. Properties of the method and resulting representations are as follows:

- The derived representations yield a high degree of abstraction. Currently, approaches focus on simplification of single buildings, aiming at the reduction of computational complexity and memory consumption while maintaining visual appearance.
- The technique derives hierarchically structured representations on multiple scales. Items on different levels of abstraction are connected by a generalization relation.
- Landmark objects are integrated in full detail to facilitate orientation. A hierarchical model is employed to provide an appropriate number of landmarks on different levels of abstraction.
- The technique runs automatically once the parameters are set up. Nevertheless, the outcome can be customized to highlight local areas of interest.

The technique has been implemented prototypically both within Autodesk's LandXplorer platform[1], and based on Open Scene Graph (OSG)[2]. It was presented, for example, in (Glander and Döllner, 2009, 2007) and is the basis for follow-up research as presented in the application section of Chapter 3.

1.4.2 Geometric Landmark Enhancement in 3DCMs

Landmarks are crucial elements of 3DCMs to facilitate orientation. However, classical 3D rendering implies perspective effects such as foreshortening and occlusion, which may hide landmark objects. We present a method to scale landmark objects depending on virtual camera distance to highlight them to the user. As the scaling is done in real-time the presentation is adapted to the interactions of the user. Properties of the technique are as follows:

- The concept of geometric highlighting effectively enhances the skyline to show landmark objects beyond their original visibility.
- The scaling factor is computed in real-time based on pre-defined distance intervals for each landmark object. The scaling effect is controlled by different weight levels for the landmark objects which determine the distance intervals.
- Displacement of geometry surrounding the landmarks as a consequence of the scaling is done using two strategies: landmarks apply a force to each other that repel them; non-landmark objects are deformed within a radial distortion zone that is centered at the nearest landmark.

The technique has been implemented both within Autodesk's LandXplorer platform and in a research prototype with Nokia maps[3]. It was presented, for example, in (Glander and Döllner, 2009; Glander et al., 2007).

1.4.3 3D Isocontour Terrain Visualization

As a complementary visualization style to classical terrain rendering, we present 3D isocontour or stepped terrain visualization. The properties are the following:

- The visualization provides a de-noised, stair-like surface for comprehensive presentation of the terrain's structure similar to visualization techniques used in cartography and landscape architecture.
- The technique is entirely based on the graphics processing unit (GPU) and does not require preprocessing. Hence, it can be integrated seamlessly with the generic terrain rendering pipeline.
- It is a real-time technique, allowing flexible use of time-varying terrain models, application of magic lenses, interactive re-parameterization, and transitions between original and stepped visualization style.

The technique was implemented both within the Virtual Rendering System (VRS)[4] (Döllner and Hinrichs, 1995) and the Open Scene Graph (OSG). It was presented, for example, in (Glander et al., 2010b, 2011).

[1] `www.autodesk.com/landxplorer`, now Infrastructure Modeler (accessed 1.7.2012)
[2] `www.openscenegraph.org` (accessed 1.7.2012)
[3] `www.maps.nokia.com` (accessed 1.7.2012)
[4] `www.vrs3d.org` (accessed 1.7.2012)

CHAPTER 2

Fundamentals of Multi-Scale Representations

In the context of this thesis, multi-scale representations are representations of a given model in several degrees of detail. The principle of multi-scale representations can be found in various areas of visualization such as rendering virtual 3D environments, graphical modeling languages, and cartography. Generalization as the "selection and simplified representation of detail appropriate to the scale and/or the purpose of the map" (International Cartographic Association, 1973) provides theory and methodology for the creation of multi-scale representations in cartography. In this chapter we study the basics of multi-scale representations, map generalization, and how generalization can be included in the visualization pipeline.

2.1 Model, Scale and Multi-Scale Representations

2.1.1 Model

A *model* is an incomplete description of an entity or phenomenon that targets a specific purpose. Thus, the notion of a model involves mapping of the *described* to its *description* (the derived model), i.e., the description shares attributes with the described. The described can be phenomena or structures perceived in the real world or described by another model.

The Object Management Group (OMG) (2011a) defines it as follows:

> "A model captures a view of a system. It is an abstraction of the system, with a certain purpose. This purpose determines what is to be included in the model and what is irrelevant. Thus, the model completely describes those aspects of the system that are relevant to the purpose of the model, at the appropriate level of detail."

This definition introduces the model's *purpose* as a criterion to identify the attributes of the described that must be included in the description. In general, this involves

- the task that the model shall support, i.e., what is the goal when using the model;
- the user, i.e., how can comprehension of the model be maximized;
- and limitations of the mapping process itself, i.e, which are the limitations of the modeling / acquisition technique.

These issues are part of the abstraction process that adapt information density and create the appropriate level of detail.

2.1.2 Scale

In the context of spatial models, the notion of abstraction is strongly related to scale. Scale is a fundamental principle referring to the frame of reference of a model or an observation (Sankey, 2008; Mackaness, 2007). The frame of reference describes the context in which the model serves its purpose, or the observation is made. To make hypotheses, judgments or further observations, the appropriate frame of reference must be taken into account.

According to the original meaning, scale is defined geometrically as the ratio between distance in the model and the real world. Cartographic scale goes beyond this definition as, in addition, it is connected with geometric and thematic abstraction to provide a better visualization (Mackaness, 2007).

As an example consider wayfinding using a classical map: instead of just scaling an aerial photo to fit it on a map sheet, the map shows symbolized data, contains less details, and some map features such as roads are displayed larger than a correct geometric scale would require. The map shows reality in a way that aims at maximizing legibility and relies on abstraction to do so.

2.1.3 Characteristics of Multi-Scale Representations

The goal of multi-scale representations is to provide several representations where each representation is adapted to a different information density. Typically one *primary representation* is used to derive *secondary representations* with adapted scale as needed. In practice multiple discrete representations are typically prepared and stored in advance. This has some disadvantages (Cecconi and Galanda, 2002):

- The requested scale may be arbitrarily chosen from a continuous interval whereas only a limited discrete set of representations have been precomputed. Hence, there is a potential mismatch between the requested and the available representation.
- Precomputing a number of representations requires preprocessing time and storage space for the additional representations.
- In case of changes to the primary representation, all derived representations have to be recreated.

Ideally the required variant can be generated on-the-fly for a continuous range of resolution requirements. However this is not always feasible either for conceptional reasons, as abstraction may not be fully automatic, or for technical reasons, as on-the-fly generation may not be computable fast enough. To overcome the problem of mismatch between required and prepared representation, the representation with the closest resolution is used. Also, transition techniques apply blending between the most appropriate discrete representations to workaround the limited number of precomputed variants.

In their simplest form, multi-scale representations form a ordered, linear sequence of representations $R^0, R^1, ..., R^n$, where R^0 has the highest detail and R^n the lowest. The examples in Section 2.1.4 belong to this class. Frequently, multi-scale representations are organized hierarchically, i.e., their relation forms a tree structure rather than a linear structure. Preprocessing a primary representation in a hierarchical way allows one to follow a divide-and-conquer approach, i.e., to split the problem into smaller portions and process them independently. Once the multi-scale representations have been precomputed, selection of the appropriate representation for the current need can be controlled using different criteria. For example, in spatial context it is interesting to base refinement on location.

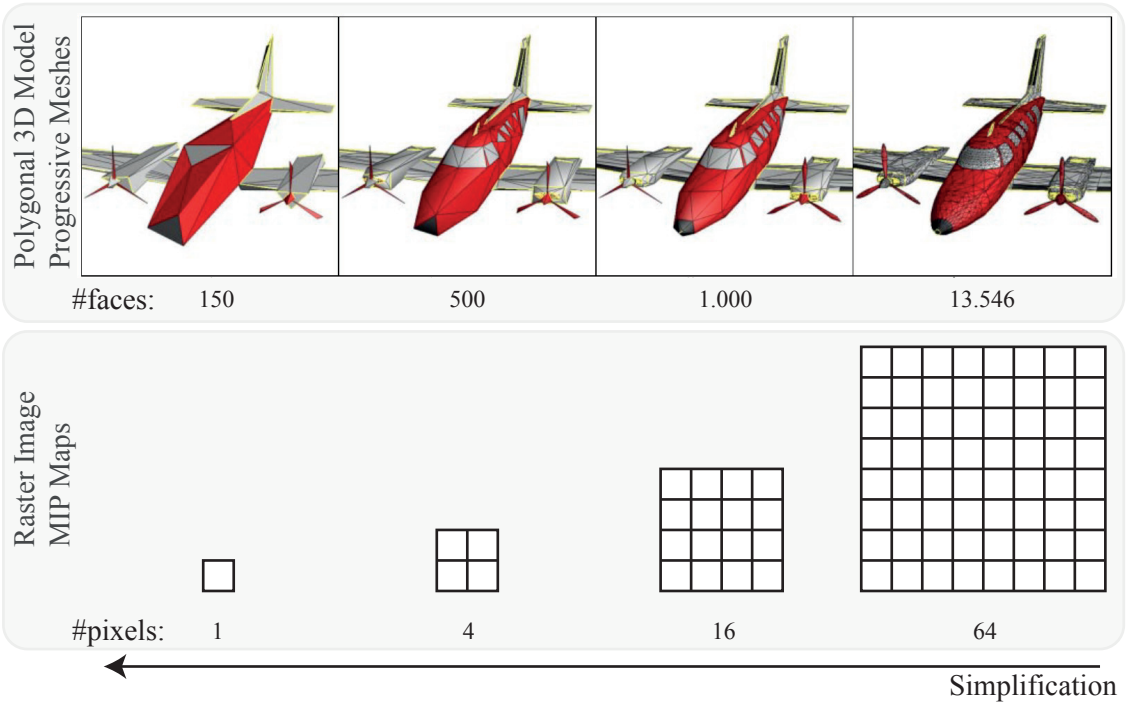


Figure 2.1: Progressive meshes are multi-scale representations of a polygonal 3D model providing reduced level of detail while maintaining the overall shape (upper row, taken from (Hoppe, 1996)). MIP maps are multi-scale representations of a raster image providing reduced resolution images for texturing surfaces.

2.1.4 Level of Detail Modeling in 3D Computer Graphics

In 3D computer graphics, level of detail (LOD) modeling represents a fundamental principle (Figure 2.1). The geometry of a virtual 3D model is frequently stored in multiple LODs (resolutions) (Foley et al., 1995). The goal is to reduce the computational complexity for rendering the virtual 3D model while maintaining its visual appearance. LOD modeling enables interactive rendering of data sets that otherwise could not be rendered interactively or could not be rendered at all, as their size exceeds main or graphics memory, or processing power is too low. If the virtual 3D model is rendered very small, the user ideally cannot see the difference between the original geometric model and a geometric model with reduced details. The LOD models are models with low polygon count either created by hand or derived automatically from a primary model (Hoppe, 1996; Garland and Heckbert, 1997; Pajarola and Gobbetti, 2007; Hu et al., 2009). At runtime the LOD models are typically exchanged dynamically depending on their screen size, virtual camera distance, or available computational capacity. To avoid disturbing popping artifacts when switching, geomorph techniques perform a smooth geometric interpolation between different LOD models (Hoppe, 1996).

As another example, a LOD model for texture application is employed in 3D computer graphics. Textures are used to describe the surface appearance of virtual 3D models. A common multi-scale technique is *multum in parvo* (MIP) texture mapping where several raster representations are pre-filtered and stored in different resolutions to avoid aliasing artifacts at runtime (Foley et al., 1995). Starting from the input image, an image pyramid is built where each subsequent level image has only a quarter of the resolution of the current level image. Thus at rendering time, depending on the projected size of the surface on screen, the most appropriate texture can be selected for mapping onto the surface. To avoid popping because of switching between textures, trilinear filtering blends between the two most appropriate images.

2.1.5 Abstraction in Structure and Process Visualization

Business process modeling (BPM) is concerned with the representation of processes in an enterprise to analyze them and improve efficiency. A commonly used language for modeling and visualization is Business Process Model and Notation (BPMN) (Object Management Group (OMG), 2011b), which is a graph-based language having elements such as activities, events, gateways, and data objects. As processes can become complex, containing many entities and relations, BPM abstraction can be applied which seeks to provide a simplified overview visualization. Towards this goal several tasks can be clustered to an aggregated activity based on a semantical analysis, reducing the number of entities considerably (Smirnov et al., 2011).

2.1.6 Multi-Scale Maps in Cartography

Cartography has a long history with multi-scale representations, as maps are made for different scales and with different sizes. In addition to geometrical scaling, cartographic generalization is applied to optimize legibility. Maps are made in a number of discrete cartographic scales. For topographical maps, national mapping agencies (NMA) have established products with common cartographic scales to provide maps at city, federal, or state level. Interactive cartographic maps such as provided by web mapping offer access to multi-scale maps by allowing the user to zoom and to switch between maps of different scales at the current position.

However, no formalization or standardization for multi-scale maps and related generalization requirements exist in cartography. Mapping agencies implicitly define it through rule sets they apply to generate their map products. Researchers report on the different application of generalization and characteristics of scale in Europe and the United States of America: Foerster and Stoter (2008) survey eleven European national mapping agencies and document the use, importance, and problems of applying generalization operators in different scales. The survey is part of a larger project that evaluates the applicability of commercial software for map making and generalization (Stoter et al., 2010). Brewer and Buttenfield (2007) present practices of map making in the US focusing on the symbolization aspect. As part of the *ScaleMaster* project they report on the changes that are applied to the map in different scales, differentiated by map object categories, such as buildings, roads, and vegetation objects. The community-driven *OpenStreetMap* project also focuses on symbolization. A styling file explicitly defines the validity of different map object categories and their symbolization (OpenStreetMap, 2012).

One can conclude that multi-scale representations in visualization are commonly motivated by two aspects. One aspect is the simplification as a means to improve and make more efficient computational processing, including rendering, simulation, storage, and transmission of representations. The other aspect is the improvement of human processing including perception, comprehension, and memorization of representation. An important characteristic of multi-scale representation is the similarity between the representations and the described subject, where similarity is defined depending on the purpose. In the following sections we discuss how to derive multi-scale representations in context of map and 3DCM visualization.

2.2 Map Generalization

The informal meaning of *to generalize* something means to make it more widely applicable (McKean, 2005). In the context of maps, generalization means the reduction of details and enhancement of the necessary elements as demanded by legibility requirements, e.g., when going

from one scale to a smaller scale (Hake et al., 2002). McMaster and Shea (1992) define digital generalization

> "as the process of deriving, from a data source, a symbolically or digitally-encoded cartographic data set through the application of spatial and attribute transformations. Objectives of this derivation are: to reduce in scope the amount, type, and cartographic portrayal of the mapped or encoded data consistent with the chosen map purpose and intended audience; and to maintain clarity of presentation at the target scale."

According to Hake et al. (2002) a general challenge of generalization lies in the competitive nature of its objectives: clarity of presentation or *legibility* requires that map objects are shown with a minimum size and clear shape. Geometric transformation applied to enforce legibility, e.g., scaling or displacing objects, in itself contradicts the principle of *geometric correctness*. In addition map objects compete for the available space, so eventually less important objects need to be omitted which contradicts the principle of *completeness*.

A term related to generalization is *schematization*. In cognition science (Brunet, 1993) this term is defined as "intentionally simplifying a representation beyond technical needs to achieve cognitive adequacy" (Klippel, 2003). The process of schematization consists of analysis how spatial problems, e.g., a route description or a thematic map, are processed cognitively and memorized internally. The findings are used to create representations, e.g., schematic map drawings or verbal route instructions, that are cognitively enhanced (Reimer and Fohringer, 2010; Klippel et al., 2006).

2.2.1 Towards Automatic Map Generalization

Automation of generalization is desired for several reasons: data sets used for map creation are usually very large, making manual processing a tedious and error prone task. Also repeatability of the process is desired to achieve consistent results. This is important for the application to different geographic areas, and in case of re-generalization when the source data set is changed, e.g., because of re-acquisition. On-demand maps, which are adapted to specific needs and purposes, also motivate automation techniques (Cecconi et al., 2002).

Historically, cartographers employed both artistic and technical skills when drawing a spatial setting on a map sheet, taking into account aesthetic and legibility aspects. Abstraction or generalization is applied in this process while deciding what and how to draw. It therefore largely depends on personal experience and is highly subjective. Even the same cartographer may come to slightly different results when executing the same mapping task.

With increasing computing capabilities in the 70s, digital processing became a part of map making, and automation began to be a subject of research. Conceptual models of map generalization were developed which describe the generalization process and identify generalization operators applied during map making, e.g., (Brassel and Weibel, 1988; McMaster and Shea, 1992). Operators such as simplification of linear features, e.g., a river course, roads, and coast lines, were among the first automated techniques and became part of the cartographers' tool set.

In the 90s constraint-based generalization was proposed (Beard, 1991). This describes what the outcome of map making should look like by hard (must-be) and soft constraints (should-be), allowing the generalization process to be automated as a whole. This approach is implemented in agent-based generalization (Lamy et al., 1999), where the map is created by multiple autonomous agents representing map features. The constraints are modeled as goals aimed for by the agents which have a repertoire of techniques to improve their situation. Another holistic approach is optimization through least squares adjustment where constraints are expressed as equations in an overdetermined system (Sester, 2000b).

Today many NMAs rely on software systems implementing constraint-based generalization for their map production (Stoter et al., 2010). The solutions are based on, e.g., agents, least

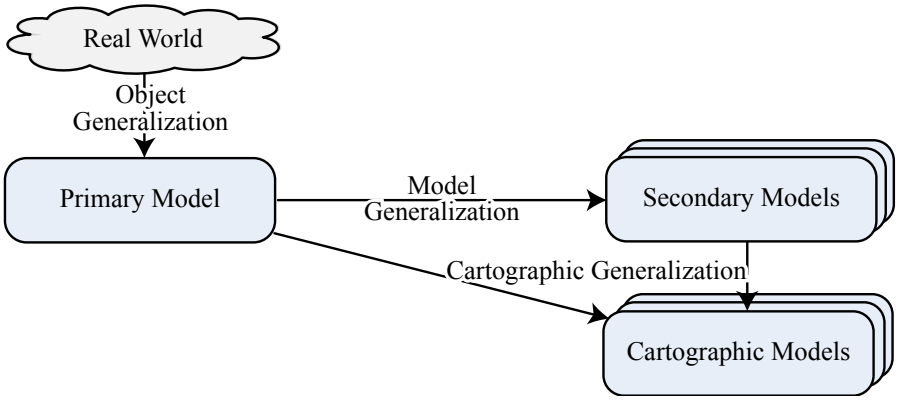


Figure 2.2: Object generalization occurs when real-world objects are expressed in a primary model. Subsequent model generalization yield secondary models with reduced detail. Cartographic generalization creates a tertiary model, which explicitly adapts the symbolized model to increase legibility (Hake et al., 2002).

squares adjustment, or stochastic optimization. Research in generalization addresses – in addition to improving single generalization operators and generalization frameworks – complementary directions, such as the employment of generalization in distributed settings, i.e., web service infrastructure (Foerster and Stoter, 2006; Neun et al., 2008)), symbolization specifications (OGC, 2006; Neubauer and Zipf, 2007), on demand mapping for special purpose maps (Cecconi et al., 2002), and generalization in virtual 3D environments (Section 2.3).

2.2.2 Conceptual Models for Map Generalization

According to Hake et al. (2002), generalization occurs in different stages during map creation (Figure 2.2):

Object generalization occurs during the initial stage where real-world objects are mapped to a primary model. The main reasons for generalization are the intended selection of aspects to be captured and the unintended limitations of the acquisition process, e.g., because of sensor resolution.

Model generalization is the transformation of the primary model to a secondary model aiming at data reduction. It is not directly intended to improve the legibility, but rather to improve memory consumption, data transmission, and computational processing efficiency.

Cartographic generalization is the stage where a model is transformed into a tertiary or cartographic model to improve legibility, after symbolization has been applied. It handles visual conflicts between symbolized features, e.g., by enforcing minimal distances, smoothing, and removing overlap.

The primary model is independent of cartographic scale, whereas secondary and tertiary models depend on scale, purpose and user (MacEachren, 1995). While it is apparent that all three stages contribute to the final map output, this conceptional model provides a separation between data processing to create digital landscape models (DLM) and creation of digital cartographic models (DCM) used as visual representations.

Elementary tasks applied during generalization have been identified as generalization operators by several authors (Brassel and Weibel, 1988; McMaster and Shea, 1992). They represent generic patterns rather than concrete algorithms; each generalization operator may have several implementations. The following list is based on (Hake et al., 2002):

Simplification removes minor details of a shape, e.g., small protrusions of a building footprint or small variations of a curve representing a road.

Enlargement makes shapes bigger or wider to ensure graphical minimum dimensions, e.g., by broadening a road.

Displacement occurs as a consequence of enlargement to counter symbols overlapping each other, e.g., a building located directly next to a road that needs to be broadened.

Aggregation is the combination of several objects into one shape, e.g., by representing several single neighboring buildings by a single shape.

Selection chooses which objects to include in the representation (and which to omit), e.g., by considering only major roads and omitting small building shapes.

Classification replaces objects of different sub-categories by objects of a more general category, e.g., when expressing both coniferous and broad leafed tree areas by a forest area signature.

Exaggeration enhances shapes or characteristics of map objects that are of high importance and should be emphasized, e.g., by scaling an important building.

The first three tasks are specified as geometric operators, while the others are thematic operators with geometric effects (Hake et al., 2002). They refer to both cartographic and model generalization.

The operators are not standardized and other authors identify slightly different sets. For example, McMaster and Shea (1992) differentiate aggregation (joining several point features to one area), amalgamation (joining several polygon features to one area) and merging (joining two parallel linear features to one curve) depending on the dimensionality of the input features. In context of the AGENT project which only addresses cartographic generalization, generalization operators have been classified depending on whether they transform single or multiple features (Cecconi, 2003).

2.2.3 Implementing Generalization Operators

A number of implementations for generalization operators have been developed and presented in the last decades. The goal of generalization operators shifted from supporting the map maker with tools automating tedious work to building blocks of systems that attempt to automate the whole process of map making. In the following we will discuss the general methodologies from the domain of building generalization and give examples that implement them. For a more detailed discussion of generalization operators and their implementation see (Regnauld and McMaster, 2007).

Among the first approaches implementing generalization operators were simple rule-based techniques, which define a number of detectable conditions with respective actions. An example for a rule-based technique is the building footprint simplification suggested by Staufenbiel (1973) as used in, e.g., (Sester, 2000b). As initial condition, the algorithm compares the length of the footprint's edges against a given threshold value to detect short edges. Depending on the local curvature, three cases are distinguished and a rule is applied to remove the short edge (Figure 2.3). This is iterated until no edges below the given threshold remain, resulting in a simplified building footprint.

Another methodology is the classification of map objects into a set of simple shape classes and subsequent replacement by simpler, prototypical template objects. An example for a template-based approach is presented in (Rainsford and Mackaness, 2002). The algorithm classifies the building footprints' shapes into letter shapes, e.g., L-, U-, T-shapes, and replaces them by template footprints adapted to best fit through rotation, scaling, and translation (Figure 2.4).

Scale-space and curvature-space methods are based on mathematical morphology, which was first defined for images to smooth them or remove noise. These methods rely on morphological

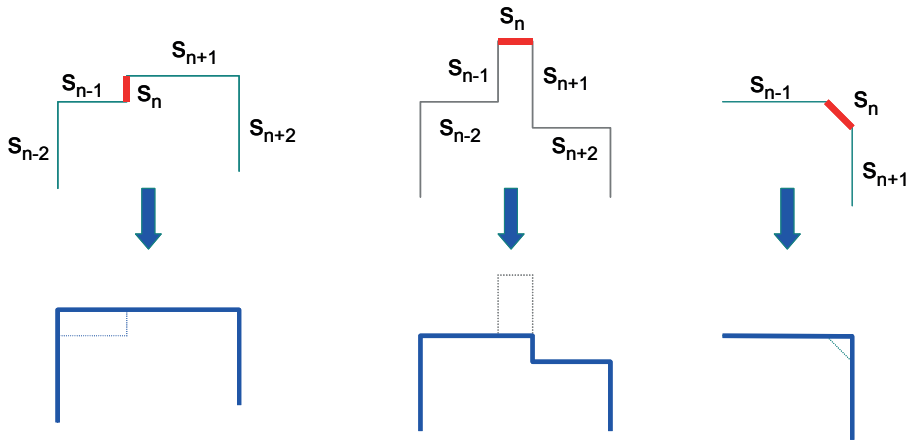


Figure 2.3: Depending on the local curvature at short edges (red), the cases *offset*, *intrusion/extrusion*, and *corner* are detected and simplified by removing the short edge (left to right, figure from (Sester, 2000b)).

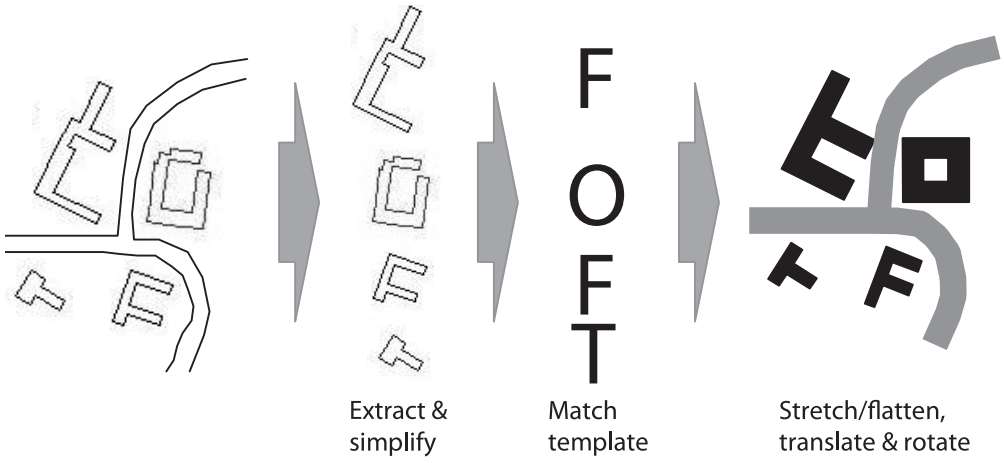


Figure 2.4: The original building footprints are simplified and matched with templates. At the position of the footprints, prototypical shapes are placed and adapted through basic transformations (figure from (Rainsford and Mackaness, 2002)).

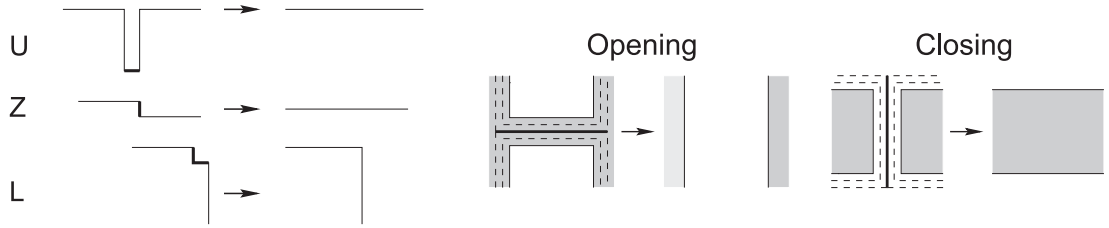


Figure 2.5: Left: scale-space operations lead to internal events, i.e., structure of one feature is changed through movement of edges. Right: they also lead to external events, when shapes connected by narrow parts are separated during opening, or neighboring shapes are connected during closing (figure from (Mayer, 1998a)).

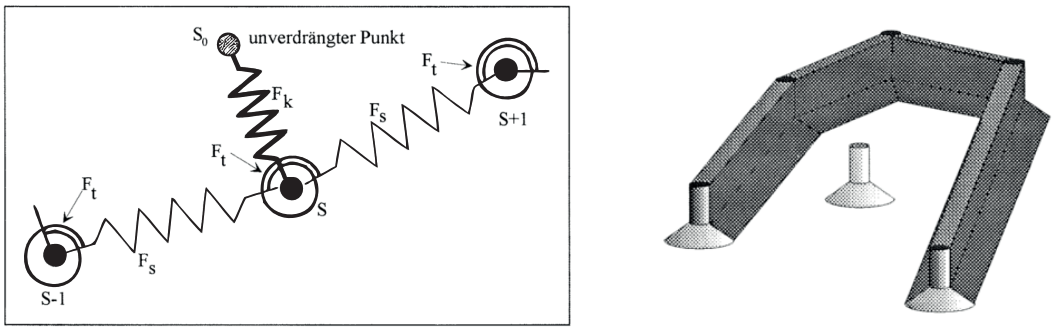


Figure 2.6: Springs model the distance and relative orientation of vertices of a linestring S-1, S, S+1. Displacement by S_0 is implemented as another spring F_K effecting a force on the line string (left) (figure from (Bobrich, 1996)).

operations defined on polygons. There are basic morphological operations, i.e., erosion and dilation, and composed morphological operations, i.e., opening (erosion followed by dilation) and closing (dilation followed by erosion). During their application, scale-space events occur which change the geometric structure, e.g., by removing small shape parts or merging neighboring shapes. An example is presented by Mayer (1998a), where dilation and erosion are defined by moving edges along and against the direction of their normal, respectively (Figure 2.5). Closing a polygon leads to elimination of concave elements such as U-structures, while opening a polygon eliminates small protrusion elements.

A different approach is the application of machine learning techniques, which provide a formal framework for automatic categorization / detection of patterns. A number of geometric, topological, semantical, and thematic properties can be used to automatically classify the map features. As an example, Sester (2002b) uses artificial neuronal networks, more specifically Kohonen feature maps, to typify the distribution of building centroids. Initially a reduced number of centroids is selected randomly which represent connected neurons. The neurons are attracted by all building centroids which act as stimuli. Iteratively a random building position "fires" an impulse and the nearest neurons respond by moving towards the impulse. As the learning rate decreases, the neurons converge to fit the distribution with reduced number of elements.

Physically based techniques express the generalization problem through a model from physics. As an example not restricted to building models, Bobrich (1996) uses springs to model geometric relations of map features, including curvature, neighborhood, and distances. In the approach map features create a displacement "mountain" that applies force to other point and line features within reach. Simultaneously, springs opposing the force draw the features back towards their original position and shape. The map configuration is created by balancing the map features and their forces. Spring tensions are set individually for different feature classes to reflect their priorities.

For smooth transitions between different levels of scale, generalization operations need to be applied continuously. Sester and Brenner (2004) deconstruct a complex generalization operation

into a sequence of elementary morphable generalization operations. The sequences are constructed in a preprocessing step. At runtime, they are served as requested by the viewer application. Hence, the building features are morphed smoothly into the simplified or more detailed shape.

Optimization techniques are used in generalization to cope with conflicting goals, e.g., because of many map features fighting for space. An example for iterative solving is the building footprint displacement technique of Ruas (1998). First, the topology of buildings and roads is constructed to identify map objects that are too dense, and possible movement directions. Then, building footprints that need displacement are displaced one after another, starting with the biggest conflict, i.e., overlap. If the conflict persists, other movement directions are tried until no alternatives remain. In this case, the initial density was to high, and the building footprint is removed to relax the local density.

Iterative solving is suitable for finding a good solution, however algorithms might get stuck in a local minimum. To overcome this limitation, the application of simulated annealing was suggested by Ware and Jones (1998), using the analogy of cooling metal. In their application the authors define a cost function expressing the quality of the map, e.g., conformance to minimal distances. In addition a displacement vector template is defined storing trial positions for each building footprint. Iteratively, one random change is applied to the map features, potentially being accepted as an improved solution, i.e., it has less cost. However, even a degradation, i.e., an increased cost, can be accepted with a certain probability. The probability depends on the current simulated temperature and decreases as the process evolves.

A way to find an optimal solution in problem space is to calculate the global maximum / minimum of the cost function. An example applying least squares adjustment (LSA) is presented by Sester (2000b) to simplify and displace building footprints. First, the rule-based simplification (Staufenbiel, 1973) as described above is applied to the original footprint to yield a simplified and approximate footprint as a first solution. Then, the approximate footprint is expressed in parametric form supporting rectangle and L-shaped footprints. This enables to put them into an equation system together with the original coordinates and additional constraints, e.g., minimal distances to neighboring objects, yielding a overdetermined system. Using least squares adjustment an optimal solution is computed.

Automated map generalization has evolved from single operators to holistic generalization systems being used by the mapping agencies. Though the NMAs report difficulties and automation is not complete, yet, it is increasingly applied with success. In addition, complementary fields are getting into the focus of research through new technologies.

2.3 3D Generalization

Along with the growing interest in 3DCMs in the last decade, also generalization of their components has come into focus. In the following we concentrate particularly on generalization of virtual 3D building models. So far a number of concepts and techniques exist that simplify models. However, theoretical foundations specific for 3D generalization are still largely missing.

2.3.1 Generalization in CityML

The CityGML specification takes into account generalized models as a central aspect of 3DCMs (Gröger et al., 2008). It provides an explicit level of detail (LOD) concept for its principal components (buildings, terrains, transportation network, water bodies, ...) as well as an implicit and generic generalization relation.

Figure 2.7: The CityGML specification contains these examples to illustrate typical use of its five consecutive levels of detail (LOD) (figures taken from (Gröger et al., 2008)).

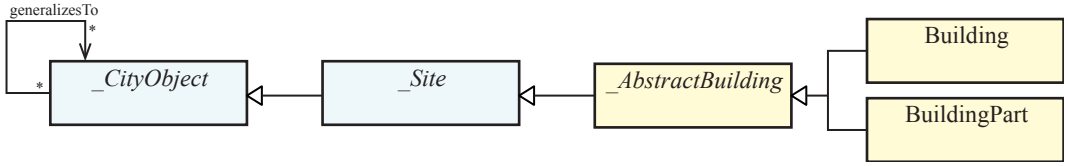


Figure 2.8: With the *generalizesTo* relation available at the central CityObject, CityGML offers a generic mechanism to describe multi-scale representation. It does not limit the cardinality of the linked objects, thus, $1:1$, $1:n$, and $n:m$ relations are all possible (figure adapted from (Gröger et al., 2008)).

CityGML defines for city objects five LODs (LOD-0 to LOD-4, Figure 2.7) and requires that geometric and thematic aspects of a city object are described in context of one of the LODs. Thus, the same city object may be described by multiple models. It is an explicit definition, since the standard describes properties and gives examples in a taxonomy of the LODs (Figure 2.9):

LOD-0 should be used for regional scale and contains a 2.5D terrain model with a surface texture applied. Building footprints and roof edges will be part of the next CityGML version 2.0.

LOD-1 additionally contains prismatic block building models with flat roofs and no façade textures.

LOD-2 contains buildings with differentiated roofs as well as thematically and geometrically differentiated surfaces, including textures. Vegetation objects may be included.

LOD-3 contains highly detailed architectural buildings with high resolution textures as well as highly detailed vegetation and transportation objects.

LOD-4 adds interior structures to buildings, such as stairs or furnitures.

The standard also provides details for minimum sizes and geometric point accuracy of objects which are required to be represented by a certain LOD. However, these requirements are not formalized and the authors rather present them as proposals for discussion.

The generic generalization relation is the second mechanism to describe multi-scale models in CityGML (Figure 2.8). It is a generic specification since it does not define requirements for its use. It is specified for *CityObjects* to map them to other *CityObjects* with a "generalizes-to" semantic. Thus, various generalization hierarchies can be created, including 1:1 relations beyond the five LODs, n:1 relations as demanded by aggregation, or n:m relations as demanded by typification.

The CityGML standard is flexible in that it does not constrain how multiple representations of a 3DCM should be used:

- Combinations of different LOD representations of buildings within the same scene is possible. For example, landmark buildings can be represented in a higher LOD than residential buildings.

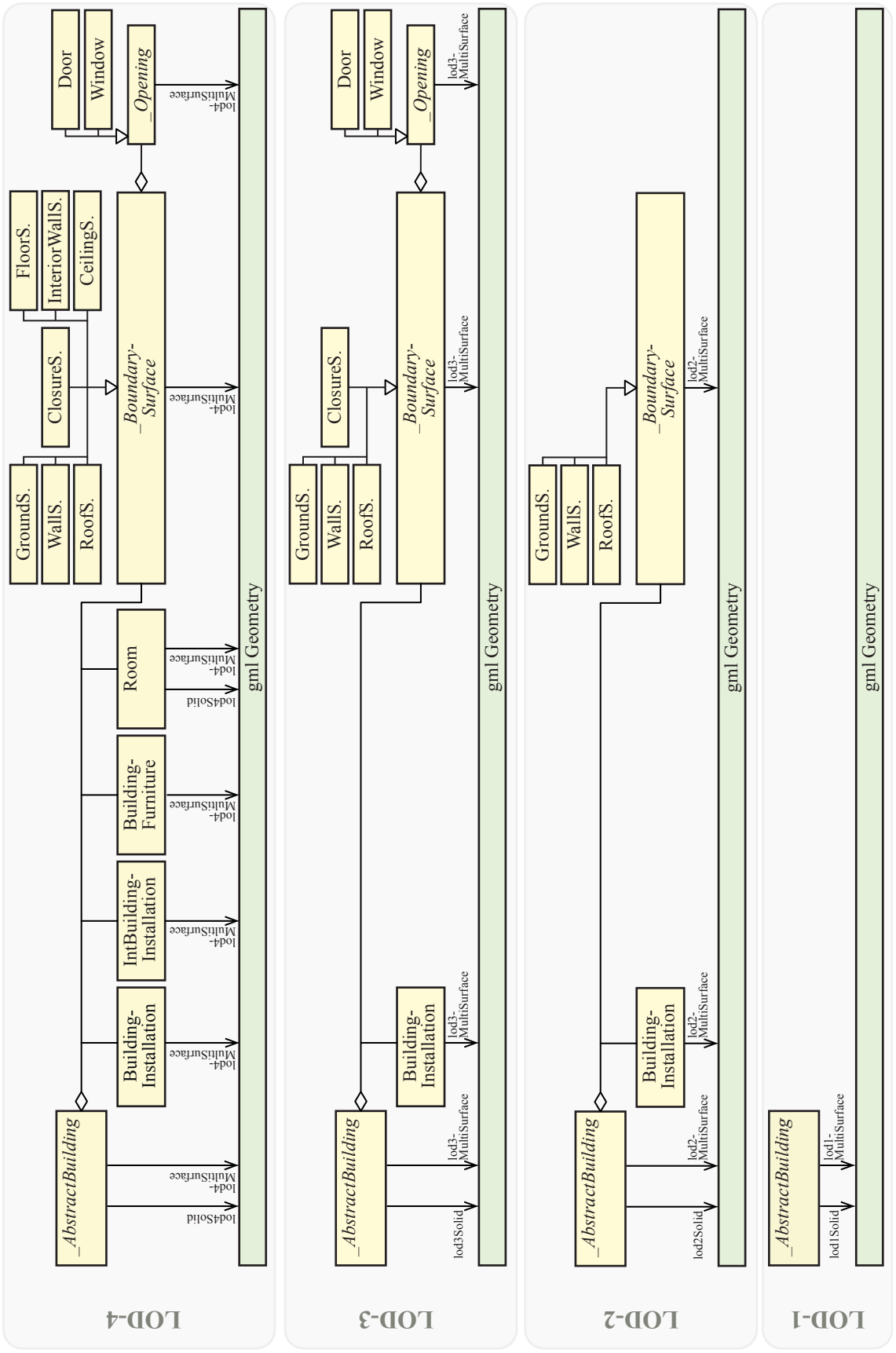


Figure 2.9: For building models, CityGML specifies the required elements, geometric resolution, and thematic completeness for each level of detail. Thus, multiple representations of a single building model can be linked with a generalization relation. LOD-0 does not show building geometry (figure adapted from (Gröger et al., 2008)).

- Combinations of different LOD representations of the relief model is possible. CityGML supports this by validity extents that describe how to integrate relief components. For example, a coarse grid can be used for contextual area, while a focus region is represented using a high resolution TIN representation.

- CityGML introduces the concept of the terrain intersection curve (TIC), which describes the interface between a feature, e.g., a building model, and the terrain. This enables features represented in different LODs and different terrain LODs to be combined, as applications can locally adapt the terrain model to embed the feature. For example, a building model can be combined with a low or high detail terrain model and will always be integrated without floating or sinking into the terrain.

While the specification of LODs marks an important step forward in the categorization of 3DCMs and their components with respect to their resolution, CityGML also has a couple of weaknesses regarding its generalization model:

- The formalization of the LOD concept includes only the general elements of each LOD and their storage. Desired properties for each LOD, e.g., absolute and relative point accuracy, are given only partly for buildings and vegetation, and only in a verbal form. Other components as well as formally defined requirements are left for future work.

- Pointers to algorithms that are able to generate appropriately generalized CityGML 3DCMs are not presented. Due to the required properties for each LOD not being defined, it is not clear how to design such algorithms or which existing algorithms would be capable.

- The generalization model offers high flexibility in describing generalization relations and combining different components in different LODs. However, it also offloads the problem of how to use a complex generalization hierarchy to applications implementing CityGML. For instance, generalization relations may be specified both using LODs and the generalizes-to relation. In this case, the application needs to decide, how to use the different relations, e.g., which representation is used when. As complex generalization hierarchies can be defined, this can become a non-trivial task, e.g., in case of unbalanced generalization hierarchies. In these cases additional attributes are needed to guide the applications, such as validity intervals describing the intended scale. Also pointers to best practices beyond simple example datasets would help to clarify the usage.

- The LOD concept is derived from observation of existing models, which is reasonable especially in the case of building models. However, regarding relief models the five LODs seem chosen arbitrarily and it is not clear how they differ exactly. In practice terrain models will have many intermediate representations. For terrain models, simplification is well understood from an algorithmic point of view, see, e.g., (Pajarola and Gobbetti, 2007), however it is not clear how to map these to the five LODs of CityGML. For example, LOD-4 adds interior structures to LOD-3 building models, while the outer geometry remains the same. Does this implicate the same resolution for LOD-3 and LOD-4 terrain models?

With these issues not being formalized, it is the task of applications which implement and use CityGML to establish, store, and apply conventions. These conventions include the degree or aspect of generalization, interdependency with the explicit LODs, and how to build up and make use of the generalization hierarchy. Generic attributes definable in CityGML could be used to support this and to make the conventions explicit.

Visualization software such as LandXplorer CityGMLViewer[1], or Aristoteles[2] support CityGML. Up to now, they typically let the users select the LOD for visualization. The implicit generalize-to relation is not evaluated, all representations are rendered at the same time.

To conclude, regarding generalization the CityGML specification enables integrated modeling of different LOD representations within 3DCMs. The LOD concept of CityGML has established a common taxonomy for the resolution of building models. The authors of the standard regard it as the most often cited aspect of CityGML (Gröger and Plümer, 2012). However, for future developements of 3D generalization and usage of generalized 3DCMs, more formal requirements of each LOD, their interdependency, and naming of explicit algorithms to create and use them are desireable.

2.3.2 Generalization Techniques for 3D Building Models

A number of techniques for the generalization of 3D building models exist addressing specific requirements and challenges of building simplification (Meng and Forberg, 2006; Sester, 2007). Usually, 3D building models already have relatively simple geometry. Application of standard surface simplification methods from computer graphics often results in lost general appearance and characteristics of the models, e.g., orthogonality and parallelism. In addition semantic attributes need to be taken into account during simplification. The following work represents approaches to handle these characteristics.

Generalizing Single Building Models

Single building model generalization techniques process a building model independent from their environment. Potential visual and topological inconsistencies of the generalized model in its context have to be handled afterwards as in, e.g., (Peter et al., 2007).

The building generalization technique presented in (Kada, 2002) simplifies boundary representation (BRep) models using a set of rules. First, a number of constraints between faces are set up to ensure building characteristics are maintained, e.g., parallelism, orthogonality, and symmetry. Then, surface features on the model are detected and simplified, similar to the rules of Staufenbiel (1973). Finally, the simplified model is optimized towards the original model using least squares adjustment.

In another approach Kada (2005) suggests reconstruction of a building model using half spaces. For each wall face of the original model, the algorithm creates a plane and a related buffer (Figure 2.10). Starting with the face with the largest area, the algorithm merges faces within a given maximum distance to the current face's buffer, adapting the plane's parameters and leading to a smaller number of planes. The final set of planes is used to create a cell decomposition of the building. Cells that do not cover a given percentage of the original building area are discarded, the remaining cells are used to extrude 2.5D geometry. To complete the simplification, roof faces are reconstructed separately for each cell using half spaces and finally merged to the resulting generalized building.

Kada (2011) extends his half space approach to support aggregation of building models. The adapted technique is applied to multiple buildings and applies morphological closing to enforce merging of close geometry.

Rau et al. (2006) present an approach working on building models comprised of prismatic shapes with sloped roof structures. First, the roofs are flattened and adjacent polyhedrons are merged if their height difference is smaller than a given feature resolution, yielding 2.5D shapes.

[1] discontinued by Autodesk as of 2011 but still available on `http://www.citygml.org/index.php?id=1538` (accessed 1.7.2012)

[2] `http://www.ikg.uni-bonn.de/aristoteles` (accessed 1.7.2012)

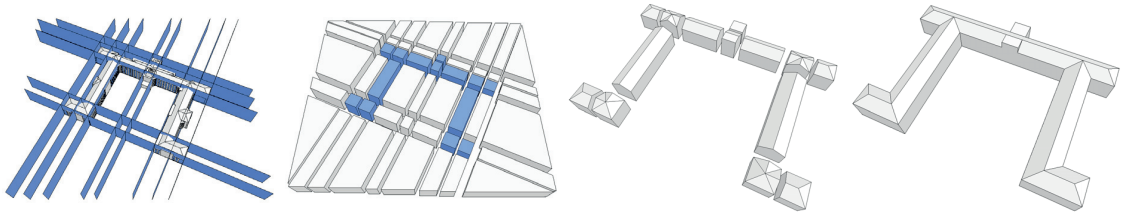


Figure 2.10: A small set of half spaces is determined that decompose the building model into major components. For each component, the roof geometry is reconstructed, and finally adjacent components are merged to the simplified building model (adapted from (Kada, 2005)).

Then, the footprint of each shape is simplified by collapsing small edges (walls). To retain the principal structure, vertices belonging to the longest edges are not allowed to move. Feature resolution is the minimum distance by which to control the simplification process. It is used at runtime to make the selection of the appropriate representation for visualization.

Thiemann (2002) focus on decomposition of a building model into its principal shapes, following an approach of Ribelles et al. (2001). Using planes built from the model's faces, the model is decomposed into smaller parts. A quality parameter controls the selection of intersection planes and the order of their application. The resulting shapes together with a Boolean operation are stored in a constructive solid geometry (CSG) tree. Minimum shape size can serve as a parameter to control the selection of displayed building components for a simplified building model.

As part of his conceptional framework to generalize 3DCMs, Stüber (2005) applies the same algorithm to decompose building models (Ribelles et al., 2001) and sorts the resulting components into a cluster graph. During visualization the components are rendered or omitted based on their visible size and minimum visibility requirements.

Using 3D templates the technique of Thiemann and Sester (2006) simplifies and typifies categorized building models by replacing them with an adapted parameterized prototype shape. The authors focus on the adaptation step, assuming categorization and template shapes are given. To guide the process, they suggest minimizing distances between randomly sampled surface points and performing least squares adjustment to adapt the prototype to the original building model.

Building on earlier 2D methods, Mayer (1998b) analyzes morphological scale-space and curvature-space operators in context of 3D building models. Scale-space operators are defined as moving faces in or against direction of their normal, either uniformly by the same distance (morphological operators) or depending on the local curvature (curvature-space operators). While morphological operators can fill gaps or connect neighboring building parts, curvature-space operators can simplify stair structures. The distance by which to shift the faces can be used as a parameter controlling the simplification. Further research in that direction by Forberg and Mayer (2002) addresses orthogonalization as a requirement for effective employment of morphological operators and the identification of concave / convex structures.

Forberg (2004) introduces another scale-space based on parallelism to generalize earlier findings and combine characteristics of both morphological and curvature space operations. The algorithm identifies parallel faces of the model and, starting with the smallest distance, moves faces towards each other so that they share the same plane (Figure 2.11). The moved faces result in merging building parts, removal of protrusions or adjustment. Again the maximum distance by which to shift the faces can be used as a control parameter.

The approach of Fan et al. (2009) is directed at generalizing CityGML LOD-3 building models through the exterior shell. In the technique all the polygons belonging to one wall are projected to the farthest of its polygons' planes; polygons that are not parallel or coplanar are discarded. Thus, the façade structure with window and door openings is maintained. Similarly the roof polygons are clustered by their orientation and projected to the farthest plane of each cluster's polygons.

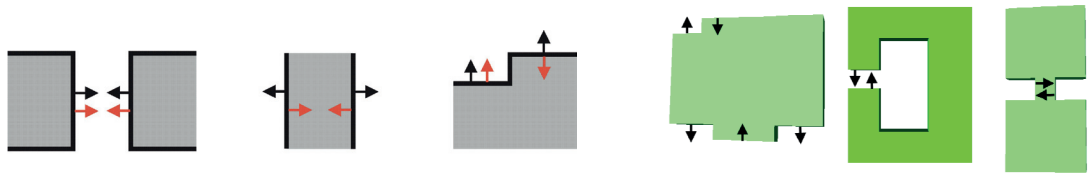


Figure 2.11: Through moving close parallel faces (normals are shown in red arrows) towards each other (move direction is shown in black arrows), faces eventually merge (left, taken from (Forberg, 2005)). The technique is shown to remove convex structures, gaps, but also small connecting parts (right, (Forberg, 2004)).

In his thesis Fan (2010) suggests another approach for the computation of CityGML LOD-2 building models. First, the building footprint is simplified using rules from Staufenbiel (1973), extended by rules to handle non-orthogonal curvature. Second. the roof geometry is generalized: individual polygons are simplified using the same rules, then aggregated if quasi coplanar, and again simplified. Third and finally, the generalized footprint is extruded until it meets the generalized roof geometry, thereby applying extensive special handling to avoid gaps and geometric errors.

Coors (2001) applies an adapted surface simplification algorithm (Garland and Heckbert, 1997) to simplify single buildings. Introducing dominance values on important parts of the building, e.g., a bell tower of a church, the simplification algorithm is adapted to conserve these parts while simplifying geometric complexity of the remaining model.

Generalizing Surface Structures of Buildings

Sester and Klein (1999) report on façade generalization including aggregation and emphasis of façade elements. With the prerequisite of having a semantically detailed building model, 2D generalization operations are employed on façade elements, e.g., aggregating or typifying rows of windows, enlarging and displacing single windows.

For the special case of high detail surface tiles, such as roof tiles or wall bricks, Guercke et al. (2004) suggest a typification technique to create simplified LOD models. Using least squares adjustment, a Bézier surface approximating the original surface is computed. Then the tile distribution, e.g., number of rows and columns, is detected. For the selection of the optimal LOD model, perception measures are suggested by the authors that adapt the tile density in the LOD models to the depicted size.

While not a generalization operator, the detection of façade elements is an important prerequisite for further typification. Generalization operators such as selection, typification, and aggregation depend on semantically enriched models. Thiemann (2005) suggests applying rules that model topological knowledge on geometric building components to classify them, e.g., into windows, dormers, and balconies. Ripperda (2008) reports on the extraction of façade attributes from photographs using a stochastic process guided by a formal grammar.

Generalizing Multiple 3D Building Models

Applying the 2D building generalization framework presented earlier, Sester (2002a) presents a 3D visualization of simplified, aggregated, enhanced, and displaced building models by extruding the generalized building footprints to solids. As a use case, adaptive generalization is applied to building models depending on their distance to important building models.

Anders (2005) aggregates linear groups of building models, e.g., along a road. The proposed algorithm first projects the building geometry onto its principal axes, yielding three sets of footprints. Then for each set, the footprints are aggregated and simplified using 2D generalization operations (Sester, 2000a). Finally, the simplified footprints are extruded and the resulting shapes are intersected, yielding the final aggregated building model.

Similarly, Tsai et al. (2008) aggregate compact building groups by projecting and rasterizing their models along the principal axes. After performing morphological operations on the images, the top raster image is segmented according to the side images. A simplified 3D shape is reconstructed by identifying vertices and edges, creating polygons from the raster images.

A conceptional framework based on pattern recognition is presented by Raheja (2005). On a micro (per building), meso (neighborhood), and macro (building clusters) level, formalized properties, e.g., building type, size, relative orientation, and proximity, are acquired following a bottom-up approach. The properties are used to guide the generalization, e.g., to identify building groups to be aggregated.

An approach especially addressing the grouping of LOD-1 building models as a prerequisite for aggregating them is presented by Guercke et al. (2011). Properties of the individual buildings, e.g., relative and absolute height, are taken into account to decide which neighboring buildings should be aggregated. The decision task is formulated as a mixed-integer programming problem that minimizes conditions, e.g., the difference between the summed original and aggregated building models' volume. In contrast to heuristic, greedy algorithms, the resulting building groups are optimal with respect to the given conditions.

A conceptual framework for 3D building generalization is suggested by Guercke and Brenner (2009) that is based on a hierarchical feature model of a city and its components. For each feature different implementations of generalization operators can be selected, while a central coordination instance is responsible for resolving conflicts. The semantical structure expressed by the feature hierarchy forms the basis for a generalization workflow presented in (Guercke et al., 2009).

2.4 Generalization in the Visualization Pipeline

The visualization pipeline (Figure 2.12) describes the different stages in the process of visualization (Haber and McNabb, 1990; Ware, 2004): original *raw data* that was obtained by measurements or from simulations is preprocessed, enriched, and selected in the *filtering* stage yielding derived, *processed* data. In the subsequent *mapping* stage the data is prepared for visualization by mapping it to visual variables, e.g., color, size, and shape (Bertin as discussed by Carpendale (2003)) for most expressiveness. The resulting *mapped data* is used to create images in the final *rendering* stage.

In context of computer generated images, the process can be controlled by the users at several levels and feedback loops: they can adapt the rendering

- through direct interaction, e.g., by changing the view parameters,
- by adapting the mapping, e.g., by selecting different colors, and
- by controlling the filtering stage, e.g., by loading a new data set or changing preprocessing parameters.

Thus, interactive visualization enables the user to become both author and recipient.

Generalization as a transformation of the model and its visual representation can be applied within the visualization pipeline at all stages (Glander et al., 2011). During filtering, generalization operations G_1 such as selection, aggregation, simplification and aggregation can be applied to the original geodata of a virtual 3D landscape model, leading to a generalized landscape model. This aligns with the application of model generalization which aims at reducing data size and selecting the necessary features, following the purpose of the visualization. During mapping the landscape model is transformed to a visual representation through symbolization and styling. Generalization operations G_2 such as exaggeration, enlargement, and displacement are applied in this stage, leading to a cartographic landscape model. Similarly, generalization operations G_3 can be applied during

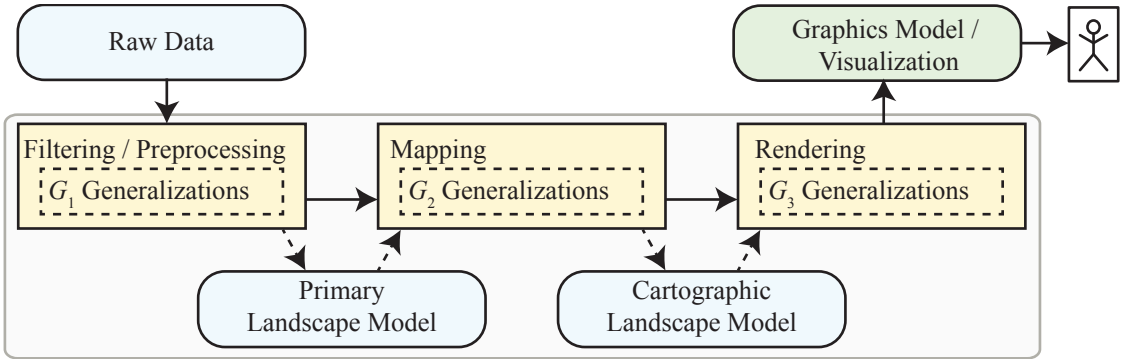


Figure 2.12: The visualization pipeline is the conceptual model explaining how to transform data into comprehensive images. The figure shows generalization applied as part of the visualization pipeline.

rendering stage to adapt the graphical representation and enforce cartographical principles as the user changes view parameters, e.g., camera position, and projection.

In practice, concrete generalization techniques use a combination of these generalization operators. In particular, if they have to provide adaptive, dynamically generalized models for interactive 3D visualization systems, generalization operators for all three steps are required. New visualization techniques may introduce new generalization operators that address specific challenges in 3DCMs.

- Geometric scale in perspective 3D visualization appears as a continuous range in a single image. In contrast, classical maps usually show a single cartographic scale at a time; a scale change may imply topological changes, e.g., when a group of buildings is aggregated to a single settlement area. To provide smooth visualizations, temporal incoherence such as hard appearance and disappearance of building representations should be avoided.

  Morphing is suggested as a generalization operator specific for interactive visualization that performs a smooth transition between different representations (van Kreveld, 2001; Cecconi and Galanda, 2002).

- Interactive 3D visualization enable many views of the scene, including oblique views. In contrast, classical maps show top-down orthographic views. Hence, occlusion needs to be taken into account for cartographic visualization goals, e.g., highlight important structures and objects such as a route destination.

  Nonlinear perspective distortions are an approach to resolve occlusion (Pasewaldt et al., 2011; Böttger et al., 2008). Controlling the distortion by the distance of individual 3D objects to the virtual camera, objects can be shown that otherwise would be hidden.

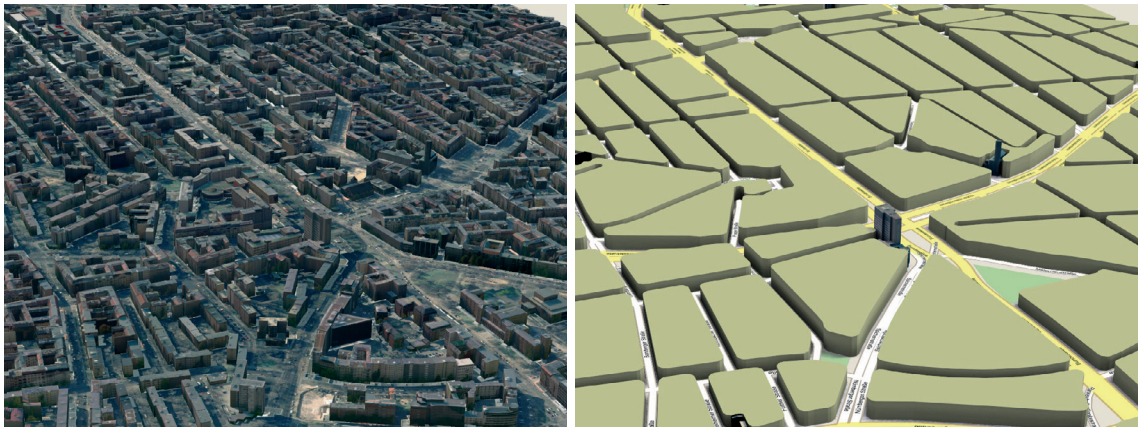


Figure 3.1: Cell-based generalization creates an abstract visualization of a given virtual 3D city model (left) by including aggregated building representations, infrastructure network, and non-building areas (right).

CHAPTER 3

Cell-Based Generalization

This chapter presents a cell-based generalization technique (Glander and Döllner, 2009). The goal of this technique is to create an abstract visualization of 3DCMs, taking into account building models, infrastructure network, and non-building areas (Figure 3.1).

3.1 Potentials and Usage of Cell-Based Generalization

We are observing increasing degrees of detail of 3DCMs driven by technological advances in remote sensing, modeling, and storing of geospatial data. This results in the development of many new applications but also presents challenges for the visualization, as described in the introduction (Chapter 1). For effective usage, the visualization of 3DCMs has to be improved beyond photo-realistic detail, as indicated by field studies (Oulasvirta et al., 2009). It can be instructive to look at methods used in map making. As a visual medium communicating spatial information, maps have a long tradition in solving the problem of depicting large amounts of information in limited space. A fundamental set of methods in the mapping process is generalization (Chapter 2.2).

Cell-based generalization, as presented in the following, is a technique to create abstract representations of 3DCMs (Figure 3.1) that are focused on giving a quick overview about the general structures of the city. Similar to topographical base maps that serve multiple purposes, the resulting abstract 3DCM is intended to facilitate multiple purposes. Examples of usage include the following:

- In car navigation scenarios it is important to limit visual clutter and concentrate on preserving only the essential information needed to drive. Being in a traffic situation the driver must be able to quickly comprehend displayed structures and align them with the current environment to make driving decisions.

- Tourist information terminals display the environment and highlight interesting routes and places, e.g., in public buildings, train stations, and airports. For quick overview tasks, the displayed 3DCM must show the general organization of the city while embedding points of interest in detail.

- For efficient visualization of arbitrary thematic data, e.g., air temperature, election data, and land use, the data is displayed together with a base model which serves to give spatial reference. The focus being on the thematic data, the base 3DCM has to be reduced in details to show only major structures.

The presented technique is inspired by the work of Lynch (1960), who describes five major elements forming a city's mental image: paths (ways through the city, e.g., streets), edges (barriers, such as coast lines and railway lines), districts (partitions of the city), nodes (e.g., central places, path crossings), and landmarks (distinguishable objects used for orientation). Therefore, a comprehensive representation of a city has to incorporate and even emphasize these elements for easy comprehension while allowing the user to connect real world structures with the displayed representations. We address this by using street network, water, railway and coast lines, as well as non-building areas of a 3DCM to create cell blocks. We assume that block cells can represent individual buildings abstractly. The cell blocks are further shaped by computational geometry operations and enhanced by landmark buildings, which are maintained in the visualization.

3.2 Generalization and Abstract Visualization of 3DCMs

In addition to general work on map generalization and 3D generalization (Sections 2.2 and 2.3), there are approaches related to the presented technique in industrial / commercial applications and research specifically addressing generalization and abstract visualization of complete 3DCMs. We will briefly review both in the following sections.

3.2.1 Commercial Applications

Commercial (car) navigation systems and mapping applications follow a trend to employ 3D in their visualization.

Most car navigation systems offer different 3D modes. They can be roughly classified into oblique 2D views, 2D views with 3D landmark buildings, and high detail 3D views. For the oblique 2D views, the road map is depicted on a plane that is viewed from an oblique angle and centered at the current position. Some products also have a 3D terrain model to visualize hills and valleys on the way. 2D views with 3D landmark buildings add textured geometry for important buildings on top of this. Residential buildings are shown as 2D footprints or grayed out 3D shapes. High detail 3D views aim at a photo-realistic visualization and show all buildings textured. They sometimes also include additional elements such as trees and traffic lights.

For commercial mapping applications, such as Google Maps or Bing Maps, there are two directions. Firstly, abstract two dimensional maps are enhanced with outlines of 3D buildings to communicate their heights. More recently, 3D building shapes are included, rendered as transparent shapes and with perspective projection in real-time. The visualization aims at adaptation to the scale: with increasing scale, buildings are first represented as footprints, then as oblique 3D shapes with reduced height, then with their full height. Secondly, photo-realistic perspective views of 3DCMs – either real-time renderings or oblique photographic imagery – are enhanced with text, icons, and rendered vector data.

Inclusion of 3D into commercial mapping or car navigation applications seems to be driven mainly by technology and customer demand. Using 3D in maps and map-like visualization does not seem to increase effectiveness (Oulasvirta et al., 2009). However, using visual landmarks presented with photo-realistic 3D models can increase recognizability (Elias and Paelke, 2008): this technique is also used in tourist maps. In general, abstract visualization is rather applied to 2D content, 3D geometry is rather not adapted to provide simplified visualization.

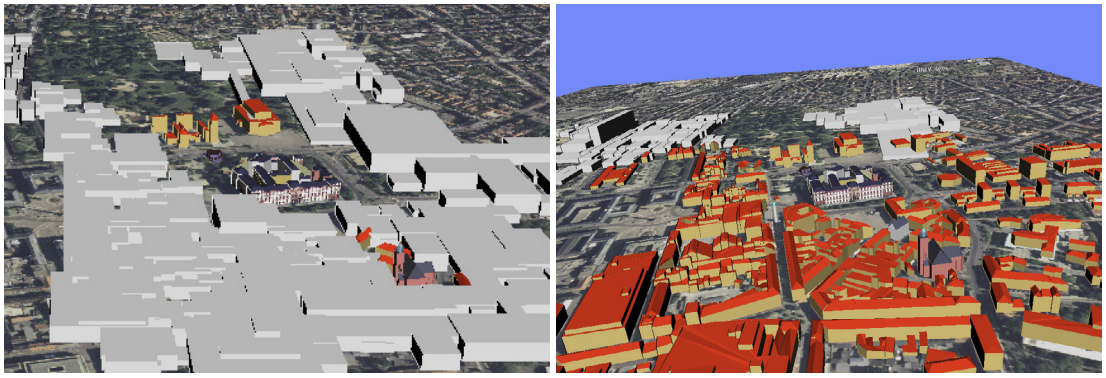


Figure 3.2: Buildings models are aggregated using boxes stored in an R-tree for efficient network transfer and visualization. Landmark buildings such as churches can be presented with a higher detail using dominance values (left). Depending on the current view point, the visualization can be adapted locally (right, images from (Coors, 2003)).

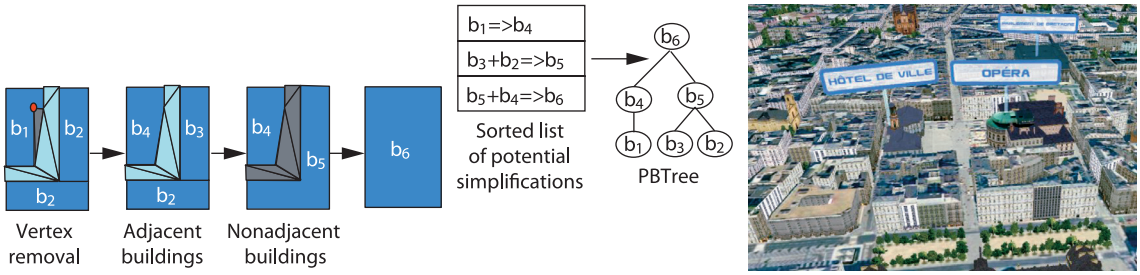


Figure 3.3: Building footprints are simplified and merged with adjacent and non-adjacent buildings. Using a cost function, the potential simplifications are sorted and stored in a hierarchical structure (from (Royan et al., 2007)) for efficient photo-realistic visualization.

3.2.2 Generalization and Visualization Techniques for 3DCMs

In academic research several techniques have been proposed for generalization and interactive visualization of multi-scale 3DCMs.

Targeting transmission of 3DCMs in networking environments, Coors (2003) proposes the use of R-Trees storing an additional height value to aggregate buildings (Figure 3.2). The bounding boxes represent aggregated buildings in low quality, facilitating fast transmission and visualization, while landmark features are preserved through dominance values. Also, view-dependent refinement uncovers detailed building models.

The technique of Royan et al. (2005) processes 3DCMs containing 2.5D building models to get a hierarchical representation usable for progressive transmission. The algorithm applies simplification operations to the 3DCM: footprint simplification by vertex removal, aggregation of adjacent buildings by edge removal, and aggregation of non-connected buildings guided by a cost function (Figure 3.3). Non-connected buildings are aggregated by performing a triangulation of the non-building space to identify potential triangles that fill the gaps between buildings. The metric to choose which triangles to merge incorporates building height, building elevation, and road triangles. If two buildings are merged, the height is averaged based on the footprint area. The operations are sorted by the assigned costs yielding a scheme for progressive transmission and visualization. The technique is demonstrated together with large terrain models in a networking environment (Royan et al., 2007).

Also aiming at efficient visualization structures while maintaining legibility, Chang et al. (2008) clusters and aggregates 2.5D building models, storing a LOD hierarchy for efficient rendering. First, building footprints are clustered hierarchically based on a distance metric taking into account Euclidean distance and cluster size. Then, for each node, the clusters are aggregated by computing

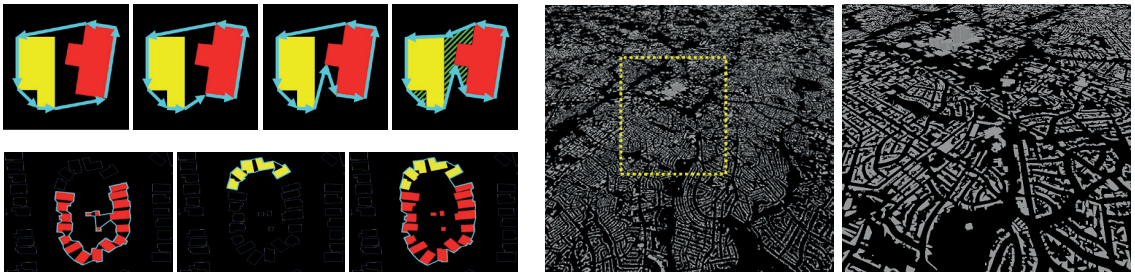


Figure 3.4: The legible simplification technique aggregates footprints of building clusters using a shrink wrap hull (left). Without zoom the simplified building clusters are not distinguishable from the original building models (right, images from (Chang et al., 2008)).

a 2D convex hull that is shrunken iteratively (Figure 3.4). The resulting cluster polygon is extruded to the weighted averaged height of all contained buildings, and textured. Individual building models significantly higher than the average are considered as landmarks and rendered in addition to the cluster geometry.

In (Chang et al., 2007), the method is applied for focus-dependent visualization of 3DCMs that maintains the legibility of the city. The paper concentrates on visualizing urban statistical data upon an abstracted 3D representation of the city together with 2D information visualization tools, e.g., parallel coordinates plot.

As part of the Mona3D project, Coors and Zipf (2007) present a visualization of 3DCMs focusing on mobile devices and based on an open standards infrastructure. To save bandwidth and take into account limited computational resources on mobile clients, geometry is simplified and façade textures are generated procedurally. To differentiate the ground surface into, e.g., river, forest, and built-up areas, styling rules can be defined. Following this line of research, Schilling et al. (2009) integrate OpenStreetMap (OSM) data and offer 3D visualization of large areas of Germany. Their technology has also been applied to global OSM data.

In their visualization framework for 3DCMs, Mao et al. (2011) apply simplification on individual buildings and aggregation to create a hierarchical level of detail structure called *CityTree*. The aggregation is performed by connecting 2D footprints of neighboring buildings: they define a criterion to select merge edges and combine them using a trimmed convex hull (Figure 3.5). The aggregated height is stored in the nodes to support reconstruction of extruded 3D shapes for visulization.

The focus is put on communication of spatial urban models in the work of Döllner and Walther (2003), who discuss 3D geovisualization and 3DCMs in the context of cartographic information display. The authors emphasize the use of non-photorealistic (NPR) styles such as reduced number of color tones, edge enhancement, and procedural façade textures, to create expressive, illustrative visualization of 3DCMs.

With a similar motivation, Pan et al. (2011) apply several NPR styles in a real-time focus+context visualization of 3DCMs. Their approach implements an image-based composition of the different NPR styles based on intensity values defined for the geometry. They discuss the application of single- and multi-focus and how to blend between focus and context areas using different distance metrics.

In the work of Grabler et al. (2008), a system for the creation of static tourist maps is presented. The authors introduce and apply to the model a complex metric to measure the semantic, visual, and structural importance of elements of the city in question. For the creation of city maps, they integrate a number of techniques for building simplification, optimization, displacement, and labeling. This approach creates pleasing but static oblique images of a city, while we focus on interactive 3D visualization.

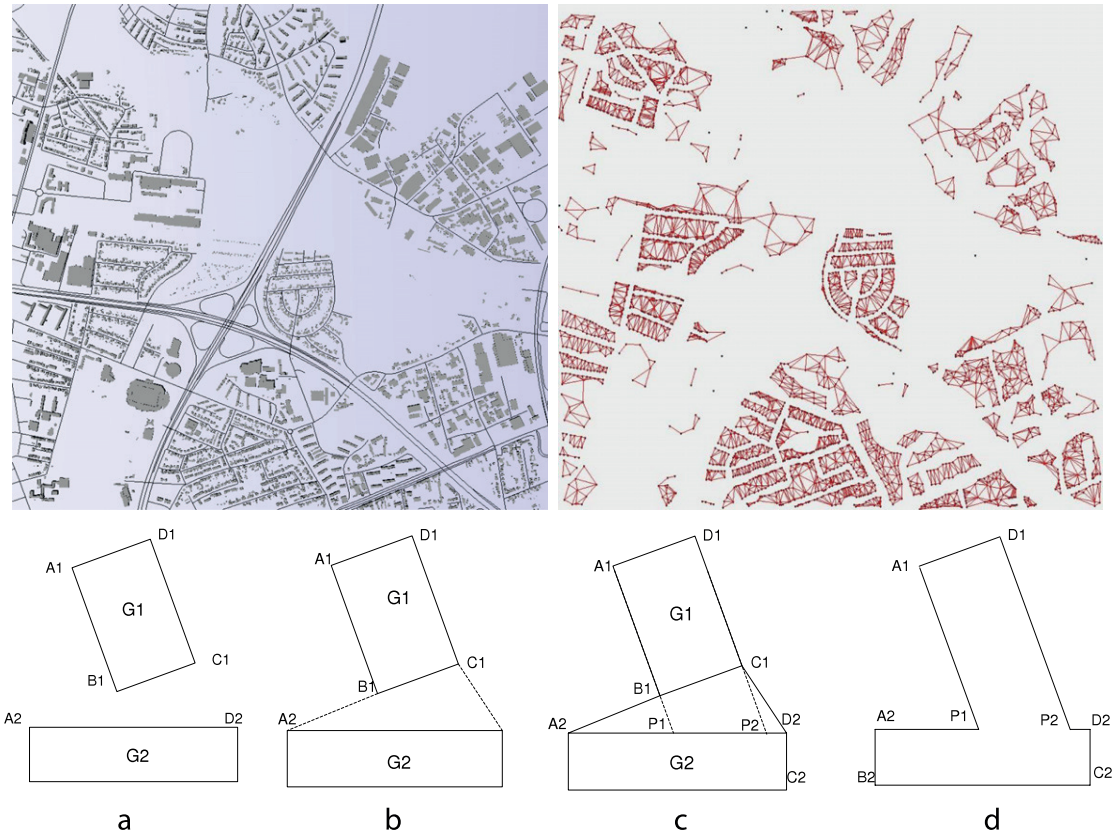


Figure 3.5: The input building models' (upper left) centroids are triangulated using a Voronoi diagram, which is used to reconstruct the neighborhood relation. Neighborhood edges crossing a road or longer than a given limit are discarded (upper right). The aggregation of neighboring building footprints fills the gap between them (lower row, all figures from (Mao et al., 2011)).

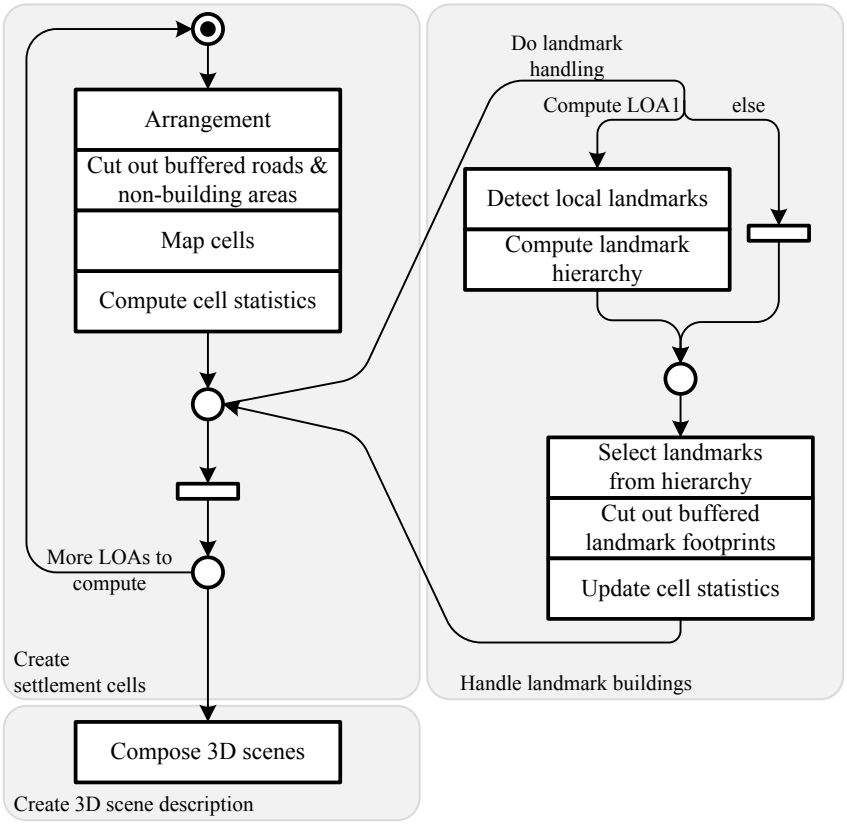


Figure 3.6: The petri-net shows the sequence of general operations for the cell-based generalization technique as well as extensions to create multiple LOAs, handle landmark buildings, and the creation of the 3D scene for visualization.

3.3 Concept Outline of Cell-Based Generalization

This section describes the algorithm to transform a 3DCM to a generalized representation called level of abstraction (LOA) as visualized in the diagram Figure 3.6. Multiple single buildings are replaced by settlement cells, i.e., block cells enclosed by infrastructure or non-building elements. The input infrastructure network is intersected with the contours of non-building polygons to create an arrangement containing polygons (Berg et al., 2000). These polygons are further processed by cutting out buffered streets and non-building polygons. To remove small sliced polygons introduced by Boolean operations, morphological closing is performed.

In the next step, buildings are mapped to those cell polygons they overlap most; then the mean height and variance for buildings of a single cell are computed and stored. The mean height will be used later to extrude polygons to 3D cell blocks, and communicate the average building height of the abstracted buildings. This mapping is stored by the generalization relation.

We define the generalization process on generalization features G, where the original buildings are set as input generalization features G^0, together with the infrastructure features I, and non-building features N. The results are G^1:

$$G^1 = \text{cellGen}(G^0, I, N)$$

After the presentation of the technique in the following, we demonstrate its application to the computation of multiple LOAs (Section 3.6), handling of landmark buildings (Section 3.7), and the creation of the 3D scene for visualization (Section 3.8).

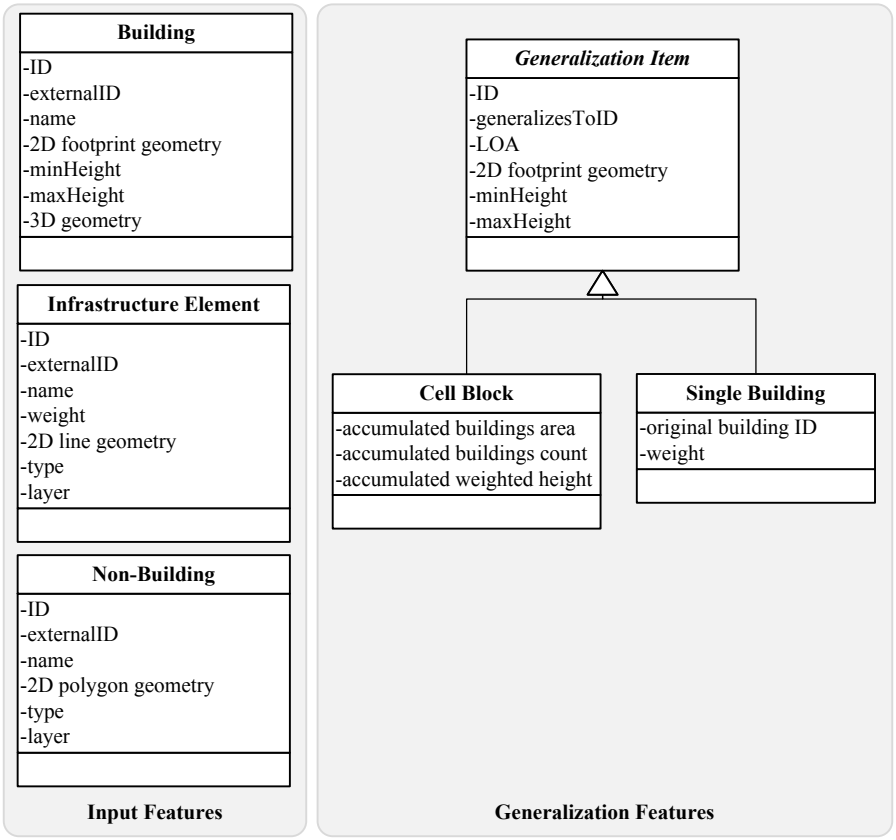


Figure 3.7: As input data we use building, infrastructure, and non-building features, where each feature comprises 2D geometry plus meta-attributes (left). The cell-based generalization creates generalization items to track LOA, averaged attributes such as height, and reference to the original building for landmark buildings (right).

3.4 Data Model

We refer to a 3DCM that is described by a 3-tuple $CM = (B, I, N)$ with a set of building models B, a set of infrastructure elements I, and a set of optional non-building areas N. In the following, we describe the possible properties and appearances of each of the input components and the resulting generalized 3DCM (Figure 3.7).

3.4.1 Input Data

Building Models

3D building models B as a central feature class for 3DCMs are used in various LOD. CityGML defines a classification with five LODs and associates required geometric, semantic, and thematic properties for each level (Section 2.3.1). For the cell-based generalization techniques we need a common access to single buildings and therefore internally convert their geometry into 2.5D prismatic models, i.e., a 2D (multi) polygon together with minimum and maximum height attributes. This allows us to operate on 2D geometry as well as exploiting 2D spatial index structures for efficient queries. Additionally, a reference to the original 3D geometry and the external ID is stored, e.g., the original ID or name of the building. Original data sources may be one of the following:

Prismatic 2.5D building models are derived from cadastral databases, which store polygon footprints along with attributes, such as height, construction date, or ownership. They can be directly used during the import process. The typical file format is shapefile (SHP).

Table 3.1: Available road types in common datasets and their mapping to weight classes used in cell-based generalization (Navteq, 2008; TeleAtlas, 2006). We adopt the categorization into five main road classes as available in community-based as well as commercial data.

our weight w_i	5	4	3	2	1	0
OSM highway classes	motorway	primary	secondary	tertiary	residential	other
TeleAtlas network classes	5	4	3	2	1	-
Navteq functional classes	5	4	3	2	1	-

"Architectural" 3D building models as used in 3DCMs are high geometric detail representations, but usually lack semantic and thematic attributes. They may either be true architectural models created for newly constructed models, or be derived from high resolution remote sensing data. They often define object identity using structured scene graphs or identifiers, allowing us to extract footprint geometry and height attributes, and store a reference to the original geometry. There are also attempts to use standardized architectural models and its components, e.g., Industry Foundation Classes (IFC) (Benner et al., 2004). It is also common that multiple buildings plus additional scene objects are stored in a single unstructured "polygon soup", which implies manual classification in most cases. For automatic processing, object identity information needs to be available to distinguish single buildings within the models. Typical file formats are, for example, 3DS, COLLADA/KML, X3D, OBJ.

CityGML building models provide the most complete description of geometric, semantic, and thematic aspects of buildings (Section 2.3.1). As for architectural models, we extract footprint and height data for further processing.

Infrastructure Elements

In a 3DCM, infrastructure elements I represent roads, waterways, and railroads that are used to visualize navigable routes or to compute a graph structure for shortest path queries. As major structuring elements of cities they are incorporated by the cell-based generalization technique. Infrastructure elements are represented by piecewise linear 2D curves geometry.

The different importance of individual roads is reflected by classes defined for road features in typical datasets. As it is a general concept, road classes can be found in all kinds of data sources, e.g., volunteered geographic information (Goodchild, 2007), as in OpenStreetMap, or offered by commercial data providers, such as Teleatlas or Navteq. We distinguish six weight classes with decreasing importance $w_5, ..., w_0$ for infrastructure elements and map input road classes to them, depending on the data provider. As the data providers have different definitions for their road classes, there are several possible mappings. The only requirement for a mapping to provide sensible results is that the overall road density in the processed area decreases with growing road weight. We suggest the mapping given in Table 3.1. Other infrastructure elements such as waterways and railroads are mapped to the highest weight.

Non-Building Elements

In addition, non-building areas N such as land use and land coverage data can be added to integrate green spaces, lakes, or wooded areas. A classification is not used by cell-based generalization itself. However, if present, classes can be used later to control the appearance styling for the visualization (Section 3.8).

3.4.2 Generalized Building Representation

As a result of the generalization process, generalized buildings are stored as generalization items $g \in G$. A generalization item represents one or multiple buildings with their generalized geometric and meta data attributes. It stores the current LOA identifier and references another generalization item using a *generalizesToID* to explicitly track the generalization relation. Multiple generalization items may have the same *generalizesToID* as they might be aggregated to a single shape.

Aggregated building blocks are expressed as cell blocks with accumulated statistics about the contained buildings. Initially, all input buildings are expressed as single buildings that refer to their original building items. Optionally, a building may have a weight higher than one to mark it as more important then other buildings. This allows for manual selection as a landmark building.

3.5 Implementing Cell-based Generalization

3.5.1 Creating 2D Settlement Cells

First, the set of line string geometries representing infrastructure and outlines of non-building elements are used to derive a subdivision into potential building parcels by computing the geometric *arrangement* $\mathcal{A}$ (Berg et al., 2000). A geometric arrangement, in general, is the decomposition of a finite collection of algebraic surfaces $\mathcal{S}$ defined in $\mathbb{R}^d$ into cells. Depending on their number of dimensions ($\leq d$), these cells are called vertices ($d = 0$), edges ($d = 1$), or faces ($d > 1$).

As we are interested in a decomposition of the two dimensional reference plane of a 3CDM, we can restrict the arrangement to two dimensions. Also, we only need to handle bounded geometry representing real world objects, i.e., no unbounded lines. Therefore, the collection of algebraic surfaces is a finite set of bounded lines, or piecewise linear bounded curves, L. We compute the arrangement $\mathcal{A}(L)$, resulting in vertices, edges, and facets having the following properties (Figure 3.8):

- The facets cover the whole space $\mathbb{R}^2$, forming a subdivision of the plane. This includes bounded facets $f_1 ... f_n$ and one unbounded facet f_0 which encloses everything else.
- The arrangement, in general, may have isolated vertices. As we restrict input to line strings, in our case, these do not occur.
- The arrangement may have dangling edges, i.e., edges that share the same facet on both sides, for example in case of dead ends.

The construction of the arrangement can be done using a plane sweep approach or by incremental insertion and intersection of curves. Internally, the result is stored as a doubly-connected edge list (DCEL, also half-edge data structure) that allows facets, edges, and vertices to be extracted easily. We further process the facets $f_1 ... f_n$, excluding dangling edges.

The next step is to cut out the space representing infrastructure and non-building areas using Boolean difference operations. All infrastructure curve features need to be converted to polygons before using them in the Boolean difference computation. Polygons can be obtained from curves by applying morphological operations as defined by mathematical morphology (Serra, 1982). For the task of buffering planar curves, morphological *dilation* can be applied, which is defined as a basic operation on two sets $A \subset \mathbb{R}^2$ and $B \subset \mathbb{R}^2$ (Figure 3.9):

$$A \oplus B = \bigcup_{b \in B} A^b, \quad \text{with } A^b = \{a + b | a \in A\}.$$

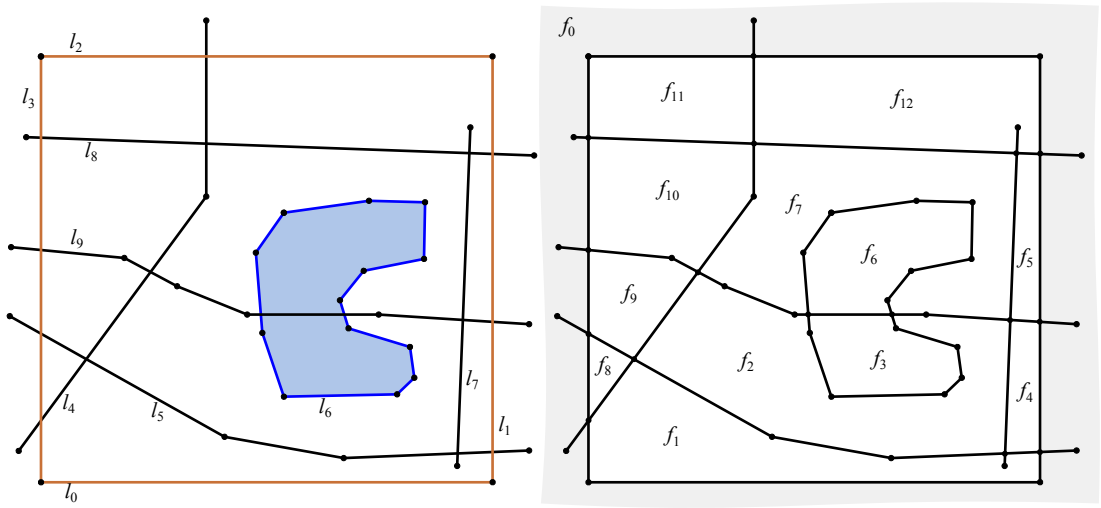


Figure 3.8: Input geometry line strings l_i consisting of infrastructure geometry, contours of non-building features, and the contour of a bounding rectangle ($l_0, ...l_3$) are used to compute a subdivision of the plane using geometric arrangement (left). The results are one unbounded facet f_0 and a number of bounded facets $f_i, i > 0$ (right).

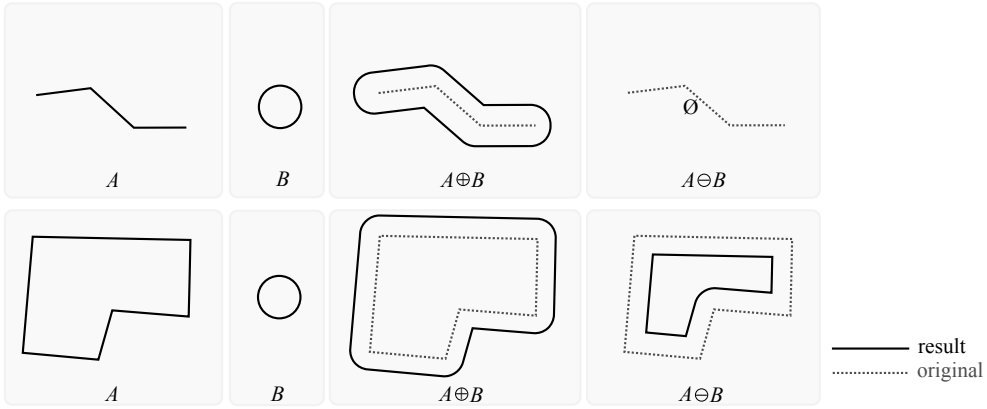


Figure 3.9: Dilation and and erosion by a circular shape is applied to a line string (upper row) and a polygon (lower row). The original shapes' edges are shifted outside or inside, respectively, and the rounded corners are introduced at convex or concave parts. Eroding a line string eliminates it.

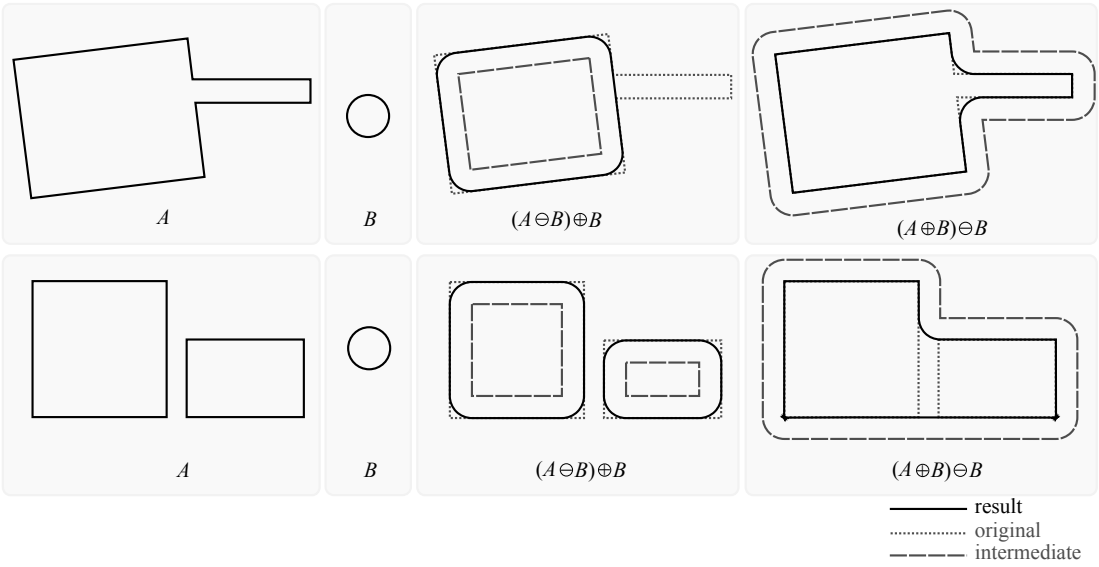


Figure 3.10: Opening and closing by a circular shape is applied to different shapes. While maintaining the general extents of the original shapes A, opening eliminates narrow parts (center) and closing connects narrow gaps between input shapes.

We say that A is dilated by B, and the result is the union of all sets derived by translating A by each vector $b \in B$. Intuitively, this can be obtained by sliding B, commonly called the structuring element, along the boundary of A and add all space covered to A. Dilating a line or a polygon with a simple shape, e.g., a circle or a box, can be implemented straightforward by shifting each edge in direction of its normal. The shifted vertices are then connected according to the shape of B, e.g., with an arc (circular shape).

To construct a road polygon with a certain width, we need to compute the dilation of the road's curve with a circle centered at the origin having $\mathrm{radius} = \mathrm{width}/2$. For different styles with respect to different road classes, varying radii can be used. Also, non-building polygons are buffered with a small constant radius. All polygons resultant from the buffering are finally subtracted from the facets, yielding a set of new facets $f'_1 ... f'_m$.

In the last step, very thin facets that can occur are removed using the composed operation of morphological opening. First, we need to introduce the basic operation of *erosion*, which is defined analog to dilation as

$$A \ominus B = \bigcap_{b \in B} A^{-b}, \quad \text{with } A^{-b} = \{a - b | a \in A\}.$$

Hence, the result is the intersection of all sets derived by translating A by each inverse vector $-b \in B$. An intuitive understanding would be to slide B along the boundary of A, this time removing everything covered by B from A.

Now we can give the definition of *opening* of A by B as an erosion followed by a dilation of A with B (Figure 3.10):

$$A \circ B = (A \ominus B) \oplus B.$$

The erosion removes narrow parts of A, while the subsequent dilation restores the general extents of the shape. As a property, the number of facets can be the same or more after the opening.

For completeness, morphological *closing* is given as a dilation, followed by an erosion:

$$A \bullet B = (A \oplus B) \ominus B.$$

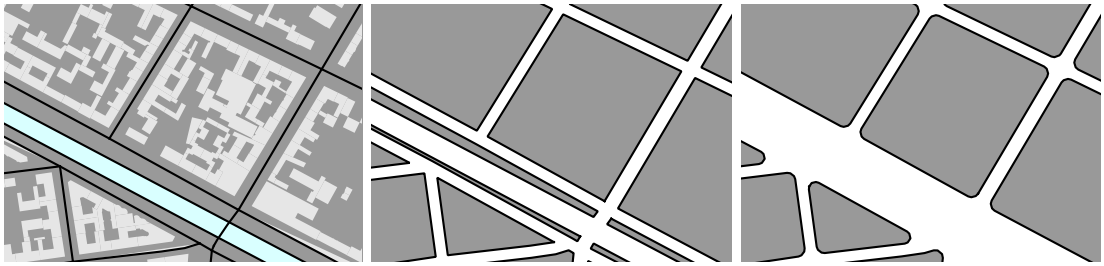


Figure 3.11: Line string geometry from the infrastructure network and non-building outlines (superimposed) are used to partition space into facets $\{f_1, ..., f_n\}$ using geometric arrangement (left). The buffered line string geometry is subtracted, yielding a new set of facets $\{f'_1, ..., f'_m\}$ (center). Morphological opening removes thin facets and creates the final cell polygons $\{g_1, ..., g_k\}$ (right).

Finally, we can remove thin facets by applying morphological opening with a circular shaped structuring element having radius r. Parts of the facets with a local diameter of $\leq 2r$ are eliminated. So far, the results are 2D cell polygons $g_1 ... g_l$ similar to 2D settlement areas (Figure 3.11).

3.5.2 Aggregating Building Features

To characterize the 2D cell polygons, we need to evaluate the contained buildings and aggregate building-individual data. Building-individual data may be geometric, visual, and semantic attributes that can be obtained from the building model, i.e., shape measures such as orthogonality, height, or average color; or provided by additional sources, i.e., community websites such as Wikipedia or Flickr.

We use building height h_i and area a_i, as geometric attributes with major importance, to characterize building i. After computing individual attributes, we set up a candidate mapping of buildings to cell polygons based on the geometric containment relation. All buildings contained in one cell polygon could potentially be generalized by this polygon. This mapping is stored by setting a unique *ID* value and a *generalizesToID* value referencing the containing cell.

As the final step, mean, variance, and standard deviation is computed for all buildings within one cell. For aggregation, we compute the aggregated height of a cell as the mean height $\bar{h}$ weighted by the area of individual buildings

$$\bar{h} = \frac{\sum h_i \cdot a_i}{\sum a_i}.$$

The mean represents the aggregated result of the individual height attributes, while the standard deviation is a measure for the homogeneity within the aggregated buildings. The standard deviation has the same dimension and unit as the individual attribute. Therefore, we can use it to identify outliers by comparing the attributes of individual buildings with the mean and relate the difference to the standard deviation. We will use this later to identify initial landmark buildings (Section 3.7).

This concept can be extended to arbitrary attributes describing relevant building properties. To do this, individual attributes are computed and stored in an attribute vector that has a component for each of the attributes. In addition, a distance metric and an aggregation function has to be provided for each attribute. For example, using a predominant color as an attribute could require a more complex distance metric and aggregation based on human perception. An aggregated attribute vector would then be stored along with the generalization feature.

3.6 Multiple Levels of Abstraction

While the cell blocks already represent abstracted representations, the workflow can be extended to create additional LOAs at smaller scales (Figure 3.12). Different LOAs may be needed for applications that rely on visualizing phenomena observed at smaller scales. Also, in an interactive visualization, the user can zoom out to switch to smaller scales. Hence, the initial problem of cluttered visualization originating from many small objects re-occurs.

Our solution is based on the observation that depending on the scale, maps do not include certain road classes anymore, but only major ones (Brewer and Buttenfield, 2007). Consequently, the blocks resulting from the cell-based generalization technique will become larger. Also, the road classes serve to model the importance of roads in reality; we can exploit this to create cells with meaningful granularity and shape.

We associate the number of infrastructure weight classes relevant for generalization with the LOA representations. If we distinguish $n+1$ infrastructure weight classes in decreasing order $w_n, ...w_0$, we can create $\mathrm{LOA}^0, ..., \mathrm{LOA}^{n+2}$, where LOA^0 is identified with the input city model, LOA^1 represents the city model generalized using all available infrastructure features, LOA^{n+1} represents the city model generalized using just infrastructure features having w_n, and LOA^{n+2} represents the whole city area with no infrastructure.

The workflow needs to be adapted at the following points:

- Infrastructure features have to be filtered to only incorporate the most important ones.
- Statistical operations have to operate on the current LOAs as input instead of building features.

This is reflected by an adapted definition of $\mathrm{cellGen}$ that works with generalization features which have been created before, and uses a filtered subset of infrastructure features:

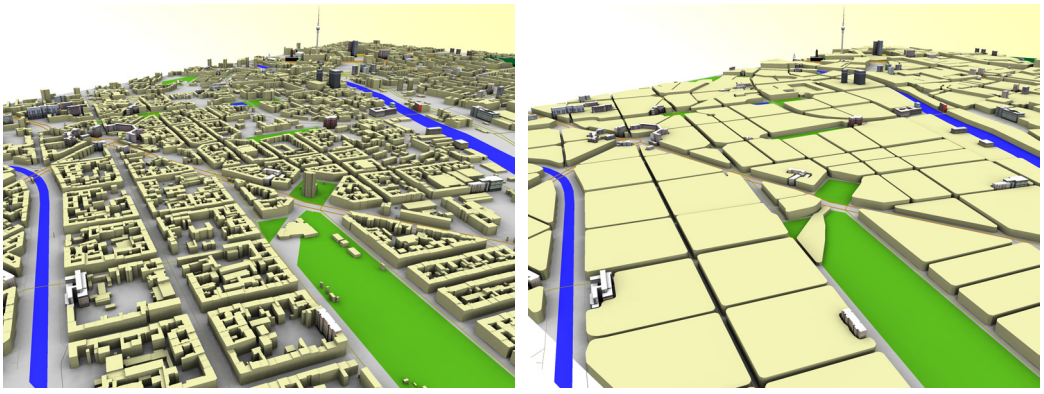
$$G^{j+1} = \mathrm{cellGen}(G^j, I^{w_j}, N)$$

with $I^{w_j} := \{i \in I | \mathrm{weight}(i) \geq w_j\}$ to filter out the most important infrastructure features. Iteratively applying $\mathrm{cellGen}$ yields multiple sets of generalization features with growing abstraction. In the process the result of the previous iteration is used as input for the current iteration (Figure 3.12).

However, the approach so far can be improved, as expensive geometric operations such as buffering and Boolean subtraction are executed redundantly. As an improvement, we can split $\mathrm{cellGen}$ into its geometric and its mapping aspect. In the adapted work-flow, we refine geometry iteratively, starting with the whole city area, and add infrastructure features having the highest class in each iteration (top-down). Each infrastructure feature has only to be buffered and subtracted once. Then, starting with building models, we iteratively do the mapping to the next geometric representation created before, and compute and store aggregated attributes (bottom-up). The result are again multiple sets of generalization features, but the number of expensive operations are reduced.

3.7 Identifying and Integrating Landmark Buildings

The visualization of cell blocks can be enhanced by complementing them with landmarks. In a spatial environment, landmarks are essential elements for navigation and orientation. They represent distinctive objects having key characteristics that cause them to stand out from their surroundings (Sorrows and Hirtle, 1999). In 3D virtual environments landmarks are especially important since, compared to real environments, fewer spatial and locomotive cues are presented



(a) LOA^0 (b) LOA^1

(c) LOA^3 (d) LOA^4

Figure 3.12: For increasingly abstract representations, the algorithm creates increasingly large block cells with decreasing numbers of landmark buildings (small Berlin dataset). The definitive version of these images can be found at (Glander and Döllner, 2009), ©Elsevier.

to users as they move through the environment. Explicitly designing landmarks in virtual 3D environments such as 3DCMs therefore can facilitate navigation and the acquisition of spatial knowledge (Vinson, 1999).

We base the landmark selection on (Winter et al., 2008), where landmarks are discussed in the different contexts of way finding and spatial knowledge. The authors suggest that *local landmarks* and different levels of *global landmarks* can be differentiated by the size of their reference region. Within a landmark's reference region, one would rather refer to that specific landmark than to any other, e.g., when verbally describing a route.

This section describes the detection and integration of local landmark buildings based on their attributes and the creation of a landmark hierarchy to identify global landmark buildings on different hierarchy levels.

3.7.1 Identifying Local Landmark Buildings

To identify landmark buildings, we need to compare individual buildings to their neighboring buildings with respect to their properties. We require these properties to be expressed by attributes of building features, and define neighborhood as all building features within the same generalization cell.

Having the average height $\bar{h}$ and the standard deviation σ computed for each generalization cell as part of the aggregation done in Section 3.5.2, we use distance between individual and average height as an indicator for outliers. Buildings that deviate significantly from the height of their neighbors are identified as local landmarks and put into the set of initial landmarks. We describe "significantly" in terms of multiples k of the standard deviation σ, typically with $k = 1.5$.

$$\mathrm{isLandmark}(b_i) = \mathrm{dist}^h(h_i, \bar{h}) > k \cdot \sigma \tag{3.1}$$

Additional buildings may be selected manually for the initial set of landmarks, for example, points of interest such as churches. A generalization to arbitrary attributes can be obtained by redefining the function to compare a vector of attributes to the vector of aggregated attributes of the generalization feature.

3.7.2 Integrating Landmark buildings

Building features which are identified as local landmarks have to be removed geometrically from the containing cells by cutting them out. Similarly, as they no longer contribute to the block cell, their attributes are removed from the aggregated value stored with the cell. It is useful for some aggregated attributes to store intermediate values for efficient removal. For example, to adapt the mean height after removing a landmark building b_j, it is useful to store numerator N and denominator D as separated values to speed up re-computation

$$\bar{h'} = \frac{N - h_j \cdot a_j}{D - a_j} \text{ with } N = \sum h_i \cdot a_i, D = \sum a_i.$$

The landmark buildings are then included geometrically unchanged into the LOA – that is, no simplification is applied to them. This facilitates their identification by the users and also makes them stand out visually from the homogeneous and uniformly colored cell blocks (Figure 3.13).

3.7.3 Creating a Landmark Hierarchy

For higher LOA representations, we use a different technique to identify landmarks. The goal is to reduce the number of landmarks, while keeping important ones and maintaining an even spatial distribution. For example, a newspaper shop may be used as a landmark in the surrounding

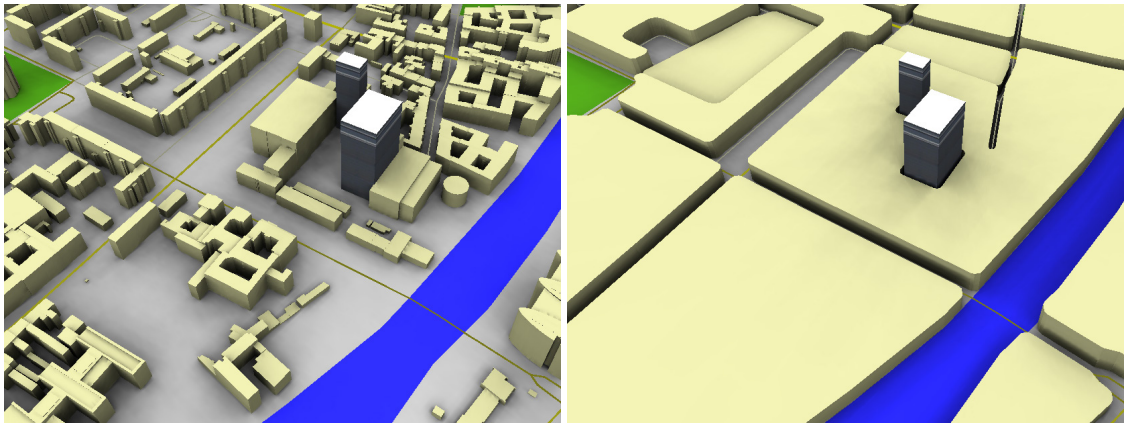


Figure 3.13: Landmark buildings are integrated into the scene by cutting out their buffered footprints from the surrounding cell(s). The landmark buildings with their original geometry and textures are then placed at their positions.

neighborhood, but is rather unimportant compared to the whole city. Visualizing it would be essential in a large scale context, but regarded as noise in small scale context.

Simply choosing a higher deviation factor k in Equation 3.1 does not lead to a desirable landmark selection, as k cannot be determined automatically for different data sets; nor does it yield a balanced distribution. Instead, an algorithm adapted from Winter et al. (2008) creates a balanced landmark hierarchy as follows:

- Triangulate landmarks using their centroids for a Delaunay Triangulation (DT) (Berg et al., 2000).
- Compare each landmark's saliency with direct neighbors in the DT and vote for the highest saliency landmark among them.
- Promote landmarks that have at least one vote to the next level.
- Iterate until only one landmark is left.

The concept of saliency includes visual, semantic, and structural properties (Sorrows and Hirtle, 1999). However, computing a complex saliency measure for a building weight is beyond the scope of this thesis. In our proof of concept we use height as a landmark weight. Direct neighborhood is defined topologically as the landmarks that are connected with an edge of the DT.

As an example, consider the situation depicted in Figure 3.14: The direct neighborhood of landmark l_7^0 at the first level of the hierarchy 0 is $L_7^0 = \{l_2^0, l_3^0, l_6^0\}$. The landmark with the maximal weight in the neighborhood of l_7^0 is itself (3), and receives one vote. Overall, l_1^0, l_5^0 and l_7^0 receive votes and get promoted to the next hierarchy level 1, where the algorithm is repeated.

In the resulting landmark hierarchy, the number of landmark buildings is steadily reduced in subsequent layers of the hierarchy. The algorithm stops when no more reduction occurs. The landmark hierarchy is then aligned with the previously created block cells (Figure 3.15): the inital set of landmarks L^0 is integrated into LOA^1, LOA^2 uses L^1, and so on. The hierarchies generally have different depths, so there might be LOAs that cannot be mapped to a hierarchy level of landmarks, as well as levels of the landmark hierarchy that are not being mapped to.

3.8 Scene Composition

To do the final rendering, we construct a scene containing the styled geometry representations.

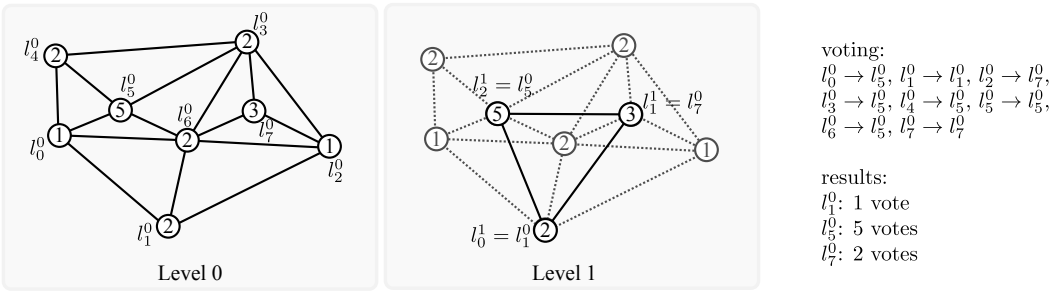


Figure 3.14: The example shows one iteration of the construction of the landmark hierarchy. Weighted landmarks l_i are triangulated in a Delaunay Triangulation (left). For each landmark's topological environment, the highest weighted landmark is chosen for the next level. Landmarks receiving at least one vote are promoted to the next hierarchy level (center), where they are triangulated again. This is repeated until only one landmark is left.

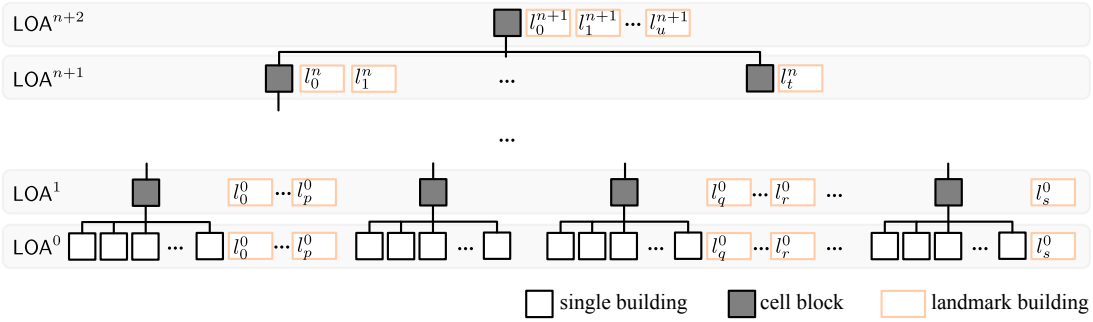
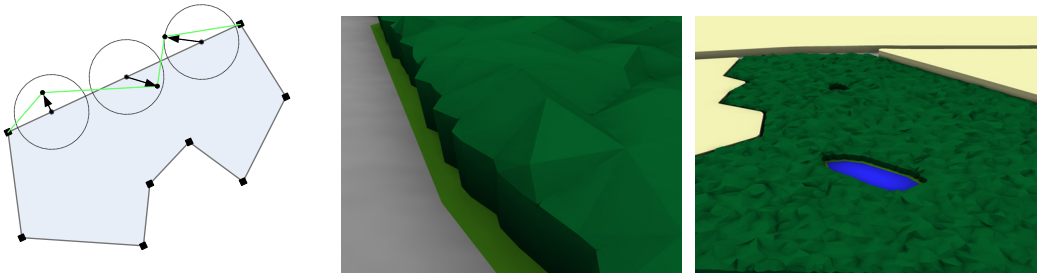


Figure 3.15: The hierarchy defined by the cell generalization is aligned with the hierarchy defined by the landmark hierarchy. Original, single buildings at LOA^0 are generalized to cell blocks, while local landmarks l_i^0 within one cell are detected and included unchanged in LOA^1. The number of LOAs depends on the number of different infrastructure weight classes n. In general, there will be more (as depicted in the schema) or less levels in the landmark hierarchy, than created during the cell generalization.



(a) Polygon manipulation (b) Wooded area close-up (c) Wooded area total

Figure 3.16: Abstract representations of wooded areas are created by adding and moving additional points at the polygon's edges (a) and extruded surface (b). Contained water areas are cut out of the polygon (c). The definitive version of these images can be found at (Glander and Döllner, 2009), ©Elsevier.

3.8.1 Creating Building Representations

We have two types of building representations: high detail 3D geometry stored for single landmark buildings and 2.5D cell blocks. Whilst the former is directly integrated into the scene, the latter requires the creation of 3D geometry by extruding their polygonal footprints. The extrusion shape consists of wall geometry and planar roof geometry.

To create the roof geometry, the polygon is lifted to the computed height and tessellated into triangles. To create the wall geometry, each pair of vertices of the footprint polygon (v_i, v_{i+}), with v_{i+} defined cyclically, together with another pair of vertices with lifted height (v'_i, v'_{i+}) forms a piece of the wall: it consists of two triangles (v_i, v_{i+}, v'_{i+}) and (v_i, v'_{i+}, v'_i). Assuming a counter-clockwise definition of the outer loop and a clockwise definition of the inner loops, the created triangles will be consistently facing outwards.

3.8.2 Handling Non-Building Elements

Non-building elements may include green spaces, wooded areas, and water areas. Their polygons are tessellated to triangles, colored accordingly and added to the scene.

For wooded areas, we create three dimensional representations to communicate the fact that they effectively occlude objects behind them. In contrast to photo-realistic visualization, which require detailed vegetation models, we need an abstract representation. In 2D maps, signatures are used to abstract from specific structures and, e.g., distinguish between built-up areas, green spaces, and water areas (MacEachren, 1995). We construct an abstract 3D representation by inserting additional points along polygon edges. The points are moved randomly within a radius to create a fuzzy shape (Figure 3.16). After extruding the polygon to typical tree heights, the top surface is also perturbed by adding slightly randomized points to the surface.

3.8.3 Visualizing the Road Network

An important aspect of the generalized 3DCM is the road network as it represents navigable space and structures the city. In addition to cutting roads out of the cell blocks, we explicitly add line geometry representing roads to the scene. For different road classes we use a style set with different widths, colors, and outline colors to enable visual differentiation (Figure 3.17).

In addition, labels are important to communicate individual attributes of depicted shapes (Maass, 2009). They help link the virtual 3D models with the real world and therefore serve as spatial references. For showing the street names we construct static, curved labels which are aligned with the local course of the road in a straightforward way (Figure 3.18, left): the midpoint of one road segment is used as the center for the label. For each letter we determine the position on the line

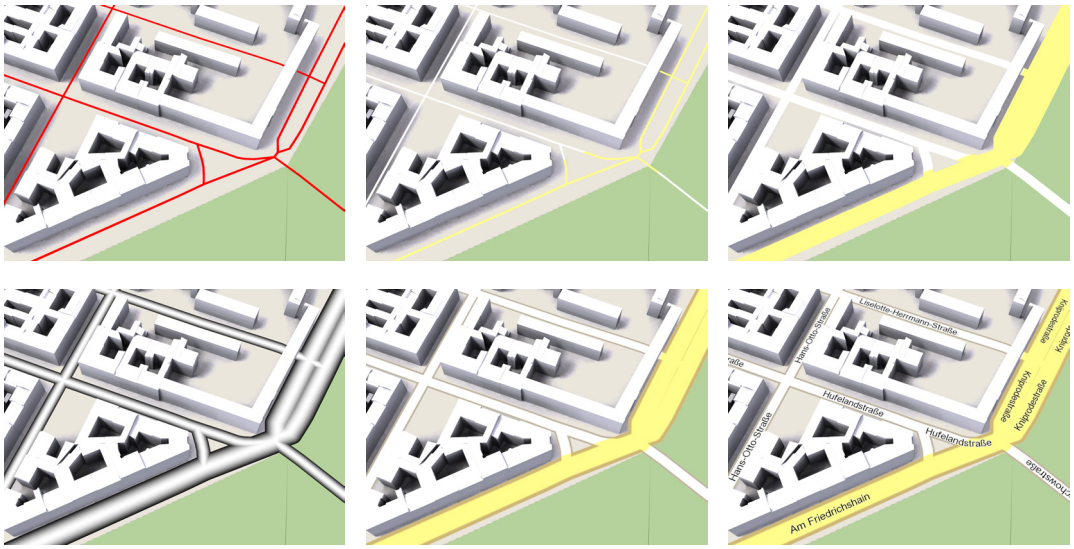
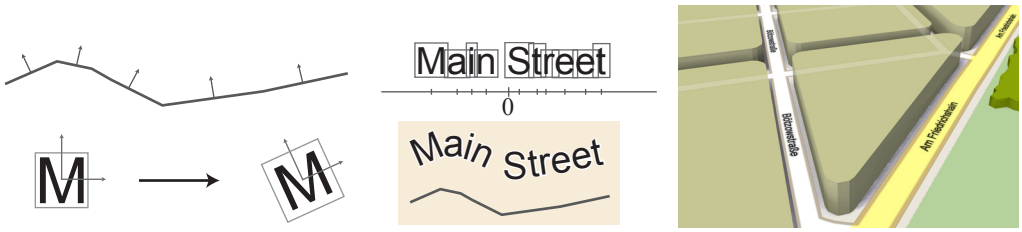


Figure 3.17: Upper row, left to right: standard line rendering has limited styling potential. Using the different classes, different colors and widths are applied. Lower row, left to right: Using a distance field, outline colors are added and road names are shown as curve following labels.



(a) Label construction (b) Styled labels

Figure 3.18: Each letter is rotated and translated according to its position on the line string, finally forming the curved label(left). The labels are rendered on top of the road (right).

string and the normal direction. The outlined letter is translated, rotated, and placed in the scene, to form the curved road label.

Rendering road geometry and the terrain generally leads to z-fighting issues because of coplanar geometry. To avoid this and occlusion of the roads by building geometry, we apply image composition. In a pre-processing pass the road network is rendered into an off-screen buffer using the same camera settings. When rendering the regular scene, the buffer containing the styled roads is available and blended with the scene rendering. This enables blending roads half-transparently with building and cell block geometry, thus improving their visibility (Figure 3.18, right).

3.9 Results and Application Examples

The presented technique has been developed in C++, building on external software libraries. The computational geometry operations are done with *Geometry Engine Open Source* (GEOS)[1] and 2D geometry is stored in a spatial database. For it, we created a library, *CGSDB*, which implements the OpenGIS Simple Feature model (Herring, 2011) while abstracting from the specific underlying spatial database management system. Currently we use *Spatialite*[2].

[1] `www.geos.osgeo.org` (accessed 1.7.2012)
[2] `www.gaia-gis.it/spatialite`, (accessed 1.7.2012)

	# buildings (app.)	# streets (app.)
Berlin	65 000	13 000
Berlin (small)	14 000	1 700
Boston	150 000	17 000
Cologne	150 000	6 000

Table 3.2: Comparison of the four datasets used.

For the 3D scene composition and visualization we have implemented applications using the scene graph-based development kits *Virtual Rendering System* (VRS) and *Open Scene Graph* (OSG) as rendering middleware.

In the following, we present results of our processing technique and visualization by applying it to different data sets. Also we explore smooth transitions between LOAs and present applications of cell-based generalization in a focus+context scenario and for cognition science.

3.9.1 Application on Different Data Sets

We applied the cell-based generalization technique on the 3DCMs of Berlin, Boston and Cologne for evaluation. The models of Boston and Cologne contain simple 2.5D building models. Berlin is modeled in CityGML and contains LOD 1-4 building models, partly with façade textures (Table 3.2). Data for road network and non-building areas is based on OpenStreetMap (OSM).

Evaluating the results for these datasets shows that cell-based generalization generally yields representations appropriate for an abstract visualization (Figure 3.19). However, the assumption that individual buildings can be substituted by a cell block built by surrounding streets fails for sparsely built areas, e.g., in rural areas (Cologne) and harbor areas (Boston). If these areas do not contain explicitly specified non-building areas, the resulting representation is misleading. One can observe that block cells are placed in sparsely built areas without necessarily having buildings there. This is a consequence of creating cells based on the road network and non-building areas only. However, the general appearance of a 3DCM is captured by the generalized 3DCM. Using average height to determine the extent of extrusion for abstracted blocks allows, e.g., to distinguish downtown areas from suburban areas.

Since the generalization process is based on the road network, a natural hierarchy of abstract blocks can be created using road weights (Figure 3.12). The derived blocks are increasingly abstract and represent the city in different scales.

Integrating landmark building models enhances the visualization and gives additional orientation marks. However, the detection of landmark buildings based on height differences compared to the block cells only, does not lead to a convincing selection of landmarks in all situations. Inclusion of additional visual and semantic criteria, and comparison to other environments is desirable, e.g., by comparing with all buildings at a road junction or in spatial proximity. Also, so far only buildings are considered, while the technique could be extended to other point based landmarks, e.g., monuments and prominent trees.

The presented results are static, abstract representations of 3DCMs that can be explored interactively. They fulfill the goal to reduce complexity of a given 3DCM while maintaining landmark buildings and important structural relations necessary for orientation and navigation.

3.9.2 Generalization Lenses

We use the computed LOA representations in a focus+context scenario by applying *generalization lenses* (Trapp et al., 2008). This addresses demands of local modification of the abstraction, as required, e.g., for highlighting a specific route or a region of interest (Figure 3.20). The lens

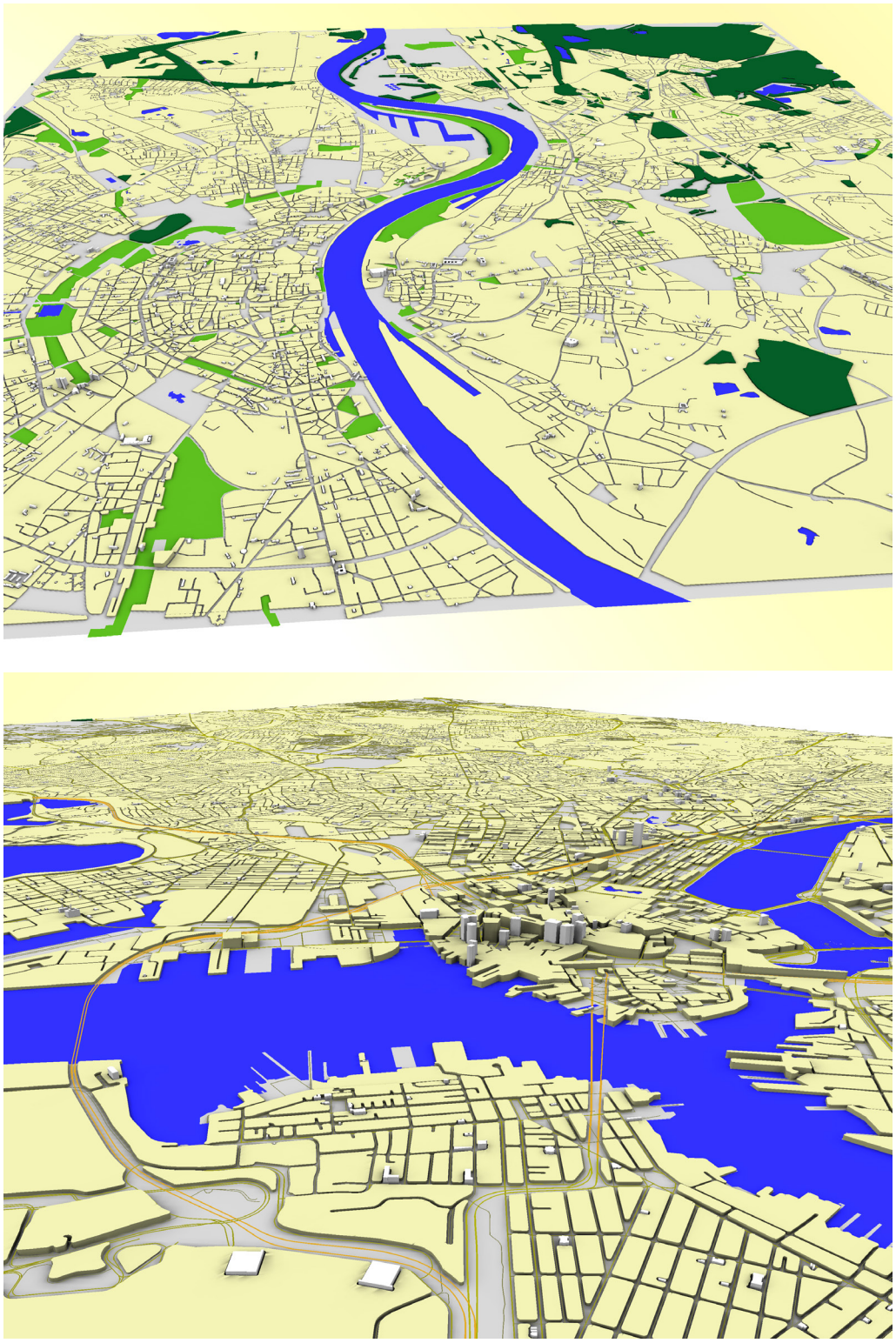


Figure 3.19: The technique was tested on city models of Cologne (top) and Boston (bottom). The definitive version of these images can be found at (Glander and Döllner, 2009), ©Elsevier.

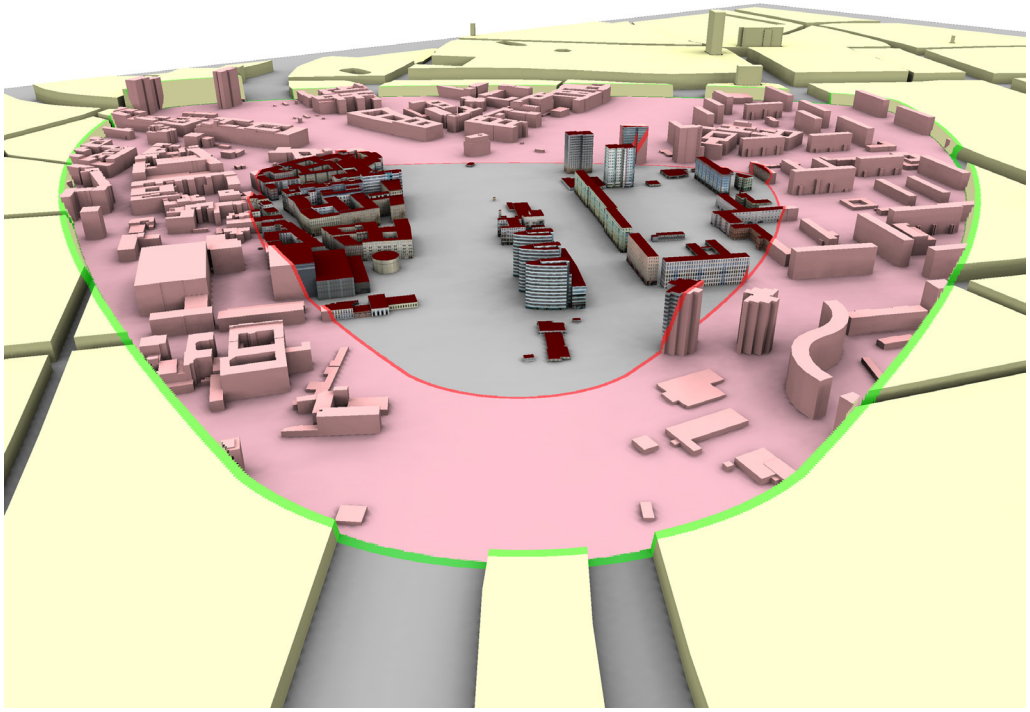


Figure 3.20: Generalization lenses can be applied relative to the virtual camera. Whereever the user moves the virtual camera, the current view reveals the highest detail in the center while showing more abstract representations in the peripheral area.

metaphor serves to explain and control the focus+context visualization by separating parts inside and outside the lens. Magic lenses (Bier et al., 1993) reveal hidden structures without necessarily magnifying the content, and can also be extended to 3D (Viega et al., 1996). We implemented a visualization technique based on volumetric lenses to integrate several LOA representations in one image. Hence, the high detail 3DCM including textured buildings can be explored along with the abstract block cells.

Lens Creation

Arbitrary 3D lens shapes are first converted into a voxel representation and stored in layered depth images called *volumetric depth sprites* (VDS) for efficient rendering (Trapp and Döllner, 2008). Lens shapes can either be modeled explicitly using 3D modeling software, or derived from buffered 2D polygons and polylines. By applying a transformation to VDS at runtime, interactive rotation, scaling, and translation of lenses is possible. To support multiple, possibly overlapping lenses, each lens is mapped to a LOA representation and assigned a priority.

Lens Rendering

During rendering, pixel-precise clipping is done within a single rendering pass to determine, which LOA representation a fragment belongs to, to draw it accordingly. The algorithm starts by rendering the context geometry, which is the geometry associated to the space surrounding all lenses, clipped against all other lens volumes. Then, starting with the lowest priority, LOA geometry is rendered successively within its respective lens volume, thereby clipping it against all lens volumes with a higher priority. This is done until all lenses are rendered.

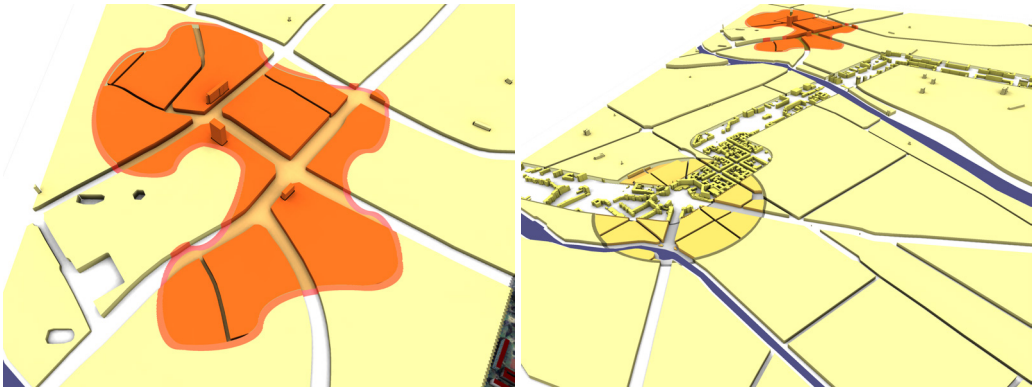


Figure 3.21: Generalization lenses allow the combination of different LOA representations in one image. Therefore, different areas of interest to the user can be highlighed as focus regions. Lenses of focus regions may have arbitrary shape (left) and overlap themselves. An example is the highlighted course of a route that crosses a highlighted plaza (right). The definitive version of these images can be found at (Glander and Döllner, 2009), ©Elsevier.

Results

The lens technique allows for the flexible combination of different LOA representations within an efficient rendering framework. Use cases include the highlighting of focus regions and routes within 3DCMs (Figure 3.21). Since lens volumes can be moved in real-time, dynamic regions of interests, e.g., the area around the current position of a car driver, can also be highlighted. One limitation of the approach is due to the limited resolution of the depth images used to represent lens volumes: depending on the chosen resolution, staircase artifacts can occur at the lens edges.

This application example demonstrates the interactive visualization of abstract representations in a complementary way to the multi-scale rendering technique presented before.

3.9.3 Multi-Scale Rendering with Smooth Transitions

When 3DCMs are visualized, a perspective projection is typically applied during rendering (Akenine-Möller et al., 2008). Thus, a view matching the real world experience is created: objects far away are perceived as small and grow larger as the viewer approaches them. This represents a continuous scale interval in contrast to the view presented by classical maps, which typically show a top down view on single scale.

To provide multi-scale visualization, we render multiple LOAs based on the Euclidean distance from the virtual camera. For smooth visualization, we propose the use of transition mechanisms based on vertical motion and transparency blending.

Validity Ranges

With multiple representations available, one needs to decide which representation should be shown for a certain scale. We propose to use distance with respect to a reference distance defined given generally by the virtual camera distance. Other reference distances are also possible, e.g., adapt scale relative to a given focus center.

Validity Ranges describe, in which distance interval objects of a certain LOA^i should be rendered. This is defined as a tuple of distances to the virtual camera $(d^i_{\text{start}}, d^i_{\text{fullStart}}, d^i_{\text{fullEnd}}, d^i_{\text{end}})$. While the interval $[d^i_{\text{start}} : d^i_{\text{end}}]$ describes when a LOA representation becomes visible and invisible, respectively, the interval $[d^i_{\text{fullStart}} : d^i_{\text{fullEnd}}]$ specifies the range within which it is completely visible (Figure 3.22). A non-overlapping partition of the view axis into distance intervals $[d^i_{\text{fullStart}}, d_{\text{fullEnd}}]$ would imply hard switches of LOA representations. Hence, overlapping

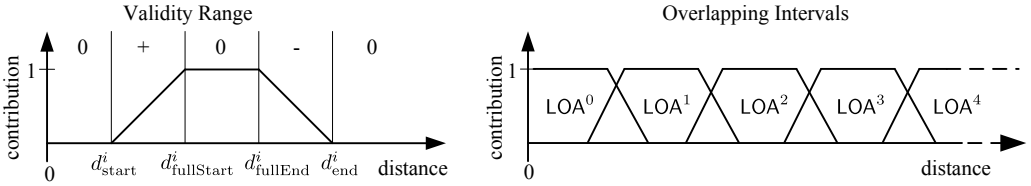


Figure 3.22: For each LOA representation, a validity range is set up by four distances (left). Overlapping intervals enable smooth transitions (right). The definitive version of these images can be found at (Glander and Döllner, 2009), ©Elsevier.

intervals are necessary to produce a visual transition effect.

Interval boundaries can be specified individually for each LOA representation or can be set automatically, e.g., according to a linear or logarithmic mapping. Based on the interval boundaries, a function $\text{contribution}^i(d)$ defines for a LOA^i and a given camera distance d a normalized contribution factor and a change direction $\text{contribution}^i(d) : \mathbb{R}^+ \mapsto (0 \leq x \leq 1, \{+, -, 0\})$. The contribution factor and change direction is computed as follows:

$$\text{contribution}^i(d) = \begin{cases} \left(\frac{d - d^i_{\text{start}}}{d^i_{\text{fullStart}} - d^i_{\text{start}}}, +\right) & \text{if } d^i_{\text{start}} \leq d < d^i_{\text{fullStart}} \\ (1, 0) & \text{if } d^i_{\text{fullStart}} \leq d < d^i_{\text{fullEnd}} \\ \left(\frac{d - d^i_{\text{fullEnd}}}{d^i_{\text{end}} - d^i_{\text{fullEnd}}}, -\right) & \text{if } d^i_{\text{fullEnd}} \leq d < d^i_{\text{end}} \\ (0, 0) & \text{else} \end{cases}$$

Because of overlapping intervals it is possible that at one position there are multiple geometries at the same time. The transition is performed by blending between these geometries.

Transparency Blending

Having computed contribution factors, a straightforward transition mechanism is to blend different LOA geometries using transparency blending. Transparency blending mixes a source color with a new color based on a blending equation and is typically controlled by an alpha value that specifies the opacity for each color value. For visualizing transitions between LOA representations, we use the calculated contribution factor as the value for alpha. As the contribution factor is determined according to virtual camera distance, a dynamic blending effect is achieved when moving through the 3DCM (Figure 3.23 (a)).

Vertical Movement

An alternative transition effect can be obtained using vertical movement of generalization items. When items become visible, that is, $\text{contribution}^i = (x, +), 0 < x < 1$, they will move into the scene by applying vertical scaling using the contribution value x directly as a scaling factor. When becoming invisible, i.e., $\text{contribution}^i = (x, -), 0 < x < 1$, items are scaled to the height of the subsequent generalization item using the *generalizes-to* relation to achieve smoothness (Figure 3.23 (b)).

Results

The results of smooth transitions are ambivalent: multi-scale rendering of 3DCMs yields a visualization that manages varying information density throughout the depicted scene. Dynamic transitions between LOA representations make navigation within the model smooth and help to

maintain coherence while the visual geometry changes (Figure 3.23 (c)). Therefore, interactive visualization of a 3DCM is enhanced by dynamically adapting information density while exploring it.

It can be argued that transition maps should only be shown during movement, as they merely represent intermediate states (van Kreveld, 2001). Therefore, as an extension, the transition technique can be configured to perform a "snap" of generalization items to show their nearest stationary form once camera movement has stopped.

However, the additional animation introduced by the transitions might be perceived as distracting. Applying geometric transitions such as vertical moving or morphing (Glander et al., 2010a) on block cells is problematic since the transformations do not have a natural equivalent. Transparency blending appears to give the best results. In addition some styling issues are open to be dealt with: precomputed lighting both on the terrain and the block cells, e.g., ambient occlusion (Döllner et al., 2006b), does not work in dynamically changing situations. Instead, dynamic shadows or screen space lighting such as screen space ambient occlusion (Kajalin, 2009) have to be applied to provide the necessary depth cues and enhance contrast.

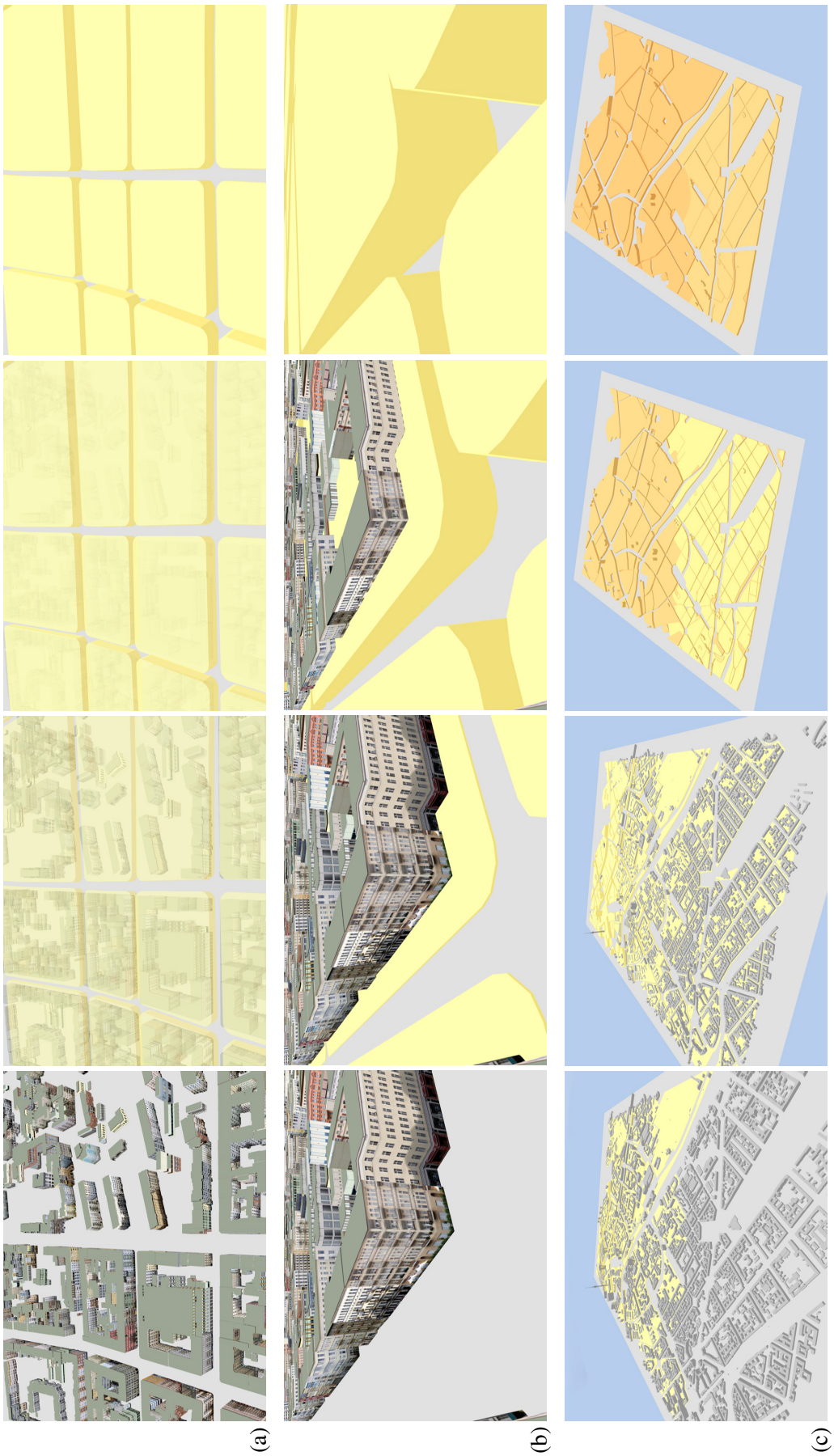


Figure 3.23: A smooth transition between 3DCM representations of different LOAs is accomplished (a) using transparency or (b) by vertically moving geometry in and out the ground. Increasing camera distances are simulated (c) to show the transition effect by vertical movement. The definitive version of these images can be found at (Glander and Döllner, 2009), ©Elsevier.

3.9.4 Application for Cognitive Studies

Orientation and navigation in virtual 3D environments has been an ongoing research question for many years. Research from cognitive science shows that route perception and processing can be improved if its visual model is adapted to correspond with mental concepts of a route. Chorematic or schematic visualization is motivated by findings about human perception and memorization (Brunet, 1993). The suffix *chor-* is greek for "space"; choremes are elementary primitives of space. Cognitively, a route is a sequence of decision points – such as road junctions – which are usually remembered in their prototypical configurations (Klippel et al., 2005). For example, asked for a route description to find the way to some place in a city, most people will give a list of turning actions: "Go straight ahead, then turn sharp left, then right,...". Especially, they will probably not describe the exact turning angles at decision points, but will resort to a few prototypical directions.

An exemplary application of these so called wayfinding choremes has been proposed for 2D maps by altering route junctions (Klippel et al., 2006). We implemented concepts for 3D wayfinding choremes in a 3DCM to be able to evaluate them in user studies (Glander et al., 2009). The cell-based generalization technique is suitable to create the required scene, as the abstract block cells are derived from roads and directly reflect changes in a road's course. In addition, they are sufficient to run an experiment without needing to displace or deform individual buildings along roads.

Choreme Processing

For this application, only a set of roads and a route along them are needed. Starting with the first route node, our technique iterates through all junctions on the route, transforming them: for each junction, the incoming road is set as the reference direction, and all outgoing roads are sorted into eight bins reflecting the prototypical directions in $45°$ steps (eight-sector model). If no unique mapping can be done – that is, more than one road is put into one bin – the junction is left unchanged.

Each outgoing road is then transformed by rotating it to the prototypical direction. New points are inserted within a given radius, and the old connection points are connected using Bézier curves for a smooth connection (Figure 3.24).

The transformed road network is then polygonized and buffered roads are cut out, as described in Section 3.5.1. However, to preserve sharp features, which are needed to communicate configurations of junctions, no morphological closing is applied. To complete the choremized 3DCM, the resulting polygons are extruded to cell blocks, and façade images are applied randomly to create the look of a city.

Results

We developed an application suitable for conducting experiments including joystick interaction, collision handling, position tracking, and monitoring. The solution is applied in psychological experiments evaluating 3D wayfinding choremes and landmark identification theories (Peters et al., 2010).

Abstracted block cells have proven suitable for this, as they allow the creation of a defined environment where unwanted influences, which might interfere with the research question, can be excluded. For example, to evaluate wayfinding choremes, other navigation aids, such as uniquely shaped buildings or significant façades that may serve as landmarks, can be excluded from the visual model. The abstraction in this application example is extended to route decision points – road junctions – which are adapted to their prototypical configuration.

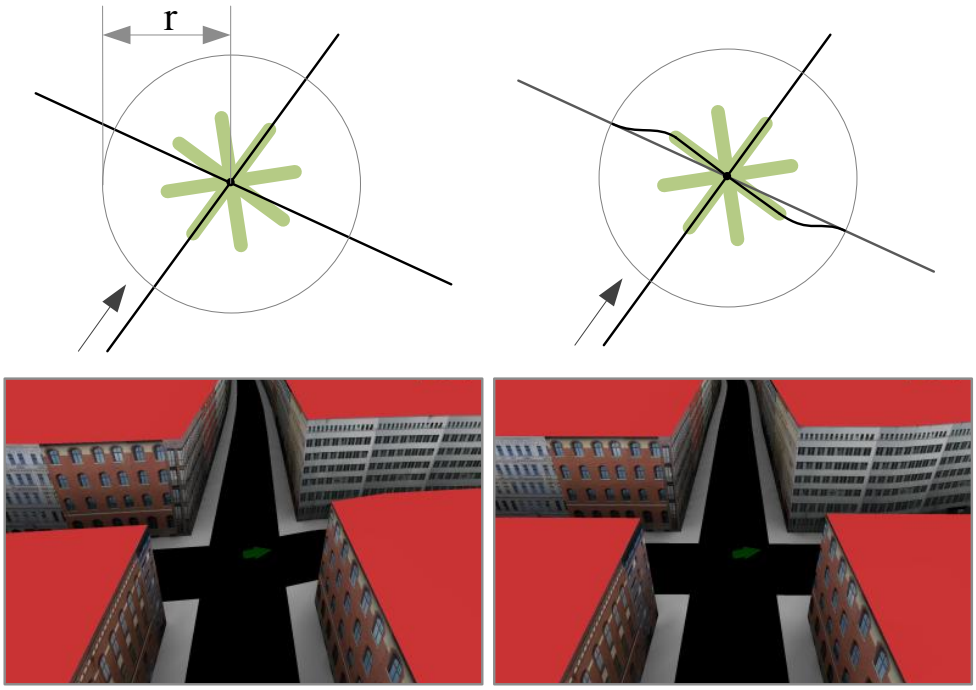


Figure 3.24: Schematic and 3D example of a junction before (left) and after (right) processing of road segments. The radius r can be specified to limit the adaptation effect. The definitive version of these images can be found at (Glander et al., 2009), ©Springer.

3.10 Discussion

The presented automatic 2.5D cell-based generalization technique creates highly abstracted representations of 3DCMs, while enhancing urban structures and landmark buildings. Distinctive geometric abstraction can be obtained on a large area, mimicking the aesthetics of 2D maps such as tourist maps. Also, cell-based generalization is road centric and therefore captures the general structure of the city very well. The capability to create hierarchies aligns well with the human mental model of space, which has been shown to work hierarchically (Tversky, 1992; Hirtle and Jonides, 1985).

Thus, the technique complements previous 3D building generalization techniques, which mainly focus on generalization with respect to small changes of scale. However, for big changes of scale, such as in an interactive visualization of a 3DCM, simplification on the level of a single buildings is hardly noticable. Also, it does not solve the fundamental problem of displaying a huge number of single items. This might be desirable for photo-realistic visualization, as geometric complexity is reduced without sacrifying visual quality. However, simplification and abstraction beyond this require techniques that fundamentally change the representation.

The feasibility of our approach is demonstrated by a number of applications as discussed in Section 3.9. In addition LOA representations have been employed in a number of follow-up works including multi-perspective views (Pasewaldt et al., 2011), a digital kiosk system (Quantz et al., 2011), and immersive 3D virtual environments (Engel et al., 2012).

3.10.1 Limitations and Directions for Future Work

In this section the conceptual and technical limitations of cell-based generalization are discussed. In our approach, being based on infrastructure as the central structuring element, the derived cell blocks are much more imprecise compared to other approaches for building generalization. The

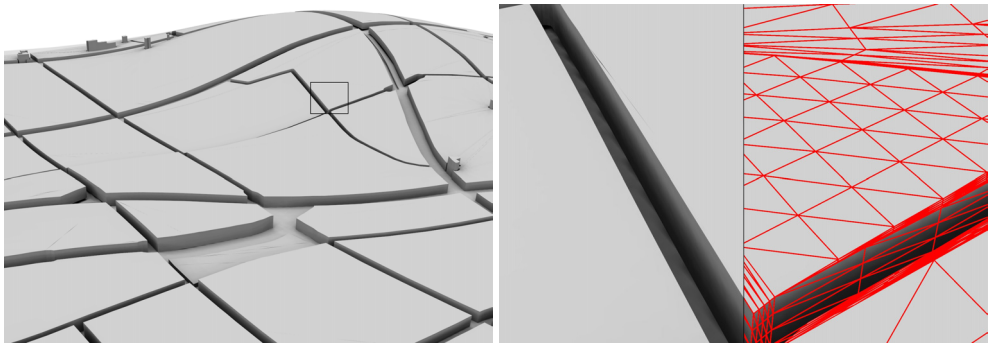


Figure 3.25: 3D curvature of the terrain surface can be incorporated by subdividing the cell geometry into smaller triangles and adding the current terrain height as an offset. The sketch shows an exaggerated elevation model.

availability of road network data in general is not a problem in times of Open Street Map. However, creating generalized building representations from road network cells works well only in densely built areas. In peripheral or rural areas, the assumption that a cell block is a good approximation of contained buildings fails. One direction for future work is to use automatically derived view axes as an approach for finer parcels (Jiang and Liu, 2010). An alternative idea is the automatic detection of free spaces, e.g., through thresholding in distance fields of the building geometry. These could then be integrated into the current processing pipeline as non-building features for subtraction. Also, a quality metric could be integrated to evaluate the quality of the approximation, e.g., computing the delta volume between the buildings of a cell and the generalized block. This metric could guide the generalization process to indicate areas where a different approach should be used.

Another conceptional limitation is the absence of 3D terrain both in presentation and generalization. Many cities have a terrain with significant height variance, and visualizing them on a flat terrain would be a wrong approximation. A simple integration would adapt the cell blocks to the underlying terrain curvature as sketched in Figure 3.25. This would require the tessellation of the currently flat roof polygon and adding terrain height for each vertex. However, more preferably would be a deep integration in the generalization process, as terrain height differences might also account for aggregated shapes and the identification of landmarks. Break lines in the terrain as indicators of significant height changes could be used as additional line features from which the arrangement may be computed. Also, the base height could be used as an attribute during landmark detection. The approach of Guercke et al. (2011) could be used for further steps to incorporate the terrain.

Looking at the technical limitations, the arrangement computation can be identified as a bottleneck. At this stage, all line string geometries have to be processed at once to create the cells; there is no tiling mechanism that could be used for an application of the divide-and-conquer-paradigm. A solution, working in parallel on tiled input line strings, would need to identify and merge boundary cells of neighboring tiles afterwards. However, all the other steps down the processing pipeline are suitable for parallelization, as they operate independently on single cells.

A straightforward practical extension would be support for standardized styling descriptions to define color and style of elements of 3DCMs, including abstract buildings, vegetation, and water features. Standards defined for 2D maps could used, e.g., styled layer descriptors (SLD) or symbology encoding (Neubauer and Zipf, 2007; OGC, 2006).

Finally, designing and conducting formal user studies would be important to evaluate the effectiveness of our approach. It should be investigated whether the presented LOAs are generally accepted as abstract replacements of single building models. Also, it would be interesting to explore if and how users do understand scene elements presented simultaneously in different degrees of abstraction. The integration of additional visual cues may be necessary to fully communicate these elements.

Figure 4.1: While the standard perspective projection depicts buildings smaller with increasing distance (left), landmark buildings can be emphasized by scaling them according to their importance for navigation and orientation, enhancing the skyline (right).

CHAPTER 4

Geometric Landmark Enhancement

This chapter presents a geometric landmark enhancement technique (Glander et al., 2007). The goal of this technique is to enhance the landmark objects of a 3DCM for improved orientation. It achieves this by scaling them in real-time, depending on the virtual camera distance (Figure 4.1).

4.1 Potentials and Usage of Geometric Landmark Enhancement

In a real world environment we tend to filter structures and objects according to, e.g., their visual, functional, and semantical properties. This way we are able to process a large amount of sensoric input data and build a meaningful internal representation of the environment, a mental map (Ware, 2004). In particular, objects that stand out from their surroundings by their properties are used as landmarks. As discussed before (Section 3.7), landmarks are essential elements for navigation and orientation. Their integration and emphasis in visualizations of 3DCMs represent a major requirement for effectively using 3DCMs.

In classical 2D maps, different visualization styles for landmark buildings are used, ranging from textual and iconic to realistic styles (Figure 4.2). The cartographer chooses the representation with the appropriate abstractness from this continuum (Elias and Paelke, 2008). In addition, depending on the current scale, landmarks can be depicted larger than their neighborhood (Hake et al., 2002; Imhof, 1972), highlighted by using different colors or drawing styles, and exposed by clearing their immediate surrounding.

However, interactive, dynamic 3D visualization compared to static maps involves additional challenges, i.e., occlusion because of perspective forshortening occurs which may hide important landmark objects. In addition the interactive visualization of a 3DCM repeatedly changes the image, requiring constant attention of the user.

A concept and implementation of geometric landmark enhancement are presented in the following. The technique aims at dynamically highlighting landmark objects by considering the current view and applying deformation as a working principle. It resolves the problem of landmarks

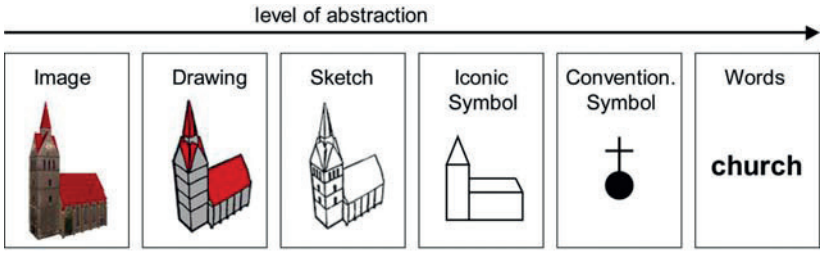


Figure 4.2: Different visualization styles for a church serving as a landmark building (from (Elias and Paelke, 2008)).

being occluded or too small. By scaling landmark objects depending on the distance to the virtual camera, their visibility is enhanced. This enhancement supports the user in taking advantage of the landmarks and can be justified with their importance. In addition, surrounding objects are displaced as a consequence of the exaggeration.

The presented real-time rendering technique is generally suitable for applications which rely on visualization of 3DCMs. Particularly, location-based services (LBS) benefit from geometric landmark highlighting, e.g., car / pedestrian navigation, social location-aware services, and queries for nearest points-of-interest. First, recognizable landmarks help in aligning visualization of the 3DCM, mental map, and reality. Second, dynamic enhancement of important objects in the visualization enables it to dynamically adapt the enhancement effect to the current location, changing purposes, and tasks. As today's mobile devices increasingly include visualization of 3DCMs to implement LBS, they have to make use of visualization techniques that help to guide the users effectively.

4.2 Landmark Visualization and Focus+Context Techniques

Two major classes of approaches related to the presented technique can be identified and are discussed in the following: first, we review visualization of landmarks in maps and 3DCMs, then we look at distortion-based focus+context techniques.

4.2.1 Landmarks in Maps and 3DCMs

Research in landmark visualization touches several areas including visualization style and generalization context. In addition, adaptive maps and map-like representations include landmarks.

Regarding visualization styles for landmark buildings, Elias and Paelke (2008) analyze different graphical representations and introduce a design matrix to help choose the appropriate one for different categories of buildings, e.g., commercial buildings, visually outstanding buildings. For the case of perspective maps, Lee et al. (2001) suggest depicting landmark buildings by placing photographs in the scene, which have been taken from a similar perspective.

In context of generalization, enhancement of important features such as landmarks is an operator belonging to cartographic generalization. The resulting conflicts, due to potential overlap with surrounding features, need to be resolved by, e.g., local incremental displacement (Ruas, 1998) or global optimization (Sester, 2002a).

The management of landmark objects in maps and map-like visualizations is an ongoing major challenge for effectively providing LBS. In contrast to printed maps, LBS require maps and map-like visualizations to adapt to the spatial, time, and social context of the current usage, leading to egocentric maps (Reichenbacher, 2005).

3D geovirtual environments, as a specific class of map-like visualizations, need landmarks to ensure that users can orientate themselves and navigate. Vinson (1999) presents detailed guidelines for design and placement of landmarks in virtual environments, including:

- Landmarks should be visible at all times, especially at all navigable scales.
- Landmarks should be distinguishable from their environment.
- Concrete representations of landmarks should be preferred over abstract ones.

4.2.2 Focus+Context Visualization

Focus+context visualization is a principle of information visualization. It highlights the most important parts of an image or a scene, e.g., by showing them with maximum details or scaled, while also showing the surrounding area, the context. With the help of interactive, dynamic, and 3D image generation, information visualization attempts to reveal structural relationships (Robertson et al., 1993). The following approaches implement focus+context visualization with the help of geometric deformations.

The application of focus+context visualization by scaling focus and context regions differently needs to address the transition between the two regions. An example for the case of 2D lenses is the work of Carpendale et al. (2004), who propose to place the image on a planar grid in a perspective view. The parts of the grid which are in focus are then moved towards the virtual camera while a dropoff function describes the transition to the contextual parts. The final rasterized image shows the scaled focus parts embedded in context. Special handling is necessary to minimize distortions, for example by requiring the distortion to be conformal, preserving the local shapes (Zhao et al., 2012).

In 3D focus+context visualization, occlusion of the focus region must be additionally addressed. To do this Carpendale and Cowperthwaite (1996) scale important features and clear the view to them, thus considering the current view configuration. These approaches distort the mesh vertices so that the impression of magnification occurs.

In applications visualizing 3DCMs, focus+context techniques are used to broaden roads (Grabler et al., 2008; Qu et al., 2009) or to scale important buildings. Exploiting the presence of a 2D reference plane in 3DCMs, focus+context techniques can compute the deformation in 2D, e.g., broadening a route, displacing buildings in a block (Qu et al., 2009), or highlighting a region of interest (Degener and Wahl, 2008).

4.3 A Geometric Landmark Enhancement Technique

In the following, we present our approach for landmark enhancement. To ensure the visibility of landmark objects across many scales in a 3DCM, we assume displayed landmarks have a minimal size that allows the viewer to recognize them. This can be obtained by enlarging the landmark objects, also resulting in less space for surrounding objects. To keep the distortion effect local and prevent objects from overlapping, compression and displacement have to be applied. However, it is important to maintain characteristic shape properties of buildings such as orthogonal and parallel walls.

State-of-the-art generalization systems used for mapping apply computationally expensive algorithms to find an optimal or near optimal solution when displacing objects, e.g., simulated annealing (Ware and Jones, 1998) or least squares adjustment (Sester, 2000a). However, for real-time visualization this is not feasible. In interactive 3D applications view parameters, e.g., virtual camera distance and viewing angle, change continously as the user moves through the virtual 3D scene. Hence, we need an algorithm enabling interactive computation of scaling and displacement.

In our approach we differentiate landmark and non-landmark buildings. Also, we assume that there are few landmark buildings at the same time, while there are many non-landmark buildings. With this separation, the technique implements the following:

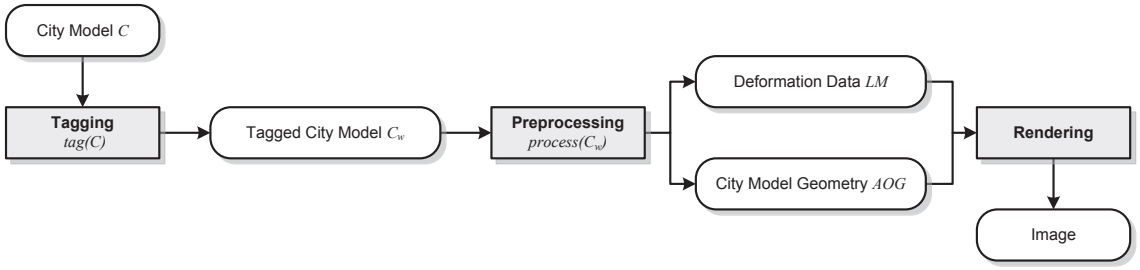


Figure 4.3: Components and processing stages of our visualization technique.

- Representations of landmark objects are scaled depending on the virtual camera distance to ensure their visibility.
- To prevent overlapping representations of landmark objects, they are displaced by repelling themselves mutually.
- To prevent representations of non-landmark objects from being overlapped, they are displaced and scaled down.

In Figure 4.3 we illustrate how the technique can be implemented in a visualization pipeline (Ware, 2004). The input data is a 3DCM. It has to be augmented with weights defined per object during the data gathering stage (tagging) to distinguish landmark objects from non-landmark objects. The tagged 3DCM serves as input for the next stage (preprocessing), where in an automatic process both geometry and deformation data are derived; these are required for the real-time deformation. At runtime the deformation data is evaluated and the objects of the city model are deformed, creating subsequent images for the impression of interactive visualization. The user can explore and navigate within the 3DCM, leading to permanent updates of the deformation model and hence, the scaling of the landmark objects.

4.3.1 Data Model and Preprocessing

As before (Chapter 3.4), we refer to a 3DCM $CM = (B, I, N)$ with features of buildings B, infrastructure elements I, and non-building areas N. In the focus of this technique are building models and their 3D geometry. We need explicit encoding about the landmark objects and their different weights. Therefore, the building features have to be enriched with weights from a weighting function $w(b_i)$, resulting in tagged building features B_w.

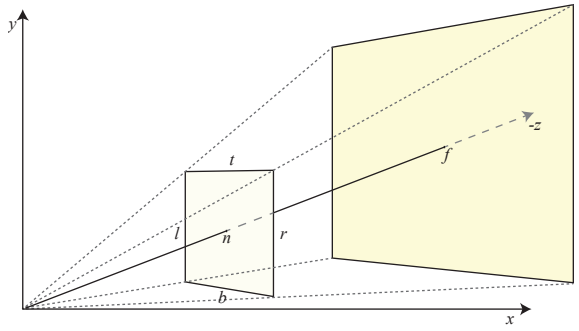
$$\mathrm{tag}(B) = B_w = \{(b_0, w(b_0)), ..., (b_n, w(b_n))\}$$

While common building features receive weight $w(b_i) = 1$ per default, buildings serving as landmark objects are mapped to higher weights. Hence, the weighting function defines a partition into two sets of building features: landmark buildings and non-landmark buildings.

For the building weights, we depend on external sources and assume a properly defined weighting function. Promising approaches propose to compute and integrate visual, structural, and semantical properties of the buildings and identify landmark objects (Brenner and Elias, 2003; Grabler et al., 2008). The weights are assigned prior to the rendering, hence any external data source can be integrated, e.g., a web service, a data base query, or results from importance computation.

4.3.2 Scale Factor for Landmark Objects

The presented technique enhances the visibility of the landmarks by scaling them depending on their distance to the virtual camera. The computation is based on the observation of the inverse



$$P = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Figure 4.4: The OpenGL projection matrix P is defined by the parameters of the view frustum.

proportional relation between distance and projected size as a consequence of the perspective projection.

In the following we assume a perspective projection, which is typically used in photo-realistic 3D visualization. To ensure that displayed elements are legible they must have certain minimal dimensions (Häberling et al., 2008), which depend on the visualization medium, e.g., the resolution of a screen, the distance of the user to the screen, and the user's abilities. The depicted size on screen depends on the settings of the projection.

Vertex Transformation Pipeline

We will briefly recapitulate the terms for the transformations that geometric models undergo in the standard rendering pipeline (Foley et al., 1995):

- Initially, the vertices of a geometric model are defined in an *object-coordinate system*.
- By applying a modeling transformation, the model is placed spatially into the context of a larger scene. The vertices are defined in a 3D *world-coordinate system*.
- To decouple the viewpoint from the scene, application of a separate view transformation encodes the positioning of the scene relative to a virtual camera. After applying it, the model's vertices are in *eye-coordinate system*.
- The next transformation puts the model into *normalized-projection coordinates system*. In case of a perspective projection, the view frustum is distorted to a canonical view volume.

Further transformations that map the scenery to the view port and subsequent rasterization are described in (Foley et al., 1995). In OpenGL the perspective transformation is done with the projection matrix P, which transforms the view frustum into the canonical view volume. It is constructed using values for near n and far f plane, top t, bottom b, left l, and right r distances of the view frustum (Akenine-Möller et al., 2008) (Figure 4.4).

An eye-coordinates vertex v_e in homogenous coordinates, i.e., $v_e = (x_e, y_e, z_e, 1)^T$ is multiplied by the perspective transform matrix P, yielding *clip-coordinates* v_c. The projected vertex v_p is then obtained by applying the perspective division, i.e., dividing by the homogeneous coordinate.

$$v_c = \begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = P v_e, \quad v_p = \begin{pmatrix} x_c/w_c \\ y_c/w_c \\ z_c/w_c \end{pmatrix}$$

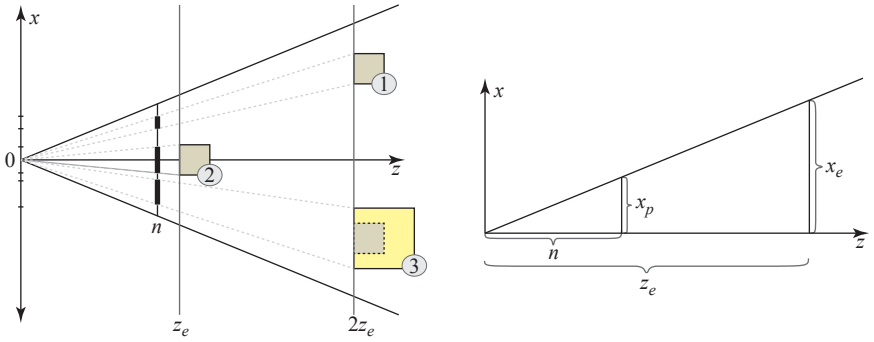


Figure 4.5: Effects of perspective projection in eye-coordinate system: left: depending on the virtual camera distance, different projected sizes are obtained for the boxes (1) and (2) at the image plane at distance n. To achieve the same projected size as box (2) at $d = z_e$, box (3) at $d = 2z_e$ needs to be enlarged by a factor of two. Right: the projected extent x_p can be derived from trigonometric relations between near plane distance n, extent in eye space x_e, and virtual camera distance z_e.

Using the standard perspective projection transformation, an object with extent x_e in eye coordinates is transformed to an extent x_p. The projected size depends on the distance of the object to the projection plane, i.e., the virtual camera distance, and an inverse proportionality can be observed. Figure 4.5 illustrates the effect: doubling the virtual camera distance halves the projected size of the same object. In consequence, an object needs to be scaled by a factor of two to obtain the same projected size when the distance is doubled. With the help of trigonometric relations, the size of the projected extent x_p can be obtained: $x_p = nx_e/z_e$. This reveals the inverse proportional relation between projected extent and virtual camera distance.

Determining a Landmark's Scale Factor

As a consequence of these observations, we design a scale factor applied in eye-space that incorporates the current virtual camera distance to maintain the projected size. As an intuitive way to control the scaling, we propose a visibility interval $I = [d_{\text{start}}, d_{\text{end}}]$ with respect to the virtual camera distance with the following properties:

- When looking at the landmark object from different distances within the visibility interval, the projected size remains the same as at distance d_{start}.
- Outside of the interval, the projected size is subject to the usual perspective foreshortening, i.e., the landmark object's size decreases with increasing distance.
- At the interval's boundaries, the scaling factor is smoothed to avoid hard switches.

Being able to keep landmark buildings visible within a given distance aligns well with the observation that landmarks are bound to reference regions (Winter et al., 2008). Moreover, humans apparently use landmarks hierarchically, as there are buildings serving as a landmark in a small environment, e.g., within a neighborhood, but there are also buildings which are referred to in a larger context, e.g., for the whole city. Hence, we map landmark weights to different distances d_{end}, which are used by our technique to parameterize the scaling function (Table 4.1). With this parameterization highly weighted landmarks get a big reference region, approximated by the radius d_{end}. As a result they are visible from greater distance compared to less weighted landmarks.

Table 4.1: Proposed d_{end} for different weight classes. As no established classification and weighting system of landmarks exist, the values are arbitrarily chosen and need to be specified as part of the configuration of the technique.

w	1	2	3	4	5	6	≥ 7
reference region	no landmark	neighborhood	district	city	regional	state	global
d_{end} in m	-	2 000	5 000	10 000	30 000	100 000	∞

It is straightforward to obtain a constant projected size of an object within an interval by scaling it proportionally to its camera distance, leading to a piecewise defined function:

$$s'(d, d_{\text{start}}) := \begin{cases} d/d_{\text{start}} & \text{if } d_{\text{start}} \leq d < d_{\text{end}} \\ 1 & \text{else} \end{cases}$$

Multiplying a vertex in eye-space with $s'(d)$ yields a projected extent that equals the projected extent at d_{start}:

$$\begin{aligned} \text{extent at } d_{\text{start}}\text{:} \quad & x_p = \frac{n x_p}{d_{\text{start}}} \\ \text{extent at } d_{\text{start}} \leq d < d_{\text{end}}\text{:} \quad & x_p = \frac{n(x_p \cdot s'(d))}{d} = \frac{n x_p}{d_{\text{start}}} \end{aligned}$$

As the scale factor is computed depending on camera distance during navigation within the 3D scene, this results in discontinuous switches of scaling at the boundaries of the interval. To smoothly embed the distortion within the regular projection, the distance dependent scale factor is blended with the unity factor at the interval boundaries. We propose a smoothness environment ϵ relative to the size of the interval, e.g., a 10% environment: $\epsilon = 0.1 \cdot (d_{\text{end}} - d_{\text{start}})$. The final smooth scaling function is composed of five intervals

$$s(d, d_{\text{start}}, d_{\text{end}}) := \begin{cases} \text{blend}(1, d/d_{\text{start}}, d_n(d_{\text{start}} - \epsilon, d_{\text{start}} + \epsilon, d)) & \text{if } d_{\text{start}} - \epsilon \leq d < d_{\text{start}} + \epsilon \\ d/d_{\text{start}} & \text{if } d_{\text{start}} + \epsilon \leq d < d_{\text{end}} - \epsilon \\ \text{blend}(d/d_{\text{start}}, 1, d_n(d_{\text{end}} - \epsilon, d_{\text{end}} + \epsilon, d)) & \text{if } d_{\text{end}} - \epsilon \leq d < d_{\text{end}} + \epsilon \\ 1 & \text{else} \end{cases}$$

with the blend function depending on the normalized distance $d_n(a, b, d) = (d - a)/(b - a)$ in the respective interval $[a, b]$. A cubic interpolation between the two given values accomplishes a smooth transition. The resulting factor is also clamped to be within the blend boundaries $[1, d/d_{\text{start}}]$.

$$\text{blend}(a, b, t) := \min(\max(a \cdot (1 - 3t^2 + 2t^3) + b \cdot (3t^2 - 2t^3), a), b)$$

Figure 4.6 plots graphs for no scale factor (a), distance proportional scale factor (b), and smoothed distance proportional scale factor (c). The resulting projected size as a function of distance shows how $s(d)$ maintains the size within the interval boundaries. The effect is illustrated in Figure 4.7 and compared with the standard projection.

4.3.3 Displacing Landmark Objects

Enlarging multiple landmark objects within the virtual 3D scene eventually leads to a collision with neighboring landmark and non-landmark objects. We handle both differently and discuss landmark objects in the following.

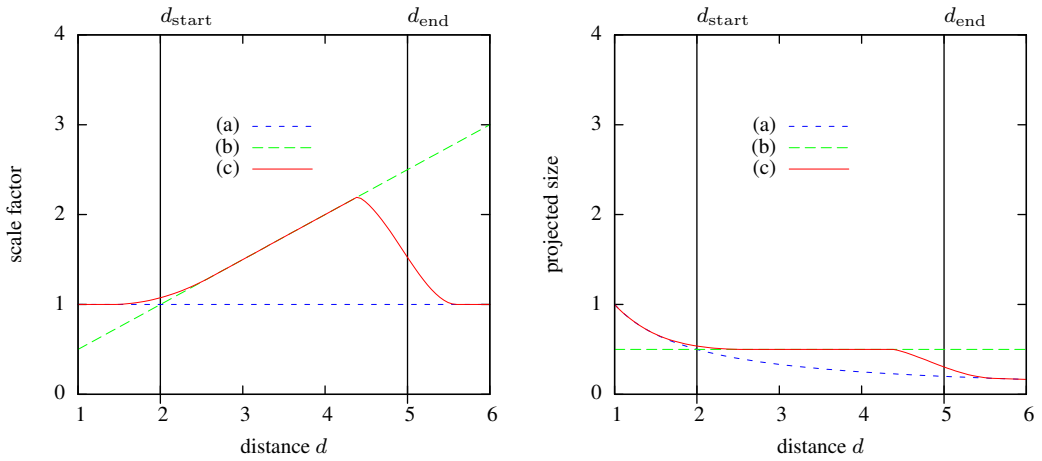


Figure 4.6: Graph shows an example with three alternatives for the scale factor (left) and a resulting projected size (right). Usually, virtual scene objects are not scaled, their scale factor equals 1 (a). To maintain a constant projected size of an object in the scene starting at distance d_{start}, the object has to be scaled by d/d_{start} (b) . For a smooth transition at the interval boundaries $[d_{\text{start}}, d_{\text{end}}]$, the two first alternatives are blended using cubic interpolation within ϵ environment, yielding the scaling function $s(d)$ (c). In the example, $d_{\text{start}} = 2$, $d_{\text{end}} = 5, \epsilon = 0.2 \cdot (d_{\text{end}} - d_{\text{start}})$.

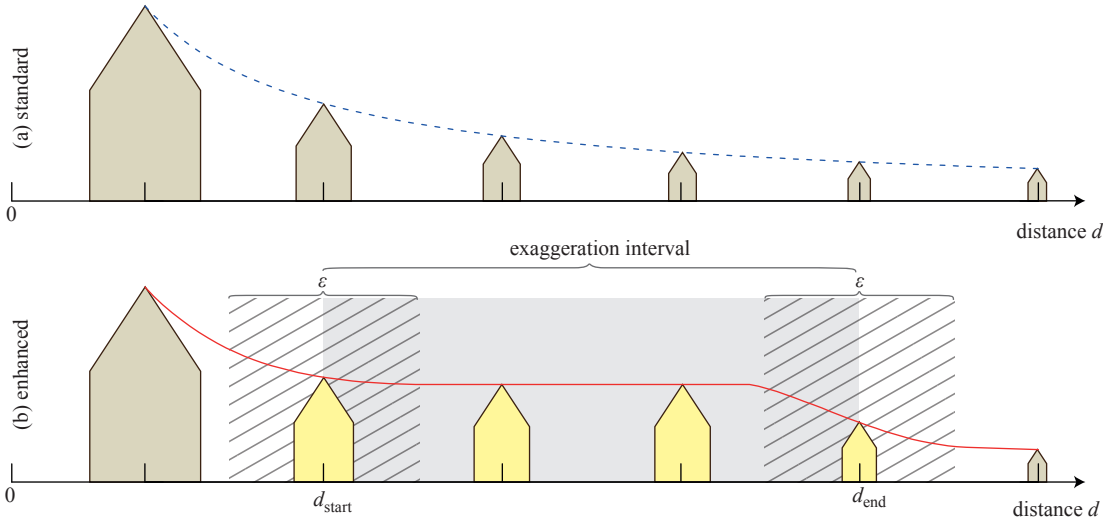


Figure 4.7: The illustration shows the effect of standard projection (a) and projection of scaled objects (b). The enhanced objects maintain their size in the exaggeration interval, smoothed at the interval boundaries within ϵ environment.

Using an appropriate spatially balanced distribution of hierarchical landmarks (Section 3.7) and careful setup of their visibility intervals, the collision of landmarks is minimized. In those cases where it still occurs we apply a simple displacement model that repels landmark objects mutually. The displacement considers the center position and scaled radius of each landmark's footprint. When displacing landmarks, an additional *distortion zone* relative to the scaled radius is cleared to take into account non-landmark buildings (Figure 4.9). The new position is obtained by computing an offset in the displacement algorithm.

The algorithm works as follows (Figure 4.8): the offset for each landmark building is initialized with zero. Then in a loop over all pairs of landmarks, the mutual displacement is computed: if the Euclidean distance between two landmarks is too small, i.e., their distortion zones overlap, a fraction of the vector between them is added to the individual offset vector. The displacement is weighted according to the weights of the two landmarks. If several landmarks are colliding, they all add to the offset stored for each landmark, so eventually after a sufficient number of iterations, the overlap is resolved. The loop stops when no more displacement occurs or when a maximum number of iterations, e.g., 1 000, is reached. If the displacement cannot resolve collision within the limit of iterations, we accept overlapping landmarks in favor of maintained interactivity of the visualization.

Due to the quadratic runtime complexity, this algorithm is only feasible for a small number of landmarks. However, assuming that the number of landmarks shown at a time should be limited to prevent information overflow, it is applicable. Also, it aligns well with the observation of the hierarchical importance of landmarks.

4.3.4 Displacing and Scaling Non-Landmark Objects

Besides a small number of landmark objects, the visualization technique has to manage non-landmark objects. As the number of affected objects become very large with extreme enlargement of landmarks, individual displacement would incur high computational costs. In contrast to offline processing for map production, processing time has to be minimized for interactive visualization. Therefore, we choose to apply a geometric transformation with respect to the nearest landmark instead of individual displacement of the affected objects.

The exaggeration effect needs to be embedded smoothly in the undistorted environment. Hence, we propose a radial distortion of individual non-landmark building models dependent on distance d_{λ^*} and current scale factor s_{λ^*} of the nearest landmark λ^*. To keep the effect local, it is limited to the distortion zone centered at the landmark's centroid (Figure 4.9). Its radius depends on the scaled radius of the landmark's footprint; we propose to use twice the scaled radius ($\zeta = 2$).

Figure 4.10 shows a synthetic scene with an enlarged cube in the center and surrounding cubes that need to be distorted to make space for the enlarged cube. A number of alternatives exist to achieve the distortion, we discuss these with respect to their suitability:

(a) Surrounding objects could be scaled down with a constant factor and displaced.

(b) Surrounding objects could be scaled according to a lens centered at the enlarged cube as described by Carpendale et al. (2004). Here the distortion is applied per-vertex according to a linear dropoff function.

(c) Surrounding objects could be scaled according to a lens, as in (b), but applying the distortion uniformly per-object.

(d) Surrounding objects could be downscaled linearly with respect to the landmark distance from the original size at the center to zero at the edge of the distortion zone.

```
// landmark attribute struct
struct LMattribute{
  int id;               // unique building identifier
  float r;              // radius of 2D axis aligned bounding rectangle
  Vec3 c;               // centroid of footprint projected to the terrain surface
  float s;              // scale factor
}

// Global parameters of displacement algorithm
int MaxLM = 10;         // maximum number of landmarks to consider in one frame
float ζ = 2.0; // distortion zone factor
float DisplacementFraction = 0.01; // step size of displacement
int MaxIterations = 10000;  // maximum iterations of the displacement algorithm

// Apply mutual displacement of landmark objects by computing
// an offset vector for each landmark object
Vec2[MaxLM] displaceLM(LMattribute landmarks[MaxLM]){
  // initialize
  Vec2 LMoffsets[MaxLM] = initializeOffsets(MaxLM); // init with Vec2(0,0)
  int iterations = 0;

  // outer displacement loop
  do{
    iterations++;
    bool needsDisplacement = false;

    // inner pairwise displacement loops
    for (int i=0; i<MaxLM; i++){
      for (int j=0; j<MaxLM; j++){
        if (i==j) continue;

        // compute scaled extent of both current landmarks
        float r0 = landmarks[i].r * LMscaleFactor[i];
        float r1 = landmarks[j].r * LMscaleFactor[j];

        // compute connecting 2D vector considering potential offsets
        Vec2 vij = landmarks[i].c.xy + LMoffsets[i] -
                   landmarks[j].c.xy + LMoffsets[j];

        // check overlap of distortion zones
        float deltaDistance = ζ * (r0 + r1) - ||vij||;
        if (deltaDistance>0){
          needsDisplacement = true;

          // compute weight for displacement: bigger scale means less movement
          float w0 = LMscaleFactor[j] / (LMscaleFactor[i] + LMscaleFactor[j]);
          float w1 = LMscaleFactor[i] / (LMscaleFactor[i] + LMscaleFactor[j]);

          // move landmarks to the opposite direction depending on weight
          Vec2 o0 = -v̂ij * deltaDistance * w0;
          Vec2 o1 = v̂ij * deltaDistance * w1;

          // soften movement by using just a fraction for offset
          LMoffsets[i] = LMoffsets[i] + o0 * DisplacementFraction;
          LMoffsets[j] = LMoffsets[j] + o1 * DisplacementFraction;
        }
      }
    }
  }while(needsDisplacement && iterations < MaxIterations);
  return LMoffsets;
}
```

Figure 4.8: The landmark displacement algorithm computes the offset vector for each landmark object.

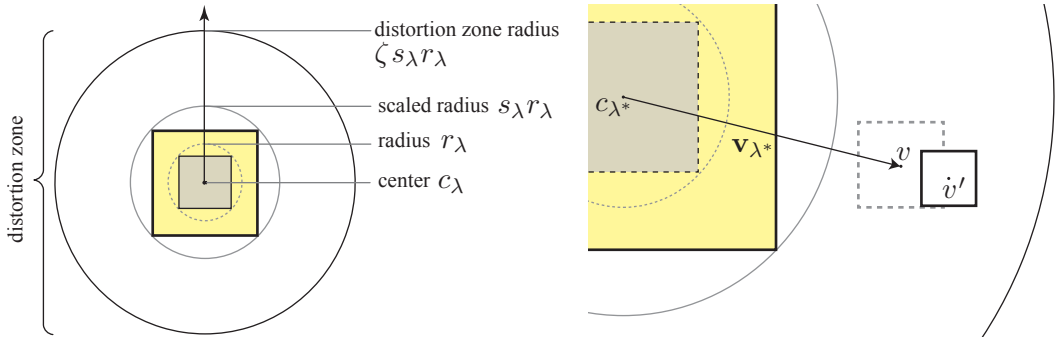


Figure 4.9: Non-landmark objects within the distortion zone are scaled and displaced. These objects are distorted to make space for the scaled landmark object but they still remain in the distortion zone.

(e) Surrounding objects could be downscaled inverse-linearly with respect to the landmark distance from zero at the center to the original size at the edge of the distortion zone.

The alternatives (a) and (d) do not embed the distortion effect smoothly into the environment, as there is a discontinuity at the edge of the distortion zone. The distortion lens (b) from all shown alternatives uses the space best, but applies the transformation independently on each vertex (Carpendale et al., 2004). Therefore, geometric properties as orthogonality and parallelism get lost; it is better suited to mimic optical zooming of a region of interest. If the distortion lens is applied per-object (c), overlap occurs between the objects. We choose the inverse-linear scale (e), as it integrates the lens smoothly in the environment, avoids overlap, and exposes the enlarged object in the center.

For the implementation we decompose the distortion into a translation and a scaling.

Translation: For an object's vertex v within the distortion zone, we use the vector to the nearest landmark's center, $\mathbf{v}_{\lambda^*} = v - c_{\lambda^*}$, to compute the displacement: the translation $t : \mathbf{v}_{\lambda^*} \mapsto \mathbf{v}_{\lambda^*}{}'$ scales the normalized vector by the sum of the scaled landmark radius and a distortion component depending on the distance of the vertex to the current landmark's center:

$$t(\mathbf{v}_{\lambda^*}) = n(\mathbf{v}_{\lambda^*}) \left(\underbrace{s_{\lambda^*} r_{\lambda^*}}_{\text{scaled radius}} + \underbrace{\frac{(|\mathbf{v}_{\lambda^*}| - r_{\lambda^*}) s_{\lambda^*}}{\zeta s_{\lambda^*} - 1}}_{\text{distortion}} \right).$$

Applying this translation on the building's vertices separately effectively would compress the geometry to fit into the distortion zone. However this would also distort the building's properties of orthogonality and rectangularity, analoguous to Figure 4.10 (c). Therefore, we translate all vertices of one building by the same offset, obtained by computing the translation function for the building's center.

Scaling: To prevent individual building objects from moving into each other, we also apply downscaling to the objects. The downscaling depends inverse-linearly on the distance to the landmark's center, i.e., an object located further away from the landmark is downscaled less. The computed value from $\text{scaleFactor} : \mathbf{v}_{\lambda^*} \mapsto s \in [0, 1]$ is applied as a uniform scaling of each building object, effectively compressing it while maintaining its orthogonal shape:

$$\text{scaleFactor}(\mathbf{v}_{\lambda^*}) = \frac{|\mathbf{v}_{\lambda^*}|}{\zeta s_{\lambda^*} r_{\lambda^*}}.$$

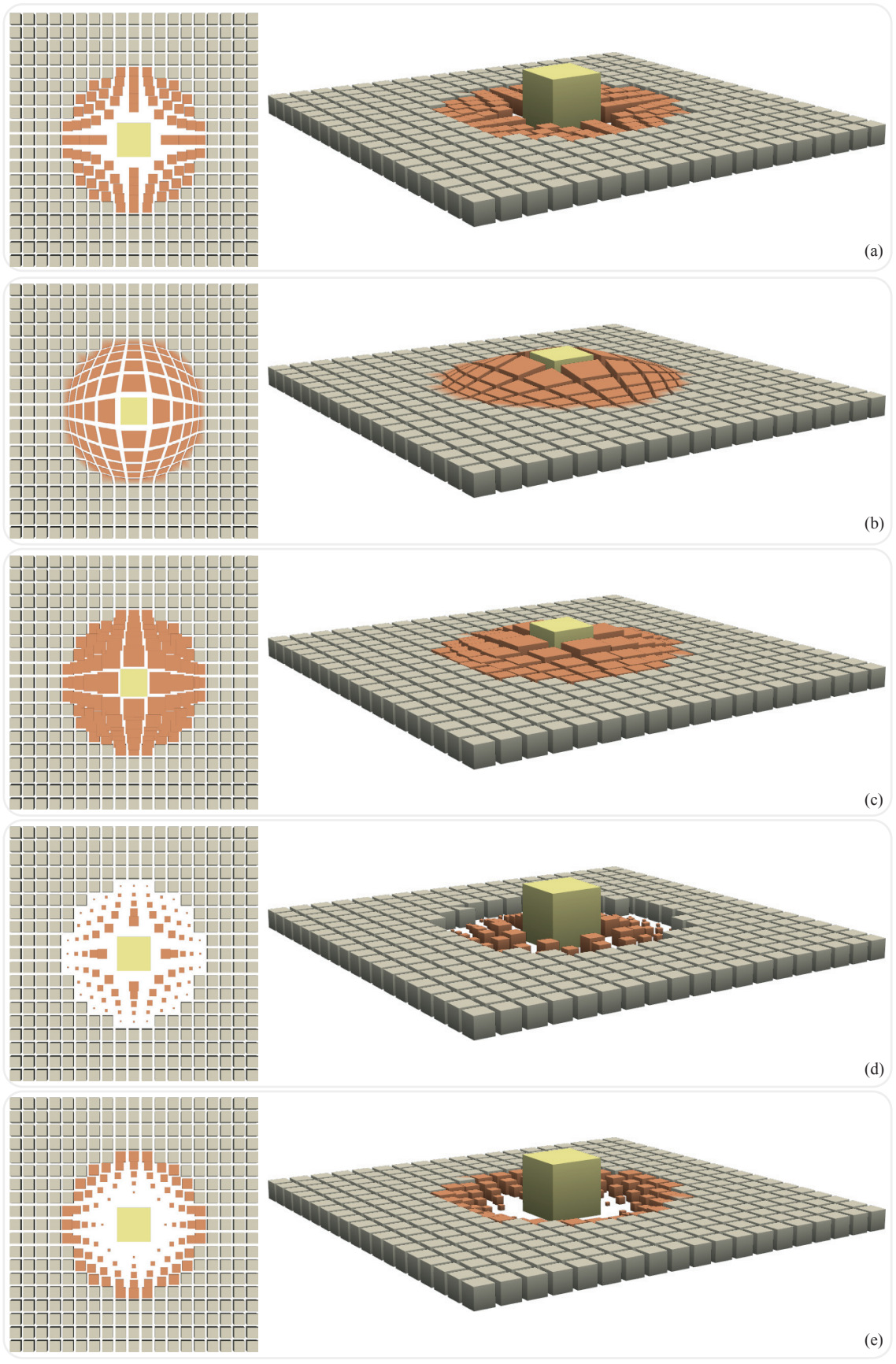


Figure 4.10: Comparison of different distortion variants for non-landmark objects. The synthetic scene shows an enlarged object (yellow) and the effect of distortion applied to objects within the distortion zone (red). The variants are (a) constant scaling, (b) Carpendale distortion per-vertex, (c) Carpendale distortion per-object, (d) linear scaling, and (e) inverse-linear scaling.

Finally we need to address overlapping distortion zones from multiple landmark objects. For it, displacement vectors and scale factors are computed with respect to each neighboring landmark object in a loop over all landmark objects. The final displacement is the averaged displacement vector, while the final scaling is the product of all scale factors.

4.4 Implementation Details

We implement the proposed technique using graphics hardware to enable parallel application of displacement and scaling operations. The computation is balanced between CPU and GPU: computation of the landmarks' scale factor from given visibility intervals and their mutual displacement vectors is done on the CPU. The application of the landmark scaling and displacement, as well as displacing non-landmark buildings, is done in parallel on the GPU. The implementation relies on the scene graph-based high-level rendering framework VRS. However, it is not bound to a specific implementation of a scene graph as the principles of the landmark enhancement technique are general. In the following we describe how to set up the rendering data sent to the graphics hardware and the actual rendering.

4.4.1 Preparing Rendering Data

To perform displacement and scaling on CPU and GPU, the scene objects have to be set up in a way that allows the GPU to efficiently access the necessary parameters. Modern GPUs have a programmable rendering pipeline to enable control of the rendering at different stages, including vertex, primitive, and fragment level. The necessary data can generally be provided in two formats:

- The data may be provided as *vertex attributes*. In addition to the vertex position, vertex attributes such as normal, color, and texture coordinates are typically transferred to the GPU. A vertex program is invoked on one vertex with its attributes, i.e., it cannot access the other vertices and their attributes.

- The data may be provided as *uniform variables*. They are typically used to store tranformation matrices, light positions, and texture raster data. In contrast to vertex attributes, they are available across invocations of GPU programs.

To adress efficient GPU processing with a scene graph structure, we need to map the scene graph to GPU efficient data. Therefore, we divide the data required for geometric landmark enhancement into attributes nodes, which are mapped to uniform variables, and geometry nodes, which are mapped to vertex attributes, later (Figure 4.11). This enables at runtime to select the buildings that are enhanced, without the need to recreate the geometry nodes.

The attribute nodes λ_i describe for each landmark the enhancement parameters needed to compute the scaling and displacement, i.e., position, radius, visibility interval $[d_{\text{start}}, d_{\text{end}}]$, and unique identifier. They represent the data that is accessible by the CPU to compute the scaling and landmark displacement.

For efficient displacement of non-landmark buildings and general rendering of the scene, the geometry and displacement data per building is prepared as geometry nodes. Efficient rendering depends on a small number of draw calls per frame (NVidia, 2008), however, geometry such as in 3DCMs typically comprises many individual objects. Therefore, we aggregate the geometry into batches of geometry buffers of vertex attributes. Instead of issuing a draw call for each triangle or building, a huge number of building objects can be drawn with one call. In addition we apply tiling to store the geometry data according to a spatial organization. This enables culling by the CPU to send only visible geometry to the GPU.

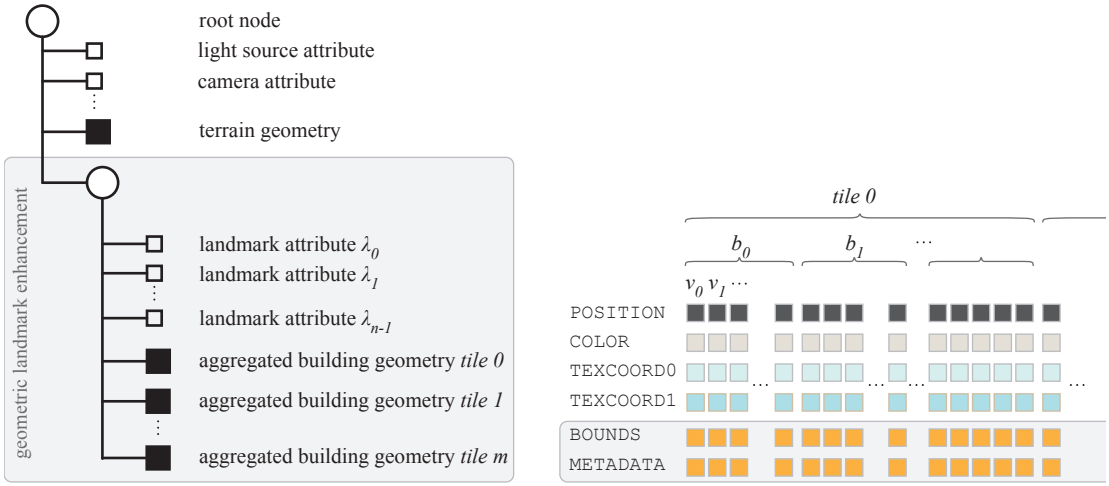


Figure 4.11: Left: the scene graph contains a landmark attribute for each landmark λ and aggregated batches of geometry data. Right: the geometry data comprises buffers of vertex attributes storing bounding rectangle, identifier, and a landmark Boolean. To enable parallel processing by the GPU, the data is aggregated into large buffers. Tiling is applied to enable CPU side culling of buffers.

Table 4.2: For each vertex of city model geometry, we store additional attributes needed for computation of the distortion effect along with the usual attributes needed for rendering. Attributes for rendering besides the position encode color and texture coordinates for color and shadow texture. The additional attributes encode the bounding rectangle of the city object, its unique number, and a Boolean encoding whether the object is a landmark.

POSITION	X	Y	Z	W
COLOR	R	G	B	A
TEXCOORD0	S_0	T_0		
TEXCOORD1	S_1	T_1		
BOUNDS	X_0	Y_0	X_1	Y_1
METADATA	*ID*	*LM*		

However, to maintain object identity as needed for the proposed landmark enhancement technique, we have to store additional data per vertex which facilitates the reconstruction of object properties within shader programs. In addition to data required for rendering, we therefore add attributes encoding object ID and bounding rectangle of the building object. The redundancy introduced by storing the same values for each vertex of a building can be justified with the maintained object identity and improved rendering efficiency of the batched geometry.

During preprocessing, the required vertex attributes are derived from the building objects. That is, for each building feature b_i a unique object *id* is created, and its axis-aligned bounding rectangle *bb* is computed and set for each vertex. The vertex attributes are then aggregated in a few large geometry buffers containing the data of many building objects (Table 4.2).

4.4.2 Implementation of a Rendering Technique

The rendering technique processes the scene graph at runtime. It uses two passes: a pre-traversal pass and a main-traversal pass. Both passes must be evaluated per computed frame, since the deformation parameters depend on the current camera settings which can be changed in an interactive system:

1. Pre-Traversal Pass: the rendering engine collects landmark attributes stored in the scene graph. Optionally, this set can be culled against the current view-frustum to reduce further calculations to contributing landmarks. Now the deformation model described in Section 4.3.3

Table 4.3: For each landmark λ we store uniform variables of identifier, bounding rectangle, scale factor, and displacement offset together with the number of landmarks, to send them to the shader program.

	λ_0	λ_1	...	λ_{n-1}
LM_ID[]	id_0	id_1	...	id_{n-1}
LM_BOUNDS[]	bb_0	bb_1	...	bb_{n-1}
LM_SCALE[]	s_0	s_1	...	s_{n-1}
LM_OFFSET[]	o_0	o_1	...	o_{n-1}
LM_NUM	$n-1 \leq n_{\max}$			

determines the deformation parameters. After this, the deformation parameters are encoded in global shader constants (uniform variables) for the subsequent rendering pass.

2. Main-Traversal Pass: During this pass a vertex shader program is activated to deform every vertex of the building geometry according to the global shader constants. Additional vertex attributes such as object identity id and the buildings bounding box bb, which were set during the preprocessing of the scene geometry, enable the distinction between landmark and non-landmark geometry. The shader program then scales and displaces the landmark geometry or applies the deformation parameters to clear space for the landmarks.

During the pre-traversal pass, the scene graph aggregates the collected landmark data. Depending on camera settings and individual landmark weights, the individual scale values are determined and the landmark displacement model is executed to compute the mutual displacement of the landmark buildings. The computed values are stored as uniform variables, encoding the identifier, bounding rectangle, scale factor, and displacement offsets per landmark.

During the main-traversal pass, the uniform variables are sent to the graphics hardware, if changed, and the shader programs are applied. The maximum amount of data transferred to the graphics hardware per frame is limited by the maximum number of landmarks $n_{\max}$ supported. In general, scale and offsets have to be changed and updated per frame, as they depend on the camera settings. Bounding rectangle, landmark identifier, and number of landmarks have to be changed less frequently, e.g., if the virtual camera moves into or leaves the visibility interval of landmarks in consequence of interaction.

This approach is efficient in terms of rendering complexity because the complete scene geometry is rendered only once per frame. Thus, the rendering performance is limited only by the number of landmarks and the number of vertices of the 3DCM geometry.

4.5 Results and Application Examples

We present in the following the results of our visualization technique by applying it to the 3DCM of Berlin, a generalized 3DCM of Berlin, and together with a non-linear projection visualization as part of a Nokia maps prototype.

4.5.1 Landmark Enhancement in a 3DCM

We apply the landmark enhancement technique to the 3DCM of Berlin. The model consists of approximately 80 000 building features. They are mostly low detail LOD-2 buildings with typical façade textures and a small set of detailed LOD-3 or LOD-4 building models. During tagging, weights are manually assigned to the building models to identify landmarks. No features apart from buildings are included in the scene.

The resulting visualization application enables interactive navigation within the 3DCM, where landmark buildings are enlarged to be visible. During interaction, the scaling appears smooth when

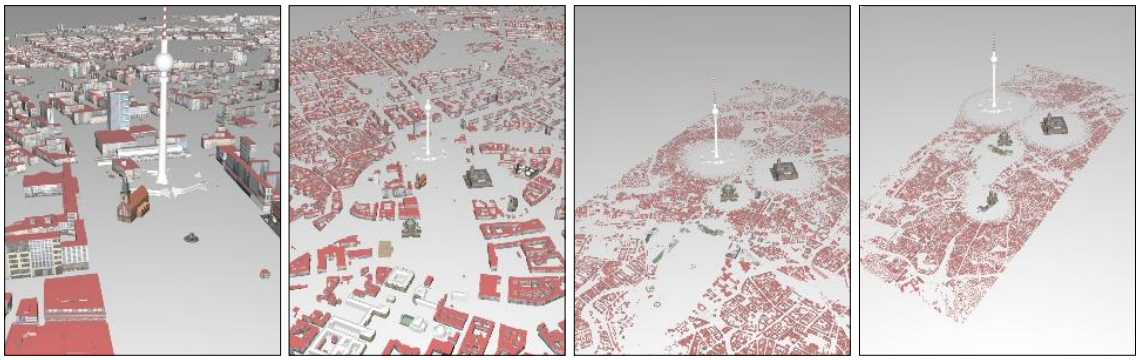


Figure 4.12: When zooming out (from left to right), eventually the virtual camera enters the visibility interval of a landmark building and the scaling and displacement is applied. On extreme exaggeration the radial distortion appears very strong (right).

zooming in and out of a landmark building. When at close range, the visualization equals the usual perspective renderings of 3DCMs. When zooming out, the landmark model is scaled to maintain a constant projected size. When the virtual camera leaves the visibility interval, the landmark model smoothly returns to its original projected size.

The displacement model pushes away residential and other landmark buildings. The radial distortion of residential buildings appears to be very strong, especially with extreme zoom out and scaling. Landmark buildings are displaced if they are getting too close to each other.

On oblique perspectives with a view point close to the ground, landmarks are distinct and clearly visible on the horizon. They enhance the city's silhouette and allow the user to identify the major directions of the city. In this perspective, the distorted residential buildings are not visible (Figure 4.1).

4.5.2 Landmark Enhancement in a Generalized 3DCM

In the second application, landmark enhancement is combined with cell-based generalization (Chapter 3). Instead of single residential buildings, generalized blocks are included in the scene to reduce details. In the application we distinguish surrounding highlighted landmark buildings and other geometries from cell blocks, land use polygons, and infrastructure elements.

As the cell blocks represent abstracted buildings, we do not displace their geometry. Instead we apply an image-based clipping against the radial distortion zone of highlighted landmarks. This simplification, however, works only for a limited scale range as it may happen that exaggerated landmarks cross surrounding roads or coast lines, which should be avoided.

In abstract visualization of 3DCM the user is more likely to expect non-photorealistic effects such as enlarged landmarks. Thus, the technique may be used to scale landmarks beyond subtle increases. This can be obtained by choosing a closer starting distance of the visibility interval d_{start} (Figure 4.13).

However with increased exaggeration, collisions between landmark models become more likely. The displacement model controlling the mutual landmarks' displacement cannot guarantee a stable frame-to-frame coherence, i.e., jumps may occur if two many collisions have to be handled.

4.5.3 Landmark Enhancement and Non-Linear Projections

In a research project together with Nokia, we implemented geometric landmark enhancement in combination with non-linear projections (Lorenz et al., 2008). Our implementation is based on *Nokia Maps*, a 3D map visualization system which can be deployed on smart phones and as

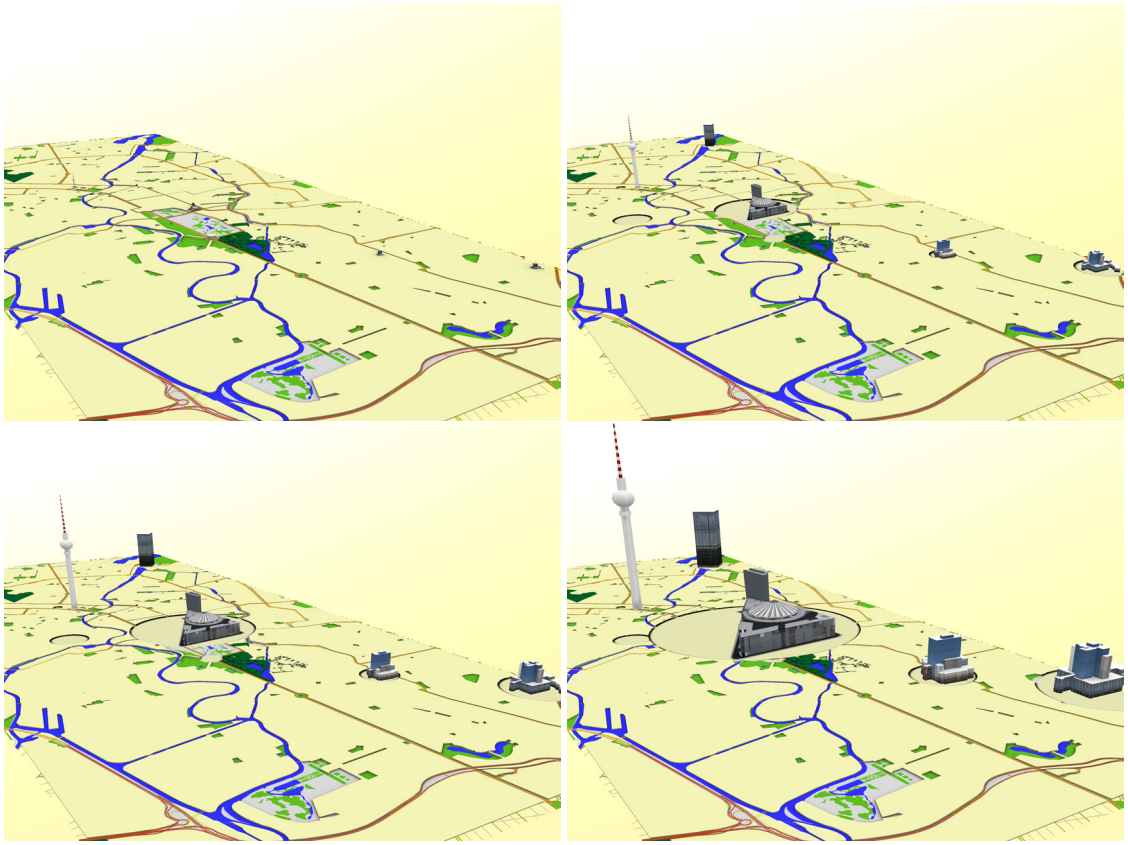


Figure 4.13: Geometric landmark enhancement applied to the generalized 3DCM of Berlin. The images show the visualization without landmark enhancement (upper left) and with different values for the starting distance from $d_{\text{start}} = 3000m$ (upper right), $d_{\text{start}} = 2000m$ (lower left), to $d_{\text{start}} = 1200m$ (lower right). Closer starting distance means greater exaggeration, allowing designers to tune the effect. The definitive version of these images can be found at (Glander and Döllner, 2009), ©Elsevier.

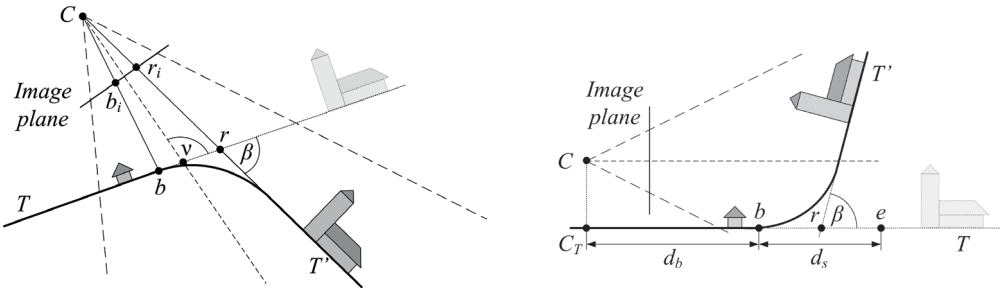


Figure 4.14: The non-linear projection deformation as presented by Lorenz et al. (2008): Left: the bird's eye view deformation shows a bird's eye view in the lower part and a horizon in the upper part. Right: the pedestrian's view deformation combines a 3D perspective view in the lower part with a top view in the upper part (schematic images from (Lorenz et al., 2008)).

a browser plugin. The research prototype includes the camera dependent scaling of landmarks without displacement. Both landmark scaling and non-linear projection techniques are implemented on the CPU for this prototype.

The non-linear projection technique aims at providing a more effective use of the screen space by seamlessly integrating multiple perspectives in one view. Depending on the virtual camera position and distance, a parametric curve defines the deformation of the terrain model and other geometry (Figure 4.14). The parametric curve is composed from two planar sections connected by a smooth transition. The authors propose two visualization modes: a bird's eye view and a pedestrian's view. The bird's eye view shows a 2D like view in the lower part of the screen, while presenting the shilouette of the city in the upper part. In the pedestrian's view the scene is bent up with increasing distance to the virtual camera. This results in a 2D like view in the upper part and a 3D perspective view in the lower part.

In our implementation, the geometry's vertices are projected onto the curve depending on their distance to the virtual camera. The new position is obtained by transforming the vertex according to the curve. Combined with geometric landmark enhancement, we can see how the enlarged landmarks add to their legibility (Figure 4.15):

- In the usual 3D perspective view, enhanced landmarks serve as distinct features and help to make the visualization more expressive.
- In the bird's eye view deformation, the landmarks enhance the horizon. Thus they facilitate orientation, as the user can compare the actual urban environment with the visualization. Highlighting the landmarks eases the alignment process.
- In the pedestrian's view deformation the effect is more visible. As the deformation resolves occlusion, the enhanced landmarks are highly visible and can be used as reference points in the navigation.

A small user evaluation suggests a general preference for the pedestrian's view deformation over classical 2D and 3D views, and the bird's eye view deformation (Pasewaldt et al., 2011).

4.6 Discussion

Geometric landmark enhancement, depending on the current view parameters, successfully improves their visibility, as apparent from the applications. Consequently, it facilitates the task of finding landmarks that have an impact on navigation and exploration of 3DCMs. The interactive

(a) usual perspective

(b) usual perspective, enhanced landmarks

(c) bird's eye view deformation

(d) bird's eye view deformation, enhanced landmarks

(e) pedestrian's view deformation

(f) pedestrian's view deformation, enhanced landmarks

Figure 4.15: Geometric landmark enhancement improves visibility of landmark objects in usual perspective projection (b) and non-linear projection modes (d,f). The implementation is based on Nokia maps, all landmarks are handled uniformly with $d_{\text{start}} = 2000m$.

adaptation smoothly integrates the effect in the usual 3D visualization, thus enabling usage by all applications of 3DCMs. The test cases show that the rendering performance is sufficient for interactive visualization even of large scale models.

The technique has been successfully applied to different data sets and in different software environments as presented before. Others have used and extended the idea to enhance landmarks by rotating them towards to camera (Semmo et al., 2012).

4.6.1 Limitations and Future Work

The technical limitation of the small number of landmarks enhanced at the same time is due to the quadratic runtime complexity in the displacement algorithm. However, in practice this does not present a problem for several reasons. If too many landmarks were emphasized at the same time, the highlighting would lose its value, as the display would be cluttered with too many items. In addition, scaling landmarks are perceived as movement, which captures the user's attention. Therefore it should be used sparsely.

The weakest point of the presented technique is the displacement model, as it creates visible radial distortion and pushes landmark models across boundaries. The visualization could be enhanced considerably by better displacement handling where impenetrable surrounding geometry, e.g., roads, water areas, and vegetation areas, would be respected. However, the important problem is to maintain interactivity in the visualization. This prevents direct usage of classical map displacement techniques (Lonergan and Jones, 2001). Therefore, it would be interesting to study approaches for real-time collision handling (Akenine-Möller et al., 2008) as used in 3D computer games. Another idea is to store boundaries and constraints in a distance field texture which is used to guide the displacement process later. Thus, landmark models could be "caged" to stay within their block.

Another drawback is that scaling the landmark objects to a constant size partly impedes the impression of the an interactive 3D environment: the model is not increasing in size when approaching and decreasing in size when moving away. To maintain this characteristic property of a perspective projection, the scaling function could be adapted slightly: a small linear scaling of the projected size could be applied that adds a distance-dependent term. While landmark models would still be enlarged to a visible size, they would not appear to have constant projected size.

Handling of a 3D terrain model including a surface texture is left for future work. In a straightforward integration one could directly deform the vertices of the terrain geometry by the radial displacement similar as the residential building geometry. It is not clear though, if the terrain shape remains legible.

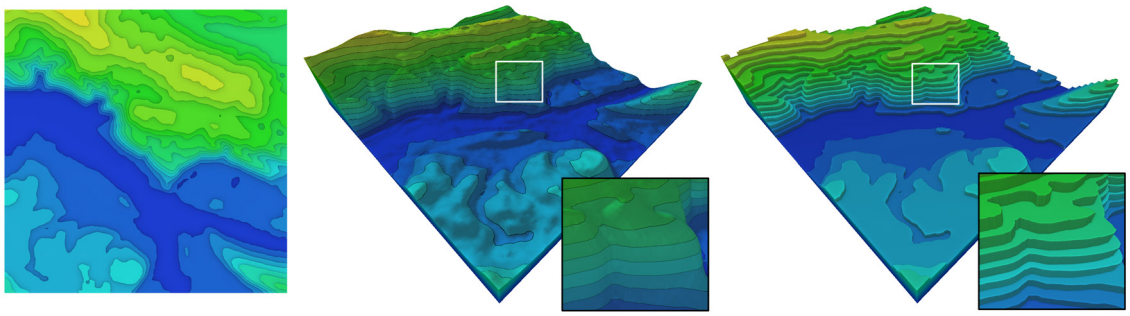


Figure 5.1: Traditional isoline / isopleth visualization (left), continuous 3D terrain surface visualization with draped isolines (center), the proposed stepped 3D terrain visualization (right). The definitive version of these images can be found at (Glander et al., 2010b), ©Eurographics.

CHAPTER 5

3D Isocontour Terrain Visualization

In this chapter we present *3D isocontours*, a real-time terrain rendering technique to create abstract 3D terrain visualization (Glander et al., 2010b). The presented approach can be integrated seamlessly with standard terrain rendering techniques by adapting the terrain geometry on a triangle-level at runtime.

5.1 Potentials and Usage of 3D Isocontour Terrain Visualization

Digital elevation models (DEMs) are central artifacts in geovisualization. They represent the geometry of a terrain's surface in high detail. As such they are used as a base model for arbitrary applications in the geospatial domain. Among its use cases are visualization, processing, and simulation, e.g., computing water delineation, erosion models, and flooding scenarios. In visualization the geometric terrain model is usually enhanced with textures from additional raster data-based surface layers, e.g., aerial color or infra-red imagery, meteorological measurements, and results from various computation. Also 2D and 3D vector data can be placed on terrain models, e.g., an infrastructure network, building, and vegetation models. Hence, digital elevation models are used both to study the terrain's geometric structure itself and as fundamental models that give spatial context to additional data.

Generalization in the case of terrain models commonly focuses on the adaptation of geometric detail to make computational processing more efficient. Through simplification the terrain model is transformed into a representation containing less geometric detail while staying as close as possible to the original terrain model. However, if the focus is on effective communication, photo-realistic visualization is not necessarily the ultimate goal (Döllner, 2007). Therefore the question is: *how to visualize a terrain abstractly?*

Increasing the abstractness of visualization is frequently desired for different use cases:

- Educational visualization aims at emphasizing specific properties of the terrain while deemphasizing others to facilitate understanding of the depicted geomorphology. For example topographic maps in atlases at a country scale enable quick identification of landforms, e.g., valleys, plateaus, plains, and mountains, by showing a small number of color coded height

levels. In mountain cartography curvature properties of the terrain surface are highlighted and exaggerated, e.g., summits, ridges, and passes, using hatching techniques (Imhof, 2007).

- Thematic visualization benefits from de-cluttered and reduced terrain models as the user is distracted less, but guided towards the thematic data, e.g., temperature, election results, and population density. The main purpose in this case is to provide spatial context and reference for the thematic data. Abstraction is achieved, e.g., by using less color or displaying only few outlines of the terrain features.
- Landscape architecture depends on abstract visualization to communicate inherent properties of the displayed architectural plan, e.g., its tentative nature, imprecision, and general "sketchiness". Abstract visualization therefore helps to keep the geodesign process focused and prevent premature discussion of peripheral details. Abstraction needs to be an integral part of geodesign tools (Ervin, 2011).
- In information visualization arbitrary real-valued data defined over a 2D domain is shown using the terrain metaphor. Using terrain visualization techniques, which users have learned to interpret, data that may not have a natural visual representation becomes more tangible. This facilitates forming and testing hypotheses visually (Ware, 2004).

In this chapter we introduce 3D isocontour terrain visualization, a real-time rendering technique to create an abstract, stepped terrain representation. It is based on classical isoline / isocontour or isopleth visualizations. Isolines are closed contours that are commonly used on maps to visualize terrain height, especially in mountain cartography (Imhof, 2007). Along an isoline, the height remains constant (*iso-*: greek prefix for equal). The height gradient can be observed in the perpendicular direction to the isoline. Such a map assumes a fixed set of equidistant isovalues; the continuous range of height values is quantized and represented by a few discrete values. Map users learn to interpret the depicted shapes, such as concentric features at local maxima (hills, summits) and minima (hollows, canyons), and to derive an understanding of the 3D surface (Ware, 2004). In the following, we use the terms isolines and isocontours interchangeably.

We propose to complement this traditional visualization style by *3D isocontours*, a 3D layered or stepped terrain visualization, in which quantization is applied to the geometry instead of color (Figure 5.1). This stepped terrain visualization can be obtained by lifting and vertically extruding isolines according to their particular isovalue, creating a stair-like structure. In cartography this kind of visualization is typically connected with Tanaka's relief map visualization, though it had been used before (Tanaka, 1950; Imhof, 2007).

The presented technique creates an artificial terrain visualization and is therefore suited to communicate abstractness. It works on arbitrary triangle-based terrain models and enables real-time rendering. The technique makes the following main contributions:

- Stepped terrain geometry is created dynamically, supporting interactive changes, e.g., through time-variant data or changing intervals of the contours.
- The proposed interpolation schema enables smooth blending with the original data, allowing for focus+context visualization with soft-edged lenses or camera dependent visualization.
- The technique is entirely GPU-based and therefore runs transparently without changes to the standard terrain rendering based on height mapping.

5.2 Abstract Terrain Visualization and Isocontours

We identify several related areas: they include manual cartography, automated processing techniques, use of abstraction in 3D geovisualization, and information visualization.

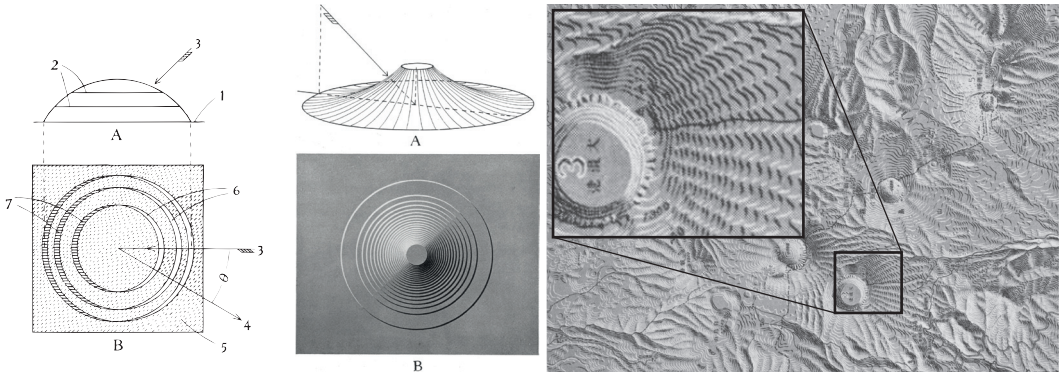


Figure 5.2: Tanaka decomposes the 3D shape into a series of slices with vertical walls seen from above and lighted from north-west. He studies the visualization on artificial shapes (left) and a Japanese landscape (right, figures from Tanaka (1950)).

5.2.1 Techniques from Manual Cartography

Visualization of terrains was studied extensively in cartography. Therefore, many techniques originate from manual map creation. One fundamental problem addressed by cartographers is how to depict the terrain's 3D surface on a static 2D medium. The techniques which originally were applied manually by cartographers have defined the visual grammar of cartographic terrain visualization.

Imhof (2007) presents an overview of the used techniques, among them are the following:

Hillshading is applied to represent the relief using lighting and shading. This visualization assumes a light source – typically from north-west – and the cartographer shades the surface depending on its orientation towards the light. Using the shading as depth cues the map users can reconstruct the 3D surface, mentally.

Hachures express the relief using strokes in direction of the slope. Surface properties, e.g., orientation, steepness, and light direction, are mapped to drawing styles, e.g. stroke density, thickness, and length.

Hypsometric tints are applied using a mapping from height to a color ramp to help distinguish different height levels above sea level. For visualization of underwater terrain, this color mapping is called bathymetric tinting.

Isolines are closed contours that represent equal heights along the contour. Usually index contours and intermediate contours are given. Index contours are drawn with a wider stroke and accompanied by text labels giving the specific heights. While being accurate, they require training to correctly interpret them.

The techniques are also used combined, e.g., hillshading with isolines provides a good general impression of the terrain surface together with exact elevation measures.

A specific techniqe is applied by Tanaka (1950). He proposes a layered relief model visualization by presenting the relief as a series of stacked terrain slices seen from above (Figure 5.2). He studies the effect of lighting as well as manual construction using differently colored inks and a filed drawing pen. His contour methods have also been automated using GIS technology (Kennelly, 2002). Exploring a less laborious alternative, Phillips (1979) creates a "wedding cake" effect for isocontours by redrawing all south-facing contours with a small vertical offset. While this is easier to construct compared to Tanaka's map, the shading of slopes is missing and the map reading performance is not improved.

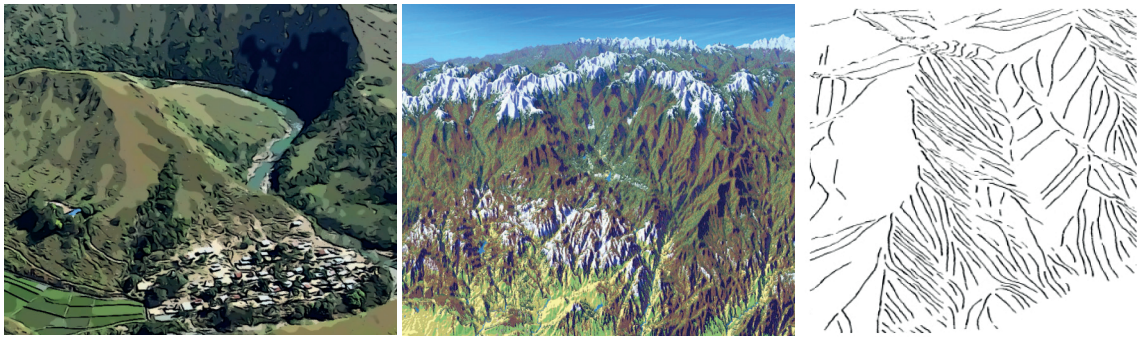


Figure 5.3: Abstraction can be applied to the surface appearance in different ways. The examples show an NPR effect applied (from left to right) (a) directly on the aerial photo (Semmo et al., 2010), (b) expressive panorama rendering using procedural textures (Bratkova and Shirley, 2009), and (c) hachures rendering based on terrain curvature and lighting (Buchin et al., 2004). All images are taken from the respective papers.

5.2.2 Abstraction in 3D Geovisualization

With the increased performance of computer systems, terrain models can be rendered in real-time, allowing interactive exploration of 3D terrain models. While the majority of terrain visualization techniques concentrate on photorealistic rendering of massive data models, e.g., (Dick et al., 2009; Pajarola and Gobbetti, 2007), there are also works that address abstraction. Non-photorealistic rendering (NPR) techniques in geovisualization aim at increasing the expressiveness with inspiration drawn from cartographic visualization techniques (Döllner, 2007).

As an example of an image-based NPR technique, Semmo et al. (2010) apply an NPR filter on aerial photos (Figure 5.3, left). The filter smoothes the image locally while enhancing edges for improved legibility. The technique is applied separately on different levels of the image pyramid to obtain scale adapted filtering. Another approach for expressive rendering is to use procedural stroke textures based on terrain surface classification (Bratkova and Shirley (2009), Figure 5.3, center).

Buchin et al. (2004) visualize the terrain curvature using hachures (Figure 5.3, right). They evaluate local surface properties of the terrain, e.g., slope direction, slope steepness, and surface orientation, and hierarchically compute slope lines using the gradient. At runtime, a selected subset of the slope lines are rendered depending on lighting. Also, surface properties and different lighting conditions can be mapped flexibly to visual variables, e.g., stroke width, density, and tone.

While these techniques change the surface appearance of the terrain, it is also interesting to change the surface geometry to obtain a simplified visualization. Leonowicz et al. (2010) argue that small-scale relief visualization require specific shading, as straightforward diffuse shading does not expose landforms and provides too many small details. In their proposed technique they apply a low-pass filter on the terrain grid geometry. Also, they detect ridge and valley landforms using surface coefficients. Two grids are created storing exaggerated valleys and ridges, respectively. Finally, the grids are blended depending on user defined weights and the diffuse shading is computed. Jenny et al. (2011) propose terrain generalization constrained by local curvature. In a preprocessing step, the terrain model is interpreted as an input signal and decomposed into frequency bands using a Laplacian pyramid. Ridge lines are identified with the help of curvature coefficients computed over the surface. At runtime, the terrain is reconstructed depending on virtual camera distance using the Laplacian pyramid. Using the curvature coefficients as weights, high-frequency details can be added locally at ridge lines, while generally presenting a smoothed and simplified surface geometry.

5.2.3 Automated Processing of Isolines

With the shift to digital acquisition, processing, and visualization of geodata, terrain models predominantly became available in grid or TIN representations. From these, isolines can be

extracted and processed using automated techniques. Among the specific challenges are the efficient extraction of isolines from huge terrain datasets and their generalization for increased legibility.

Existing approaches for isoline extraction seek to be efficient by precomputing index structures. Seed cells, which are are guaranteed to intersect the isoline, or seed points, which are lying on an isoline, serve as starting points for contour propagation (Carr et al., 2003; van Kreveld et al., 1997). To process massive data sets, these algorithms need to be I/O efficient (Silva et al., 2002). The inverse process, i.e., extracting grid- or TIN-based digital elevation models from given isolines, addresses the need to digitize and reuse historical printed maps (Chai et al., 1998; Rognant et al., 2001).

Isolines are also subject to map generalization. Common line simplification algorithms (Visvalingam and Williamson, 1995) are candidates, but special care is needed to generalize all isolines as a whole. Especially in areas with dense isolines, it is a requirement that the generalized lines maintain the topology. In addition, based on the steepness, isolines eventually have to be omitted from the resulting map to ensure legibility. Mackaness and Steven (2006) present an approach to remove isolines in steep areas. First they smooth the underlying terrain model from which the isolines are derived. Then, using a mask depending on the local steepness, intermediate isolines are removed while index contours are always preserved.

Isoline visualization in the case of 3D terrain rendering can be obtained in real-time using image-based techniques, which do not result in vector data but colored pixels at the isolines. The straightforward approach, to apply color to the terrain surface depending on the height, has problems at plateaus coinciding with an isovalue. Zhuo et al. (2009) avoids this by applying a two-pass rendering technique. In the first pass a quantized elevation gradient map is rendered to a buffer, which is used in the second pass to perform edge detection. The edges are finally superimposed on the terrain surface as isocontours.

5.2.4 Information Visualization

In information visualization, 2D data is often presented using a terrain metaphor (Schroeder et al., 2006), either by mapping the variable of interest to color (2D) or height (3D). Here, isocontours reveal classes in the data. A typical application is dot spatialization, e.g., in visualization of document collections: to show clusters in the data, a density field is computed and presented as a terrain surface.

Isocontours are extended by van Wijk and Telea (2001) using a non-linear mapping of the variable of interest to a height field and shading it. They argue that standard isocontours need to be combined with color tinting and shading to facilitate comprehension. The resulting "cushion look" improves the identification and comparison of maxima and minima in the data.

5.3 3D Isocontours

The goal of 3D isocontours is to create a layered relief representation known from physical architectural models and related to cartographic terrain visualization. In contrast to existing methods the additional geometry to represent 3D steps has to be created in real-time to support interactive change of isovalues intervals and input terrain data. Therefore, we choose to implement this technique on the graphics hardware (GPU), where the geometry data is processed in parallel and without depending from the underlying terrain representation, e.g., TIN or grid.

Processing on the GPU requires the algorithm to be designed using shader programs (shaders), which are executed on a vertex, primitive, and fragment level (Akenine-Möller et al., 2008). This prevents the use of contour tracing techniques as would be possible on the CPU, e.g., (Carr et al.,

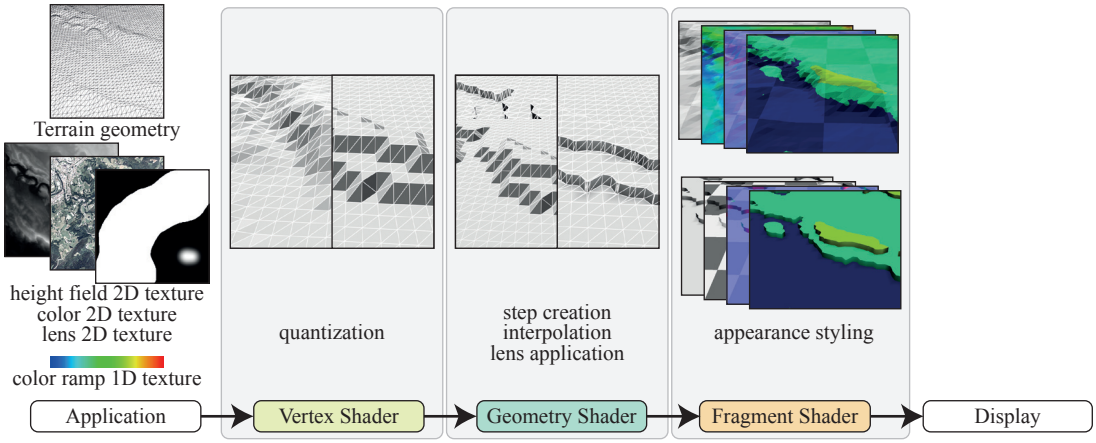


Figure 5.4: The proposed technique is implemented on the GPU and requires no changes to the way the data model is handled on the application side. Using vertex, geometry, and fragment shaders the input geometry is adapted to create a styled 3D isocontour terrain visualization.

2003); GPU processing requires vertices, primitives, and fragments to be processed independently and in parallel.

Our technique uses a shader at vertex level to adapt vertex positions to the nearest particular isovalue. In a shader at primitive level each triangle intersecting an isoplane is tesselated into geometry consisting of vertical or horizontal triangles to form a stair structure with sharp edges. The resulting geometry can be blended with the original geometry according to a given factor, allowing transitional states. At fragment level, we apply shading and color to the resulting surface to yield different styles.

5.3.1 Preliminaries

Formally, input data commonly used in terrain visualization can be defined as a scalar function $f(x, y) = z$ that maps continuous coordinates $(x, y) \in \mathbb{R}^2$ to a continuous value $z \in \mathbb{R}$. *Isovalues* in this context are a discrete, in general uniformly distributed, and small set of real values from an index set $\alpha \in \mathcal{I} \subset \mathbb{R}$. They implicitly define 2D *isolines* as

$$g(f(x, y), \mathcal{I}) = \{(x, y) : f(x, y) \in \mathcal{I}\}.$$

2D plots of $f(x, y)$ map values of z to a color, and display isolines in a distinctive color. For presentation, the color between isolines is often also quantized according to the isolines.

The smooth 3D surface is described in $\mathbb{R}^3$ by all points $P = (x, y, z)$ that satisfy f. Accordingly, isolines embedded in 3D are given by

$$h(f(x, y), \mathcal{I}) = \{(x, y, z) \in \mathbb{R}^3 : f(x, y) \in \mathcal{I}\}.$$

In practice, different digital terrain representations are used which approximate the continuous surface by discrete samples. The CityGML standard provides the following classes of terrain representations (Gröger et al., 2008):

Grid models sample the surface at uniformly distributed points. They are easy to handle but have the same resolution all over the terrain.

TIN based models (triangular irregular networks) can adapt better to local details but add some overhead.

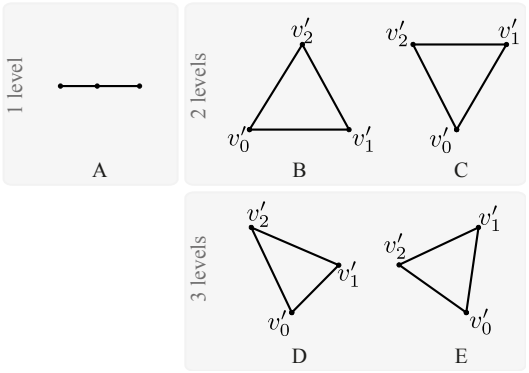


Figure 5.5: The basic triangle types are differentiated by the number of different isolevels they lie on.

basic type	sequence	initialization
A	-	-
B	012	$s_l = (0,2), s_r = (1,2)$
	120	$s_l = (1,0), s_r = (2,0)$
	201	$s_l = (2,1), s_r = (0,1)$
C,D,E	012	$s_l = (0,2), s_r = (0,1)$
	120	$s_l = (1,0), s_r = (1,2)$
	201	$s_l = (2,1), s_r = (2,0)$

Table 5.1: Depending on the triangle configuration we initialize helper segments.

Mass points describe the terrain by a set of measured points, either from land surveying or photogrammetry. For rendering or further processing, the surface between the points usually needs to be reconstructed.

Breaklines store the height along a line string. Usually they complement grid-based models to allow for sharp discontinuities independent from the grid resolution, e.g., at rivers or ridges.

For efficient rendering on the GPU, these representations are generally converted into triangles. As our techniques works on the GPU only, it is independent from the initial terrain representation.

In the following, we therefore assume the terrain to be given as triangles only. Also we assume that the input triangles are consistently oriented in counter-clockwise order. For 3D stepped terrain visualization, we have to subdivide triangles crossing isolines and create new geometry representing steps.

5.3.2 Creating Step Geometry

Without loss of generality, we assume terrain height to be normalized to $[0, 1]$. At vertex level, we quantize the height of each original vertex $v = (x, y, z)$ by transforming it to the quantized vertex $v' = (x, y, z')$ (Figure 5.6). The quantized height z' is the greatest isovalue α that is smaller than the vertex' height z:

$$z' = \alpha_j, \quad \alpha_j = \max(\{\alpha_i\}) \text{ with } \alpha_i < z.$$

The major part is done at primitive, i.e., triangle, level, where we generate vertical walls to approximate the isoline avoiding alignment to the original triangle structure.

At first, the configuration of the triangle is identified. Depending on the vertices' height distribution on isolevels, we distinguish five basic types A (one level), B, C (two levels), and D, E (three levels). We define a vertex sequence to start at the lower left corner of the triangle and store the indices to the vertices. Each basic type then has three possible sequences: 012, 120, and 201. Careful handling of the sequence is necessary to create consistently oriented sub-triangles. Summarizing, the configuration is defined by the basic configuration type A-E plus the vertex sequence 012,120, or 201.

Having the configuration identified, we can handle the different cases: the first basic type A is a triangle having all of its quantized vertices v'_i on the same layer. Nothing else needs to be changed, hence just the triangles' vertices are emitted without further subdivision.

More effort is necessary for the other cases B-E, where additional geometry is needed to form the steps. To guide the geometry construction, we compute a left and a right segment $s_l = [v_i, v_{i^-}]$, $s_r = [v_j, v_{j^+}]$ that describe the current part of the original triangle (Figure 5.6). The segments

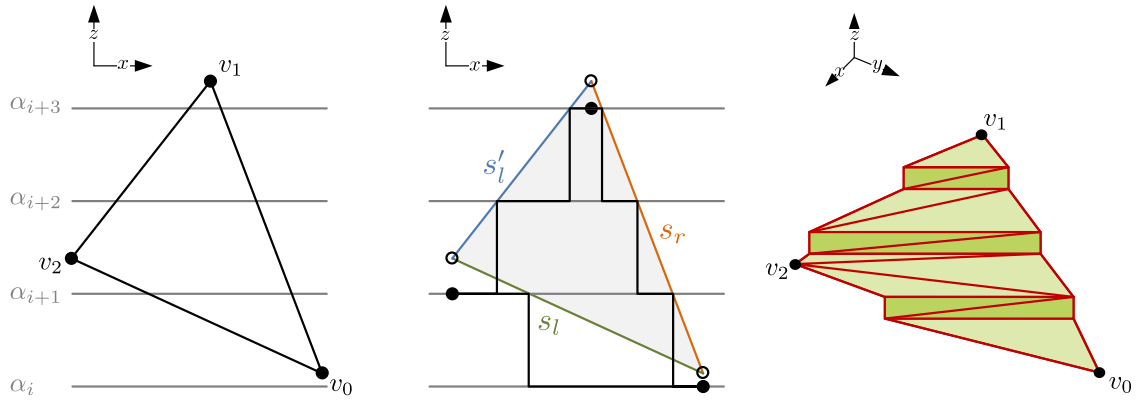


Figure 5.6: Left: input triangle spanning several quantization levels, Center: segments s_l and s_r describe the triangle stepwise. Right: tesselation forming step geometry.

are necessary to find the intersection with the isoplane and, thus, the position of the sub-triangles. Note that for the cases B and C the segments are valid for the complete triangle without change. In contrast, for the cases D and E, they need to be changed when approaching the middle vertex. In addition to the segments, we store the last step vertices a_l and a_r from the previous step.

The actual construction is done in a loop that increases height in steps of the quantization interval from zero until the maximum height is reached. The step segments are initialized depending on the triangle configuration (Table 5.1). The the vertices of the last step a_l and a_r are initialized with the original triangle's vertices. Note that in cases C,D, and E, a_l and a_r are sharing the same vertex at the beginning. At each step, we do the following:

Calculate intersection positions: At points $p_l^{\alpha_{i+1}}$ and $p_r^{\alpha_{i+1}}$ the segments cross the next isovalue α_{i+1}. Their positions can be obtained by linearly interpolating each segment's boundaries using the isovalue height α_{i+1}:

$$p_l^{\alpha_{i+1}} := (1-\beta_l)v_i + \beta_l v_{i^-} \text{ with } \beta_l = \frac{\alpha_{i+1} - h(v_i)}{h(v_{i^-}) - h(v_i)}$$

$$p_r^{\alpha_{i+1}} := (1-\beta_r)v_j + \beta_r v_{j^+} \text{ with } \beta_r = \frac{\alpha_{i+1} - h(v_j)}{h(v_{j^+}) - h(v_j)}$$

Create geometry A step connects the previous step vertices a_l, a_r with the newly found vertices $b_l := p_l^{\alpha_{i+1}}, b_r := p_r^{\alpha_{i+1}}$. It consists of a horizontal quad with vertices $s_0 = a_l$, $s_1 = a_r$, s_3, s_2 and a vertical quad s_2, s_3, $s_5 = b_r$, $s_4 = b_l$. The concave vertices s_2, s_3 correspond to b_l, b_r with their height set to the current isolevel α_i.

The algorithm generates triangles from these (Figure 5.7) and sets color, normal, and texture attributes as appropriate, using β_l, β_r to interpolate the attribute values from the original vertices. Correct orientation is guaranteed by the differentiation of left and right segments.

Set previous intersection positions To prepare the next step, the identified intersection positions are set as the previous step vertices $a_l := b_l$ and $a_r := b_r$.

Special handling is needed for the configurations having vertices on three different isolevels, i.e., D and E (Figure 5.5). In this case, at the beginning of the loop body, a pseudo "finishing" triangle (a_l, a_r, v) must be created and either a_l or a_r replaced by v, depending on the segment v belongs to.

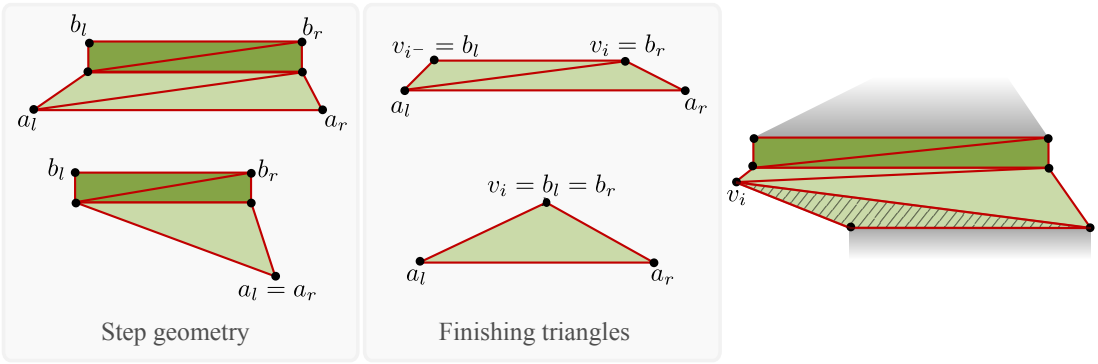


Figure 5.7: Left: step geometry consists of two quads with coordinates derived from intersections with the isoplanes. The degenerated case with $a_l = a_r$ occurs at the first iteration in configurations C, D, E. Center: finishing triangles connect the last step with the original vertex (/-ices). The degenerated case with $b_l = b_r$ occurs at the last iteration in configurations B, D, E. Right: special handling is necessary for the middle vertex v_i in configurations D and E. At first, a finishing triangle is created to connect v_i with the previous step vertices, then the usual step geometry is created. The definitive version of these images can be found at (Glander et al., 2010b), ©Eurographics.

The loop stops after maximum height has been reached. To complete the step geometry, finishing triangles are generated (Figure 5.5). In case of degenerated cases (B, D, E), just one triangle is necessary. The algorithm is given in Figure 5.8.

The tessellation creates step geometry based on input triangles and a desired quantization. If, additionally, a height field is given, the geometry can be smoothed by applying multi-sampling when reading vertex height from the height field in the vertex shader. Also, newly calculated normals at the steps can be smoothed by specifying the same normals at vertices of neighboring triangles, using local gradients in the height field.

5.3.3 Interpolation Schema for Smooth Transitions

Some applications require the ability to generate intermediate states of the stepped terrain visualization, e.g., for smooth activation of the effect or a soft lens application. Therefore, it is desirable to have an interpolation schema that allows for specifying the degree of the effect in an interval $\gamma \in [0, 1]$.

To implement this, we have to linearly blend the quantized heights with the original height $v.z = (1 - \gamma) \cdot v.z + \gamma \cdot h_q(v)$. For the step geometry, heights of the vertices on the isoline s_2, s_3 have to be blended with the intersection height at isolevel α_{i+1}.

To control the interpolation effect, γ can be given in a three alternatives: it can be specified globally, applied depending on camera distance, or depending on the location, i.e., using a gray-scale texture. The resulting visualization blends smoothly between the different representations of the terrain: 3D isocontours and the smooth terrain surface (Figure 5.9).

5.3.4 Surface Appearance Styling

After the primitive level processing, the geometry is rasterized and the fragments are processed by the fragment shader. We can compute the color of the output fragment based on all input data to obtain customized surface styling. The options include:

- to apply color from a given color ramp based on the quantized height,
- to use the surface normals to compute shading,
- to apply a color surface texture, e.g., an aerial photo,

```
// Tessellation function taking the original and the quantized vertices as input
tessellateTriangle(in vec4 v[3],  in vec4 v_q[3]){
   // config contains the vertex sequence in .xyz and the configuration in .w
   ivec4 config = identifyConfiguration( v_q);
   // store only indices to the vertices
   ivec2 left = ivec2(-1,-1);   ivec2 right = ivec2(-1,-1);

   if (config.w == 0){          // configuration A
      // one flat triangle, emit now
      emitTriangle(v_q[0], v_q[1], v_q[2]);
      return;
   }else if (config.w == 1){    // configuration B
      left = config.xz;
      right = config.yz;
   }else{                  // configuration C-E
       left = config.xz;
       right = config.xy;
    }
   // initialize previous step vertices left and right
   vec4 a[2];   a[0] = v_q[left.x];     a[1] = v_q[right.x];
   vec4 b[2];              // current step vertices left and right
   int vertexCount = 0;         // break loop, if vertexCount == 3

   for (int i = 0, iEnd = numQuantizations; i<iEnd; i++){
      float isoValue = i / float(numQuantizations);             // αi
      float isoValuePlusOne = (i+1) / float(numQuantizations);  // αi+1
      float intersectionHeight = isoValuePlusOne;

      if (v_q[config.x].z > isoValue) continue;     // not yet at lowest vertex

      // handle middle vertex at D type
      if (config.w == 3 && v_q[config.y].z == isoValue){
         // emit one triangle, adapt right
         emitTriangle(a[0], a[1], v_q[config.y]);
         // adapt right
         right = config.yz;
         a[1] = v_q[right.x];}
     // handle middle vertex at E type
     else if (config.w == 4 && v_q[config.z].z == isoValue){
         // one triangle, adapt left
         emitTriangle(a[0], a[1], v_q[config.z]);
         // adapt left
         left = config.zy;
         a[0] = v_q[left.x];}

      // b is on the level of the next threshold
      // compute horizontal intersection position
      b[0] = vec4(0.0,0.0,isoValuePlusOne,1.0);
      b[0].xy = getPos(v[left[0]], v[left[1]], intersectionHeight);
      b[1] = vec4(0.0,0.0,isoValuePlusOne, 1.0);
      b[1].xy = getPos(v[right[0]], v[right[1]], intersectionHeight);
      // emit triangles for a single step
      triangulateSingleStep(a[0], a[1], b[0], b[1]);
      // store b in a
      a[0] = b[0];
      a[1] = b[1];
      // no more steps, go out of loop
      if (max(v_q[config.y].z,v_q[config.z].z) <= isoValuePlusOne) break;
   }
   // finish with one or two finishing triangles
   emitTriangle(a[0], a[1], v_q[right[1]]);
   if (config.w ==2){ // C is the only configuration with 2 triangles finish
         emitTriangle(a[0], v_q[right[1], v_q[left[1]]);
   }
}
```

Figure 5.8: Geometry shader code in OpenGL Shading Language (GLSL) for tessellating the triangles. Handling of color, normals, texture coordinates, and lenses have been omitted for better readability.

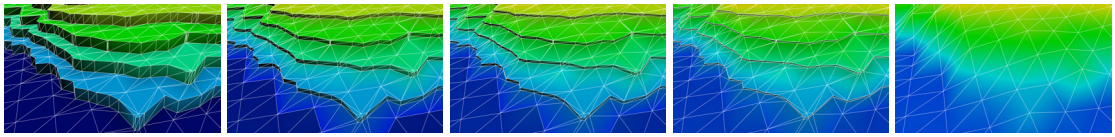


Figure 5.9: With the interpolation schema, smooth transitions between 3D isocontour and smooth terrain surface visualization can be obtained. The image sequence shows the result for $\gamma = 0, \gamma = 0.25, \gamma = 0.5, \gamma = 0.75, \gamma = 1$; from left to right.

- to mix the style based on a lens texture or surface properties, e.g., height or local orientation.

5.4 Results and Application Examples

As a proof of concept, 3D isocontours are implemented using a rendering middleware. We use the VRS and OSG to abstract from the low level rendering commands.

We tested our technique on an Intel Core2 Duo (3GHz) with 4 GB memory and a nVidia Geforce 9800 GT graphics card having 512 MB VRAM. Our performance measurements show interactive rates for small data sets, dropping below 6 FPS for 256x256 sized terrain models (Table 5.2). They show that the technique is limited by the size of the input model and the number of quantization steps. In our tests, screen resolution had no significant impact on the performance.

#steps	64x64	128x128	256x256
5	47.42	22.25	5.08
10	27.26	14.04	2.98
15	24.65	11.51	2.32
20	21.25	11.07	1.87

Table 5.2: Frames-per-seconds, measured at a screen resolution 1024x768, are given for different terrain grid sizes and different configurations.

Generally the visual quality is high, with isocontours being traced smoothly. However, if a high number of isolevels are used and many steps have to be created per input triangle, artifacts start to occur. At steep areas with a high density of isocontours, many additional triangles need to be created. At worst, for each input triangle, the number of output triangles is four times the number of quantization intervals. Due to a hardware dependent limitation of geometry shader output to, e.g., 4096 Bytes, in some occasions not all the required triangles can be created. This leads to unpleasant artifacts as holes begin to appear in the terrain geometry, where triangles are missing

We demonstrate the stepped terrain visualization on different application examples that make use of the terrain metaphor. Datasets can be loaded and explored interactively in the applications, exposing, e.g., contour intervals, color schemes, and lighting to the user.

5.4.1 Terrain Visualization

Terrain data is most naturally shown as a physical or virtual 3D relief model, closely approximating the terrain. In printed atlases, the 3D relief is often visualized using isolines to capture the 3D surface and communicate height differences. These height differences are emphasized with explicit color discontinuities marking the isolines.

Using stepped terrain visualization, we transfer this concept to 3D, showing height differences as discontinuities in geometry. High-frequency data is eliminated to produce a clean, schematic visualization. Shadows introduced by sharp edges further highlight the course of isolines (Figure 5.10). In contrast to only applying color or draped isolines to the smooth terrain surface, 3D isocontours dramatically change the silhouette, emphasizing the horizon line. An important requirement for

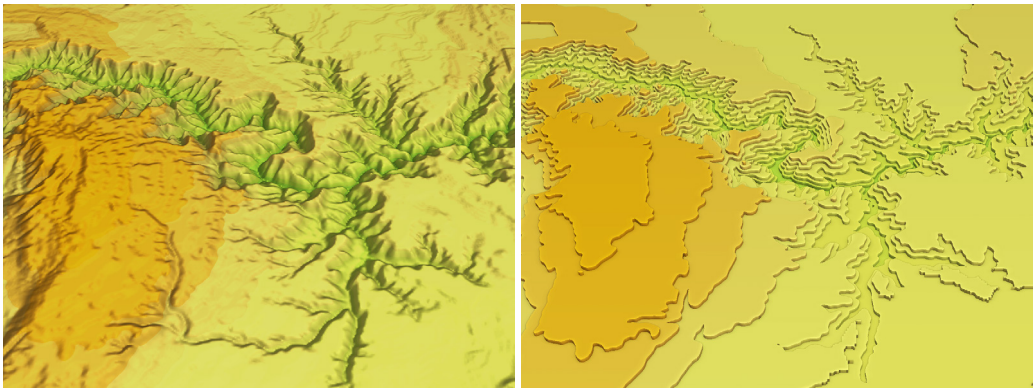


Figure 5.10: The Grand Canyon dataset shown with usual geometry and shading (left) and with 3D isoline visualization (right).

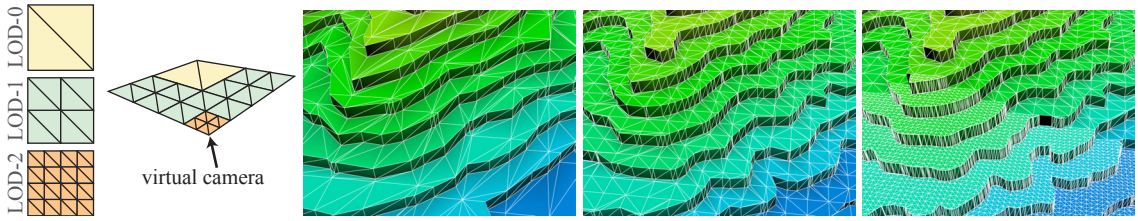


Figure 5.11: LOD rendering chooses the appropriate resolution from a precomputed multi-scale terrain representation. The superimposed wire-frame visualization shows the underlying resolution. A coarser resolution results in generalized isocontours. Transitions between neighboring tiles of different LODs are hardly noticeable (right).

terrain rendering is that massive datasets can be handled (Pajarola and Gobbetti, 2007). To support these, the terrain model is usually preprocessed into a hierarchical LOD representation. A common approach for grid-based terrain models is to compute a quad tree consisting of fixed-sized tiles on severel levels of detail, as implemented, e.g., by OSG: starting with a coarse top level grid tile representing the full terrain, the tile is divided into four sub-tiles. This is repeated until the resolution of the input data is matched by the tiles. At runtime, the tiles with the appropriate resolution are loaded and rendered.

We observe that LOD rendering is handled well by 3D isocontours. As the standard rendering pipeline is only altered on the GPU, the LOD mechanism, e.g., computing the screen error, loading and unloading the terrain tiles, and performing the draw calls, remains the same. In addition, 3D isocontours visualization is suitable for large preprocessed datasets. The additional step geometry is created at runtime and only for the tiles currently being rendered. It is apparent that the coarser terrain geometry also results in smoothed isocontours (Figure 5.11). Transitions between tiles of different LODs are hardly noticeable as geometric discontinuities of the isocontours.

The surface appearance can be styled in a number of ways (Figure 5.12):

- Shaded relief visualization presents the surface with shading applied depending on the local orientation towards the light source.
- Photorealistic relief visualization applies shading and photo-realistic textures.
- Hypsometric tints use a color ramp to emphasize height.
- 3D isocontours quantize height to render stepped geometry at equidistant isovalues. In addition color can be applied to emphasize the curvature characteristics.
- 3D isocontours without color can be blended with photo-realistic textures to give spatial context.

- 3D isocontours can be displayed black and white for high contrast. The visualization reminds of Tanaka's hand-drawn maps.

Technically the styles can be combined in arbitrary ways; although some care is needed to design meaningful and legible representations. For example blending the photo-realistic texture improves the communication of spatial references to the users. However, it also adds visual noise to the visualization, so a trade-off is involved.

5.4.2 Focus+Context Visualization

Lenses are used to show different data representations in the same place using a lens metaphor (Bier et al., 1993). Through the presented interpolation schema, we are able to blend between the original, smooth terrain surface and the 3D isocontour representation. The lens may have soft edges or be applied only gradually, i.e., the application of the lens does not have to be binary.

In our implementation, the effect is controlled by a normalized lens value $\gamma \in [0, 1]$. It is applied to each vertex, hence it may be defined differently over the terrain, e.g., based on local surface properties, from an additional lens texture, or based on virtual camera distance.

Using an additional lens texture offers artistic freedom to highlight a certain area in the terrain, e.g., a dominant ridge (Figure 5.13). As the 3D isocontours technique is applied in real-time, manipulation is also supported, either by interactively brushing the lens into the terrain visualization or by moving a lens, e.g., under the mouse cursor.

The lens can also be applied by evaluating local surface properties. In Figure 5.14 we evaluate the normalized height to derive γ. It is applied to smoothly interpolate between original, photo-textured geometry and stepped, color-tinted geometry. This reveals and emphasizes the mountains while maintaining the flat, residential areas which give context.

5.4.3 Information Visualization

Information visualization relies on the terrain metaphor to communicate abstract data. Among the challenges is the mapping of dimensions within the data to visual variables, e.g., height, color, or textures (Ware, 2004). In doing so, one must carefully select visual variables that communicate the scale of the data to be represented. For example data on a rational scale, i.e., allowing statements like "a is three times of b", is difficult to map to color. While color is – within limits – appropriate to represent interval scale data, i.e., "a is more than b", it is not a visual variable that supports a rational scale.

3D isocontours have a number of properties that are important for information visualization.

- 3D isocontours visualization maps the data to height, which – as a geometric visual variable – supports rational scaled data.

- Geometric isocontours, in contrast to draped isolines, introduce sharp edges which make them distinguishable through shadows and parallax effect.

- The quantization of the continuous scale introduces distinctive, countable features, which improve estimation of height. Also the resulting clear, angular shapes support Gestalt effects, i.e., the users can more easily recognize characteristic patterns.

In our application sketch, we visualize dynamic data through an animation. Dynamic data changes over time or an additional dimension, e.g., weather forecast, measured audio frequencies, 3D functions. To demonstrate our technique with dynamic data, we generate a series of raster images from a 3D noise function. At runtime, our application uses these images as height maps of an artificial terrain model. By flipping through the sequence of images, the impression of an

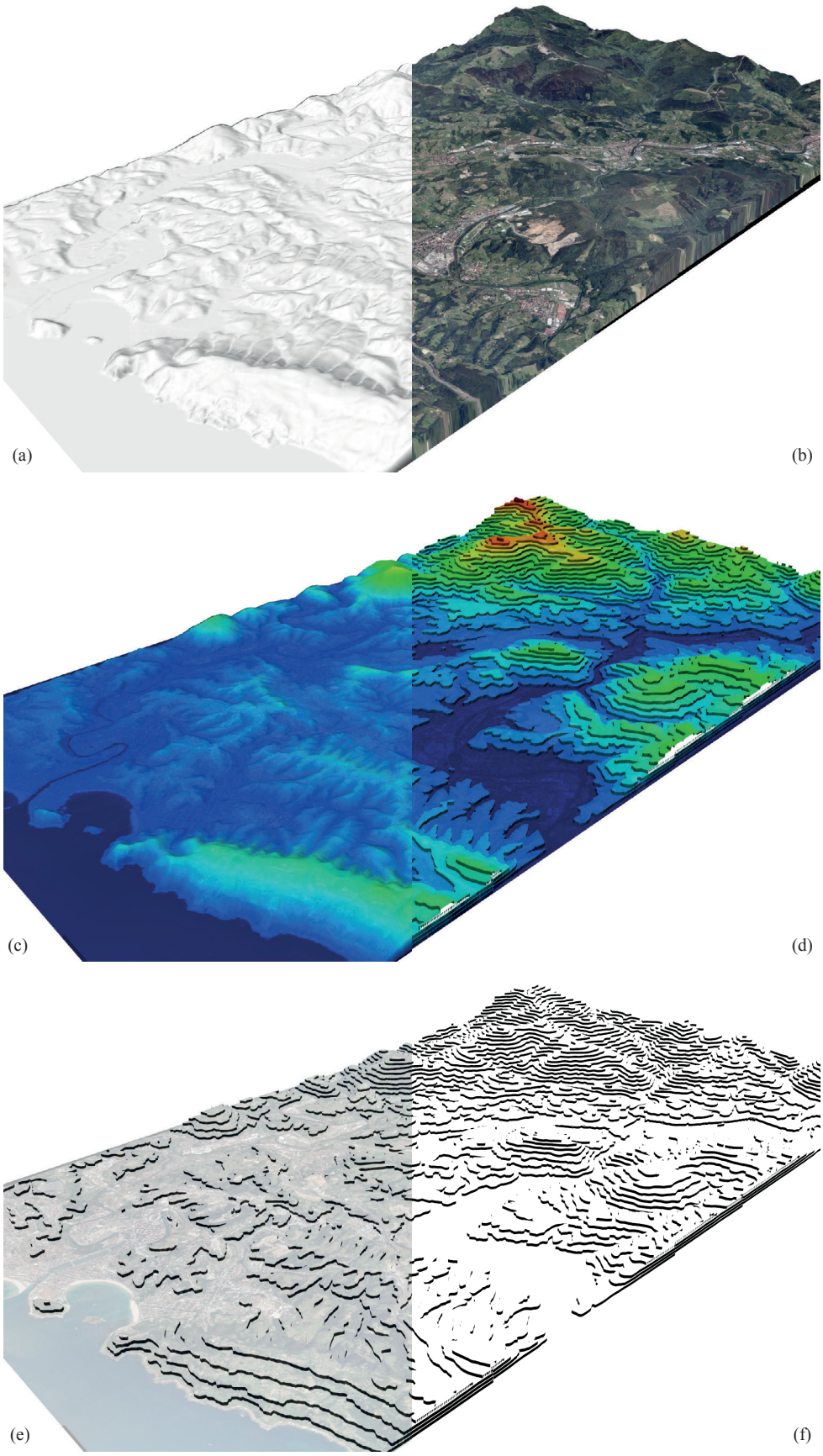


Figure 5.12: Surface appearance can be changed in a number of ways. The images show the Donostia dataset with shaded relief (a), photo-realistic relief (b), hypsometric tinting (c), tinted 3D isocontours (d), shaded 3D isocontours with blended texture (e), and black and white 3D isocontours (f).

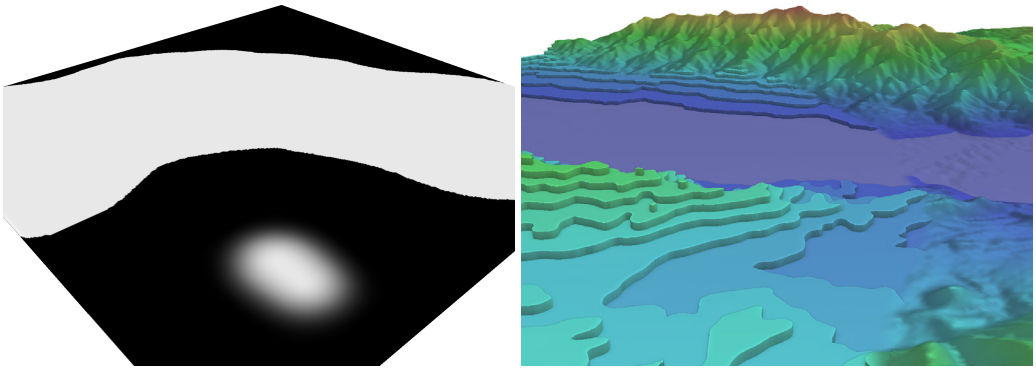


Figure 5.13: The lens reveals the ridge on the background while presenting layered steps on the left side and foreground. The definitive version of these images can be found at (Glander et al., 2010b), ©Eurographics.

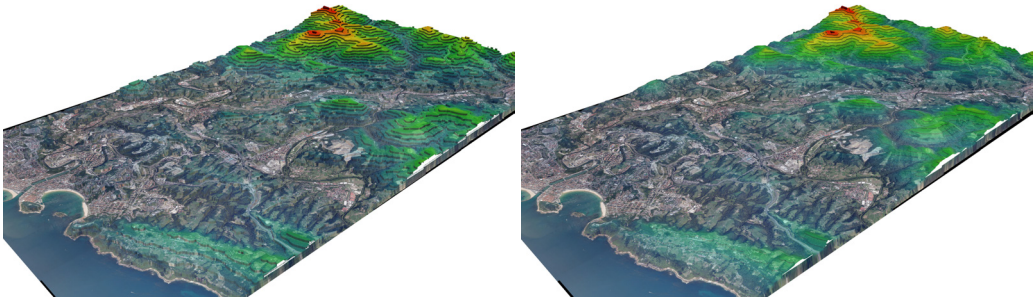


Figure 5.14: The Donostia dataset shown with height applied as lens value γ. Left: 3D isocontours visualization, γ applied to color only. Right: γ applied to blend geometry and color.

animated model is created. 3D isocontours are created in real-time, allowing the technique to be applied without further changes to create an animated stepped terrain visualization. In the rendering, emerging and collapsing peaks can be observed in quantized visualization.

As a second application sketch, we use 3D isocontours as an additional data layer on top of a smooth terrain surface: while the terrain is rendered as usual, the data of interest is added as geometric steps (Figure 5.16). This effect can be obtained with minimal changes to the geometry shader, assuming the terrain height is given as another texture: for each newly created triangle we need to read and add the additional height to the current vertex' z coordinate. In the pseudocode (Figure 5.8), this can be integrated in `emitTriangle`. Even though both data sets (terrain height, additional data layer) use height as a visual variable, they can be distinguished visually because of the 3D isocontours visualization.

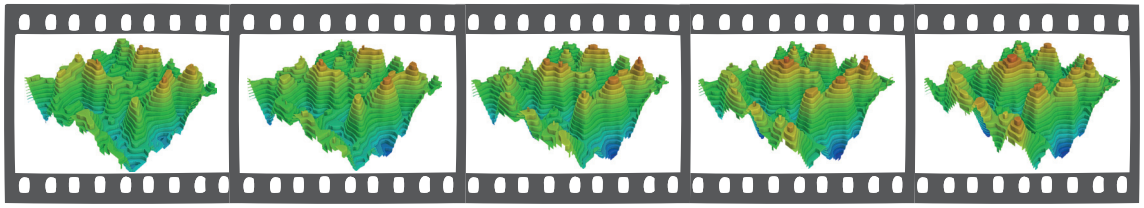


Figure 5.15: Flipping through a sequence of height images creates the impression of animation. The 3D isocontour geometry is created per frame.

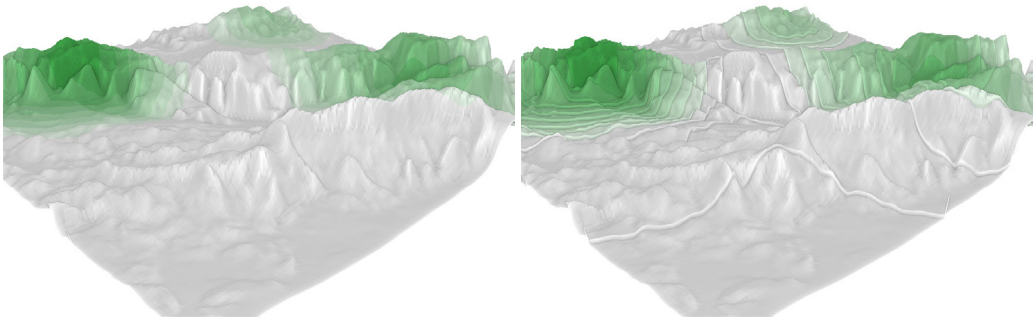


Figure 5.16: Left: the shaded terrain model is overlaid with a data layer (shades of green). Right: the data layer is additionally rendered as geometric steps on top of the shaded terrain surface.

5.5 Discussion

3D isocontours visualization presents the terrain models in abstract style, with reduced high-frequency details, and in 3D perspective. It complements 2D isocontours or hypsometry visualization, as it is relies on the more dramatic effect of 3D impression that matches our everyday experience. Using color as visual variable is problematic because of humans' difficulties to use it for rational scaled data and color vision deficiencies. In contrast, using 3D layered isocontours supports rational scale data. It also complements standard 3D terrain visualization, as it additionally communicates abstractness and enables easier comparison of height levels. 3D isocontours build on the users' experience with 2D isocontours visualization, but are also easier to understand as their appearance is closer to the original terrain. In addition, with the proposed interpolation schema, they can help to explain the relation between isocontours and the underlying 3D terrain surface.

The presented application examples have shown that the technique is flexible and applicable in different domains. Because it is GPU-centered the integration effort is reduced to applying the shader programs. The performance is efficient and – if combined with I/O efficient LOD strategies – sufficient for real-time rendering of large scale terrain models, as shown. Multiple LOD representations of the underlying terrain models also naturally enable the use of a smoothed terrain surface that results in generalized isocontours.

An alternative for the presented implementation would be to pre-process the terrain, which would also allow merging triangles on plateaus. Thus, static terrain models could be rendered more efficient. However, dynamic transition and blending effects would be more difficult. In addition, dynamic terrain models would require processing *and* upload to the graphics card to be performed for each frame.

A similar appearance could alternatively be obtained by a shading-based approach. However this would limited view points. Also the extra geometry providing distinct silhouettes would be missing.

5.5.1 Limitations and Future Work

The results also have demonstrated the limitations of the technique, which lead to directions for future work. As the major part of the implementation is done in the geometry shader stage, this is the bottleneck of the processing with direct impact on the performance. The more triangles to be created – because of a high number of quantization levels or a high resolution of the terrain model – the worse the performance. Others also have reported performance drops when excessively outputting triangles from a geometry shader, even if still within the hardware output limit (Lorenz, 2010). They suggest to dynamically instantiate sub-triangles from a small set of patterns instead of

generating them dynamically. More recently, hardware tessellation units on the GPU have been introduced, which promise to solve this problem generally. An implementation of the presented technique using hardware tessellation would therefore be interesting.

Writing out the created geometry using transform feedback (Carter et al., 2011) and reusing it in subsequent frames would also increase the performance for static geometry. In addition this would open a possibility to do additional processing, e.g., smoothing.

Further exploration of the vertical walls of 3D isocontours is another interesting field: they represent areas suitable for labeling, which would be an important addition. Also, using the interpolation schema, their height could be varied. In cartography isocontours represent a means to give exact heights, and index contours are distinguished from intermediate contours. An idea would be to apply altitude labels on the index contours while rendering intermediate contours blended stronger with the original terrain surface. This can also be dependent on virtual camera distance to get a smooth generalization effect: from further away, only index contours could be shown.

Virtual globes are a popular terrain rendering alternative. A generalization of 3D isocontours to support virtual globes would therefore be interesting. The current implementation of 3D isocontours relies on the definition of a static "up"-dimension, which is the z-axis. All processing and quantization steps are computed with respect to this dimension. Support for virtual globes would require the up-direction to be define as a vector, e.g., per vertex of the terrain.

Independently from the implementation, a user study is important to evaluate how users interpret 3D isocontours. Earlier studies suggest that 2D isocontours are hard to interpret correctly by unexperienced users, even if enhanced with some Tanaka-like shading (Phillips, 1979). The study could compare the estimation of height different terrain representations (3D isocontours, tinted 3D terrain, 2D tinted terrain). This would evaluate the hypothesis, that 3D isocontours improve interpretation of rational scaled data over color tinted visualization.

CHAPTER 6

Summary and Outlook

Interactive real-time visualization of 3DCMs drives a variety of applications. Abstract representations of 3DCMs provide these applications with a comprehensive visualization of the geospatial environments, helping their users to a more focused information perception. Classical cartography provides a visual language that allows the abstract visualization techniques to be easily interpreted. The concluding chapter summarizes the presented concepts and techniques, and revisits them with respect to the initial research objectives.

6.1 Summary

The central topics of this thesis are concepts and techniques that process and visualize models of our geospatial environment, i.e., 3DCMs and landscape models. Just like the environment itself, these models in general are heterogeneous, complex, and large. In 3D geovisualization we often present them by photo-realistic, perspective, and interactive images which correspond to our natural perception and facilitate intuitive interpretation. However, effective communication of large 3DCMs requires management of the presented details. Multi-scale representation of 3DCMs are introduced in this thesis, which represent 3DCMs in different levels of abstraction.

A key observation is that the abstraction of 3DCMs and landscape models requires computer-human-communication to be addressed. Its primary goal is not to reduce details to make computational processing more efficient. Instead it draws its motivation from being human-centered, seeking to make visualization more effective. Therefore abstraction techniques are fundamental elements of a content creation pipeline for 3DCMs that aims at improving visual communication.

We review the presented techniques with regard to the initial research questions, i.e., how to derive abstract multi-scale representations of 3DCMs, and how to visualize them (Section 1.3):

Cell-based generalization technique is a preprocessing technique inspired by tourism maps, which is suitable for small scale visualization of a city. It creates static representations of a given 3DCM in different scales based on a parceling of the 3DCM's plane using its infrastructure network. Dense urban areas are visualized by abstract building blocks, while landmark buildings are maintained. The utility of cell-based generalization is demonstrated by applications in research of continuous generalization, focus+context visualization, and non-linear projections, as well as in an industrial project.

Addressing the first research question, the cell-based generalization represents a methodology to create abstract representations of 3DCMs in multiple scales. Information density is greatly reduced as individual residential building models are hidden by block models, which makes the technique suitable for large ranges of scale. The methodology supports the creation of hierarchical relations, both for abstract building blocks and landmark models. Through parameters, the outcome of the automatic generalization process can be adapted, however, the input infrastructure network determines the result to a large extent.

While the focus of cell-based generalization is on preprocessing, we also demonstrate applications that are related to the second research question. We present a dynamic adaptation of the LOA as well as dynamic application of focus+context lenses, exploiting the hierarchical relations created. Furthermore, appearance styles applied to building, vegetation, and infrastructure features help to express abstractness more clearly.

Geometric landmark highlighting is a real-time visualization technique for 3DCMs that emphasizes landmark buildings interactively. Using geometric scaling and deformation, landmark buildings are enhanced depending on the virtual camera distance to ensure their visibility. We preprocess the landmark models and create a landmark hierarchy, deriving parameters that control the effect at runtime. Applications of this technique to both high detail and abstract 3DCMs as well as an industrial map application demonstrate its utility.

This technique creates an abstract visualization by emphasizing landmark buildings in their spatial context rather than reducing information density. Landmarks are perceived and used hierarchically; the technique takes this into account by computing a spatially balanced landmark hierarchy. The automatic preprocessing computes enhancement parameters for each landmark based on given initial weights. In addition, the computed enhancement parameters can be adapted and customized at runtime.

The second research question is addressed by the real-time implementation: the exaggeration factor is dynamically adapted with respect to the virtual camera distance and across large ranges of scale. The exaggeration resolves occlusion of landmark objects in oblique viewing angles. Abstractness is communicated in the visualization by the non-realistic exaggeration of the landmark building model in its context.

3D isocontour terrain rendering is a real-time visualization technique for abstract, non-photorealistic terrain visualization. It creates a stepped terrain representation that is related to classical isocontour visualizations. The GPU-based technique is complementary to the rendering pipeline of common terrain visualization techniques and representations, which enables its straightforward integration. The application examples demonstrate this with an integration using different rendering middleware systems. Furthermore the flexibility is shown with dynamic terrain data, focus+context lenses, and information visualization.

Regarding the first research question, abstract representations are created by this methodology during rendering without preprocessing. The information density is reduced by eliminating high frequency curvature. Hierarchical relations are discussed with respect to index and intermediate contours. As the technique is part of the rendering, the geometry creation is completely automated. All parameters set by the users are immediately applied to the geometry and surface appearance, e.g., number of isovalues, blending factor, and lens texture.

The second research question is directly connected, as creation and visualization are combined in this technique. Consequently, the exposed parameters can by controlled dynamically and smoothly by, e.g., time, virtual camera distance, and lens textures. Occlusion has to be resolved by user interaction. Abstractness is expressed both by non-photorealistic geometry and false colors.

The three presented techniques can be applied independently, in general. A combination is beneficial in case of cell-based generalization and landmark highlighting, as shown in an application example of Chapter 4: landmark buildings are preserved by cell-based generalization but hardly visible in small scales. Dynamically enhanced landmarks ensure their visibility in a wide scale interval. In contrast, 3D isocontour visualization is not immediately suitable for combination. The methodology of landmark enhancement could be applied in a focus+context scenario to highlight, e.g., an important summit or a ridge.

We conclude the summary with properties shared by all presented techniques. They all apply cartographic visualization means to 3DCMs and landscape models. From a cartographic perspective, they are techniques that create secondary models with special purpose. Successful communication requires shared symbols between sender and recipient. This is assured by building on the visual language and symbols from cartography, e.g., for residential areas, isolines. The techniques have been implemented in multiple rendering middleware systems and with multiple applications demonstrating their flexibility and utility.

6.2 Future Work

With the approaches presented in this thesis, geovisualization systems can be enhanced with abstract visualization of 3DCMs and landscape models. A specific class of geovisualization systems are mobile, hand-held devices, which are used in traffic situations and therefore depend on abstracted visualizations with reduced visual noise. In addition to using these devices as an output medium, their specific properties and capabilities could be exploited. For example, their built-in camera and location sensors would enable augmented reality applications by, e.g., showing enlarged landmarks or 3D isocontours superimposed on live video. Also trajectories through the city could be integrated over time to build a density map serving as a generalization lens for a 3DCM, hiding low importance areas.

In addition to using the techniques for direct visualization of 3DCMs and landscape models, their application in information visualization could be explored in more detail. As sketched in the application examples, abstract visualization is beneficial for thematic mapping scenarios, since it helps to focus on the thematic data while giving spatial reference. Beyond visualization of geodata, the potential of the presented techniques could be explored for different domains. Using 3DCMs as a metaphor, arbitrary data can be shown and visualized abstractly, e.g., using block cells or 3D isocontours to aggregate data.

The next step is a formal evaluation of the generalization techniques to judge their quality and enable quantitative comparison with other approaches. Generalization techniques motivated by increased processing efficiency can be evaluated using metrics, e.g., positional accuracy, minimized volume changes, and complexity measures. Computable measures for generalization techniques aimed at simplified visualization are subject to discussion and still need to be researched. Alternatives include a survey of experts in the field and a formal user test. For the survey, experts of the field need to be interviewed and confronted with different visualization examples, e.g., following the methodology of Häberling (2003). In a user test, different visualization configurations could be used and combined with a task, e.g., navigating within the virtual environment. User studies comparing 2D and 3D visualization of geospatial environments show better navigation performance with 2D maps (Oulasvirta et al., 2009). It would be interesting to see if abstract representations of 3DCMs can increase the performance with 3D visualization.

Another interesting direction of research is further combination of the presented techniques with interaction techniques. As the techniques create different representations of a 3DCM, the question is how to explore them? While some approaches are presented, e.g., distance-based selection, generalization camera lens, and isocontour lens based on distance, many others are interesting, e.g., lenses could be drawn directly onto the 3D scene, revealing underlying higher detail areas. Similarly, geometric exaggeration could be used as a visual thumbnail in conjunction with touch-based devices: while touching and hovering over the 3DCMs, the technique could be parameterized to highlight these areas for a quick preview without need to move the virtual camera.

Finally, we need a theoretical framework for 3D generalization. Generalization provides fundamental theory and methodology for the creation of 2D maps. In comparison, research in the field of 3D generalization is still recent. This is one reason that the notion of a "3D map" is blurry.

A theoretical framework for 3D generalization would include at least a taxonomy of elementary 3D generalization operators, formal requirements for 3D map elements, and a formalization of the effect of continuous scale. In consequence 3D map visualization systems could apply automated generalization based on constraints, as used today for 2D map production, while 3D generalization techniques so far are mostly rule-based. Constraints seem to be more effective to capture the multiple, often contradicting requirements of a map. However, a major difference of interactive 3D map visualization to classical maps is the need to support a wide range of continuous scales and viewpoints. Providing an optimized abstract visualization for each of them is constrained by the requirement of interactivity.

Abstract representations of 3DCMs and landscape models will continue to be an exciting field of research. As 3D visualization is strongly connected with technological advances, there is a constant input of emergent technologies, e.g., mobile devices, augmented reality, and social networks. How to utilize these for the benefit of increased comprehensibility will therefore remain a perpetual question.

Bibliography

T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering*. AK Peters, Ltd. Natick, Natick, MA, USA, 3rd Ed., 2008.

K. Anders. Level of Detail Generation of 3D Building Groups by Aggregation and Typification. In *Proc. of the Int. Cartographic Conference (ICC)*, pp. 11–16, La Coruña, Spain, 2005. International Cartographic Association (ICA).

M. Batty. Model Cities. Technical Report 113, UCL Centre for Advanced Spatial Analysis, London, UK, 2006.

K. M. Beard. Constraints on Rule Formation. In B. Buttenfield and R. Mcmaster, Editors, *Map Generalization: Making Decisions for Knowledge Representation*, pp. 121–135. Longman, London, UK, 1991.

J. Benner, K. Leinemann, and A. Ludwig. Übertragung von Geometrie und Semantik aus IFC-Gebäudemodellen in 3D-Stadtmodelle. *Proc. of 9th Intern. Symp. on Planning & IT, CORP*, pp. 573–578, 2004.

M. Berg, M. Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 2000.

E. Bier, M. Stone, K. Pier, W. Buxton, and T. DeRose. Toolglass and Magic Lenses: The See-Through Interface. In *Proc. of SIGGRAPH*, pp. 73–80, New York, NY, USA, 1993. ACM Press.

J. Bobrich. *Ein neuer Ansatz zur kartographischen Verdrängung auf der Grundlage eines mechanischen Federmodells*. Ph.D. Thesis, TU Darmstadt, 1996.

J. Böttger, M. Preiser, M. Balzer, and O. Deussen. Detail-In-Context Visualization for Satellite Imagery. *Computer*, 27(2), 2008.

K. E. Brassel and R. Weibel. A Review and Conceptional Framework of Automated Map Generalization. *International Journal of Geographical Information Systems*, 2(3):229–244, 1988.

M. Bratkova and P. Shirley. Artistic Rendering of Mountainous Terrain. *ACM Transactions on Graphics*, 28(4):1–17, 2009.

C. Brenner. Towards Fully Automatic Generation of City Models. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIII-B1-:85–92, 2000.

C. Brenner and B. Elias. Extracting Landmarks for Car Navigation Systems using Existing GIS Databases and Laser Scanning. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIV-3/W8:131–138, 2003.

C. A. Brewer and B. P. Buttenfield. Framing Guidelines for Multi-Scale Map Design Using Databases at Multiple Resolutions. *Cartography and Geographic Information Science*, 34(1): 3–15, 2007.

R. Brunet. Building Models For Spatial Analysis. In *Two Decades of l'Espace Geographique: An Anthology*, pp. 109–123. GIP Reclus, Montpellier, France, 1993.

H. Buchholz and J. Döllner. View-Dependent Rendering of Multiresolution Texture-Atlases. In *Proc. of IEEE Visualization*, pp. 215–222, Los Alamitos, CA, USA, 2005a. IEEE Computer Society Press.

H. Buchholz and J. Döllner. Visual Data Mining in Large-Scale 3D City Models. In *Proc. of Int. Conf. and Exhibition on Geographic Information*. CD Proceedings, 2005b.

K. Buchin, M. C. Sousa, J. Döllner, F. Samavati, and M. Walther. Illustrating Terrains using Direction of Slope and Lighting. In *Proc. of ICA Mountain Carthography Workshop*, pp. 259–269, Núria, Spain, 2004. International Cartographic Association (ICA).

C. Carneiro, E. Morello, C. Ratti, and F. Golay. Solar Radiation Over the Urban Texture: LIDAR Data and Image Processing Techniques for Environmental Analysis at City Scale. In J. Lee and S. Zlatanova, Editors, *3D Geo-Information Sciences*, pp. 319–340. Springer, 2009.

M. S. T. Carpendale. Considering Visual Variables as a Basis for Information Visualisation. Technical report, Computer Science, University of Calgary, Calgary, Canada, 2003.

M. S. T. Carpendale and D. Cowperthwaite. Distortion Viewing Techniques for 3-Dimensional Data. In *Proc. of the IEEE Symposium on Information Visualization (INFOVIS)*, pp. 46–53, Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

M. S. T. Carpendale, J. Ligh, and E. Pattison. Achieving Higher Magnification in Context. In *Proc. of ACM Symposium on User Interface Software and Technology (UIST), ACM Press, New York, NY, USA*, pp. 71–80, New York, NY, USA, 2004. ACM Press.

H. Carr, J. Snoeyink, and U. Axen. Computing Contour Trees in all Dimensions. *Computational Geometry*, 24(2):75–94, 2003.

N. Carter, C. Lao, J. Sandmel, C. Wooley, and A. Eddy. EXT_transform_feedback, `http://www.opengl.org/registry/specs/EXT/transform_feedback.txt` (accessed 1.7.2012) 2011.

A. Cecconi. *Integration of Cartographic Generalization and Multi-Scale Databases for Enhanced Web Mapping*. Ph.D. Thesis, Universität Zürich, 2003.

A. Cecconi and M. Galanda. Adaptive Zooming in Web Cartography. *Computer Graphics Forum*, 21(4):787–799, 2002.

A. Cecconi, R. Weibel, and M. Barrault. Improving Automated Generalization for On-Demand Web Mapping by Multiscale Databases. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIV-4, 2002.

J. Chai, T. Miyoshi, and E. Nakamae. Contour Interpolation and Surface Reconstruction of Smooth Terrain Models. In *Proc. of IEEE Visualization*, Vol. 98, pp. 27–33, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

R. Chang, G. Wessel, R. Kosara, E. Sauda, and W. Ribarsky. Legible Cities: Focus-Dependent Multi-Resolution Visualization of Urban Relationships. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1169–1175, 2007.

R. Chang, T. Butkiewicz, C. Ziemkiewicz, Z. Wartell, W. Ribarsky, and N. Pollard. Legible Simplification of Textured Urban Models. *Computer Graphics and Applications, IEEE*, 28(3): 27–36, 2008.

G. Chen, G. Esch, P. Wonka, P. Müller, and E. Zhang. Interactive Procedural Street Modeling. In *Proc. of SIGGRAPH*, New York, NY, USA, 2008. ACM Press.

V. Coors. Feature-Preserving Simplification in Web-based 3D-GIS. *Proc. of Int. Symp. on Smart Graphics*, 2001.

V. Coors. 3D-GIS in Networking Environments. *Computers, Environment and Urban Systems*, 27 (4):345–357, 2003.

V. Coors and A. Zipf. MONA 3D – Mobile Navigation Using 3D City Models. In *Proc. of Int. Symp. on LBS and Telecartography*, 2007.

A. Czerwinski, S. Sandmann, E. Stöcker-Meier, and L. Plümer. Sustainable SDI for EU Noise Mapping in NRW–Best Practice for INSPIRE. *International Journal for Spatial Data Infrastructure Research (IJSDIR)*, 1:90–111, 2007.

J. Dangermond. GeoDesign and GIS - Designing our Futures. In E. Buhmann, M. Pietsch, and E. Kretzler, Editors, *Proc. of Digital Landscape Architecture (DLA)*, pp. 502 – 514, Dessau, Germany, 2010. Wichmann-Verlag.

P. Degener and R. Wahl. Context Aware Terrain Visualization for Wayfinding and Navigation. *Computer Graphics Forum*, 27(7):1853–1860, 2008.

C. Dick, J. Kruger, and R. Westermann. GPU Ray-Casting for Scalable Terrain Rendering. In *Proc. of Eurographics*, pp. 43–50. The Eurographics Association, 2009.

J. Döllner. Non-Photorealistic 3D Geovisualization. In *Multimedia Cartography*, pp. 229–240. Springer, Berlin, Heidelberg, 2nd Ed., 2007.

J. Döllner and K. Hinrichs. The Virtual Rendering System - A Toolkit for Object-Oriented 3D Graphics. Technical Report 19/95-I, Angewandte Mathematik und Informatik, Universität Münster, 1995.

J. Döllner and M. Walther. Real-Time Expressive Rendering of City Models. In *Proc. of Int. Conf. on Information Visualization (IV)*, pp. 245–250, Los Alamitos, CA, USA, 2003. IEEE Computer Society Press.

J. Döllner, K. Baumann, and H. Buchholz. Virtual 3D City Models as Foundation of Complex Urban Information Spaces. In M. Schrenk, Editor, *Proc. of Int. Conference on Urban Planning and Spatial Development in the Information Society (REAL CORP)*, pp. 107–112, Vienna, Austria, 2006a.

J. Döllner, H. Buchholz, and H. Lorenz. Ambient Occlusion - ein Schritt zur realistischen Beleuchtung von 3D-Stadtmodellen. *GIS - Zeitschrift für Geoinformatik*, pp. 7–13, 2006b.

J. Döllner, T. H. Kolbe, F. Liecke, T. Sgouros, and K. Teichmann. The Virtual 3D City Model of Berlin - Managing, Integrating, and Communicating Complex Urban Information. In *Proc. of Urban Data Management Symposium (UDMS)*, Aalborg, Denmark, 2006c.

B. Elias and V. Paelke. User-Centered Design of Landmark Visualizations. In L. Meng, A. Zipf, and S. Winter, Editors, *Map-based Mobile Services*, pp. 33–56. Springer, 2008.

J. Engel, S. Pasewaldt, M. Trapp, and J. Döllner. An Immersive Visualization System for Virtual 3D City Models. In *Proc. of Int. Conf. on GeoInformatics*, Hongkong, China, 2012. IEEE GRSS.

S. Ervin. A System for GeoDesign. In E. Buhmann, S. Ervin, C. D. Tomlin, and M. Pietsch, Editors, *Teaching Landscape Architecture Preliminary Proceedings*, pp. 145–154, Dessau, Germany, 2011. Hochschule Anhalt.

H. Fan. *Integration of Time-Dependent Features Within 3D City Model*. Ph.D. Thesis, TU München, 2010.

H. Fan, L. Meng, and M. Jahnke. Generalization of 3D Buildings Modelled by CityGML. In M. Sester, L. Bernard, and V. Paelke, Editors, *Advances in GIScience*, Lecture Notes in Geoinformation and Cartography, pp. 387–405. Springer Berlin Heidelberg, 2009.

T. Foerster and J. Stoter. Establishing an OGC Web Processing Service for Generalization Processes. In *Proc. of ICA Workshop on Generalization and Multiple Representation*, Vancouver, WA, USA, 2006. International Cartographic Association (ICA).

T. Foerster and J. Stoter. Generalization Operators For Practice - A Survey at National Mapping Agencies. In *Proc. of ICA Workshop on Generalization and Multiple Representation*, Montpellier, France, 2008. International Cartographic Association (ICA).

J. Foley, A. Van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice in C*. Addison-Wesley Professional, 2nd Ed., 1995.

A. Forberg. Generalization of 3D Building Data Based on a Scale-Space Approach. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXV-B: 194–199, 2004.

A. Forberg. *Generalisierung dreidimensionaler Gebäudedaten auf der Basis von Maßstabsräumen*. Ph.D.Thesis, Universität der Bundeswehr München, 2005.

A. Forberg and H. Mayer. Generalization of 3D Building Data Based on Scale-Spaces. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIV-4: 225–230, 2002.

M. Garland and P. S. Heckbert. Surface Simplification Using Quadric Error Metrics. In *Proc. of SIGGRAPH*, pp. 209–216, New York, NY, USA, 1997. ACM Press.

T. Glander and J. Döllner. Cell-Based Generalization of 3D Building Groups with Outlier Management. In *Proc. of ACM Int. Symp. on Advances in Geographic Information Systems (ACMGIS)*. ACM Press, 2007.

T. Glander and J. Döllner. Abstract Representations for Interactive Visualization of Virtual 3D City Models. *Computers, Environment and Urban Systems*, 33(5):375–387, 2009.

T. Glander, M. Trapp, and J. Döllner. A Concept of Effective Landmark Depiction in Geovirtual 3D Environments by View-Dependent Deformation. In *Proc. of Int. Symp. on LBS and Telecartography*, 2007.

T. Glander, D. Peters, M. Trapp, and J. Döllner. 3D Wayfinding Choremes: A Cognitively Motivated Representation of Route Junctions in Virtual Environments. In M. Sester, L. Bernard, and V. Paelke, Editors, *Proc. of AGILE Int. Conference on GI Science*, Lecture Notes in Geoinformation and Cartography, pp. 407–427, Hanover, Germany, 2009. Springer.

T. Glander, J. Baresel, and J. Döllner. Überlegungen zum stufenlosen Übergang zwischen verschieden generalisierten 3D-Stadtmodellrepräsentationen. In *Proc. of 3-Ländertagung DGPF*, Vienna, Austria, 2010a. Deutsche Gesellschaft für Photogrammetrie und Fernerkundung (DGPF).

T. Glander, M. Trapp, and J. Döllner. 3D Isocontours - Real-time Generation and Visualization of 3D Stepped Terrain Models. In S. Seipel and H. Lensch, Editors, *Proc. of Eurographics (Shortpapers)*, pp. 17–20, Norrköping, Sweden, 2010b. The Eurographics Association.

T. Glander, M. Trapp, and J. Döllner. Concepts for Automatic Generalization of Virtual 3D Landscape Models. In *gis.SCIENCE*, Vol. 25, pp. 127–135, 2011.

M. Goodchild. Citizens as Sensors: the World of Volunteered Geography. *GeoJournal*, 69(4): 211–221, 2007.

F. Grabler, M. Agrawala, R. W. Sumner, and M. Pauly. Automatic Generation of Tourist Maps. *ACM Transactions on Graphics*, 27(3):100:1–100:11, 2008.

G. Gröger and L. Plümer. CityGML - Interoperable Semantical 3D City Models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33, 2012.

G. Gröger, T. H. Kolbe, A. Czerwinski, and C. Nagel. *City Geography Markup Language (CityGML) Encoding Standard*. Open Geospatial Consortium Inc., 2008.

R. Guercke and C. Brenner. A Framework for the Generalization of 3D City Models. In M. Sester, L. Bernard, and V. Paelke, Editors, *Proc. of AGILE Int. Conference on GI Science*, Hanover, Germany, 2009. Springer.

R. Guercke, J. Zhao, C. Brenner, and Q. Zhu. Generalization of Tiled Models with Curved Surfaces Using Typification. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-2:39–44, 2004.

R. Guercke, C. Brenner, and M. Sester. Generalization of Semantically Enriched 3D City Models. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-3-:28–34, 2009.

R. Guercke, T. Götzelmann, C. Brenner, and M. Sester. Aggregation of LoD 1 Building Models as an Optimization Problem. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(2): 209–222, 2011.

R. B. Haber and D. A. McNabb. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. *Visualization in Scientific Computing*, 74:74–93, 1990.

C. Häberling. *Topografische 3D-Karten: Thesen für kartografische Gestaltungsgrundsätze*. Ph.D. Thesis, ETH Zürich, 2003.

C. Häberling, H. Bär, and L. Hurni. Proposed Cartographic Design Principles for 3D Maps: A Contribution to an Extended Cartographic Theory. *Cartographica*, 43(3):175–188, 2008.

B. Hagedorn and J. Döllner. High-Level Web Service for 3D Building Information Visualization and Analysis. In *Proc. of ACM Int. Symp. on Advances in Geographic Information Systems (ACMGIS)*, pp. 1–8, New York, NY, USA, 2007. ACM Press.

G. Hake, G. Hake, D. Grünreich, and L. Meng. *Kartographie*. Walter de Gruyter, 8th Ed., 2002.

J. R. Herring, Editor. *OpenGIS Implementation Standard for Geographic Information - Simple Feature Access - Part 1: Common Architecture*. Open Geospatial Consortium Inc., 2011.

S. Hirtle and J. Jonides. Evidence of Hierarchies in Cognitive Maps. *Memory & Cognition*, 13(3): 208–217, 1985.

H. Hoppe. Progressive Meshes. In *Proc. of SIGGRAPH*, pp. 99–108. ACM Press New York, NY, USA, 1996.

L. Hu, P. V. Sander, and H. Hoppe. Parallel View-Dependent Refinement of Progressive Meshes. In *Proc. of Symp. on Interactive 3D Graphics and Games (I3D)*, pp. 169–176, New York, NY, USA, 2009. ACM Press.

E. Imhof. *Thematische Kartographie*. 10. de Gruyter, Berlin, Germany, 1st Ed., 1972.

E. Imhof. *Cartographic Relief Presentation*. ESRI Press, 2007.

International Cartographic Association. *Multilingual Dictionary of Technical Terms in Cartography*. Franz Steiner Verlag, Wiesbaden, Germany, 1973.

B. Jenny, H. Jenny, and L. Hurni. Terrain Generalization with Multi-scale Pyramids Constrained by Curvature. *Cartography and Geographic Information Science*, 38(2):110–116, 2011.

B. Jiang and X. Liu. Automatic Generation of the Axial Lines of Urban Environments to Capture What We Perceive. *International Journal of Geographical Information Science*, 24(4):545–558, 2010.

M. Kada. Automatic Generalisation of 3D Building Models. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIV-4, 2002.

M. Kada. 3D Building Generalisation. *Proc. of the Int. Cartographic Conference (ICC)*, 2005.

M. Kada. The 3D Berlin Project. In D. Fritsch, Editor, *Proc. of Photogrammetric Week*, pp. 331–340, Heidelberg, Germany, 2009. Wichmann-Verlag.

M. Kada. Aggregation of 3D Buildings With a Hybrid Data Model. *Cartography and Geographic Information Science*, 38(2):153–160, 2011.

V. Kajalin. Screen-Space Ambient Occlusion. In W. Engel, Editor, *Shader X7*, Ch. 6, pp. 413–424. Charles River Media, 2009.

K. Karner, J. Bauer, A. Klaus, F. Leberl, and M. Grabner. Virtual Habitat: Models of the Urban Outdoors. In E. P. Baltsavias, A. Gruen, and L. V. Gool, Editors, *Proc. of Int. Workshop on Automatic Extraction of Man-Made Objects from Aerial and Space Imaging*, pp. 393–402. Swets & Zeitlinger, 2001.

G. Kelly and H. McCabe. A Survey of Procedural Techniques for City Generation. *ITB Journal*, 14:87–130, 2006.

P. J. Kennelly. GIS Applications to Historical Cartographic Methods to Improve the Understanding and Visualization of Contours. *Journal of Geoscience Education*, 50(4):428, 2002.

J. Klimke and J. Döllner. Combining Synchronous and Asynchronous Collaboration within 3D City Models. In S. I. Fabrikant, T. Reichenbacher, M. van Kreveld, and C. Schlieder, Editors, *Proc. of the Int. Conference on Geographic Information Science (GIScience)*, LNCS, pp. 115–129, Berlin, Heidelberg, Germany, 2010. Springer.

A. Klippel. *Wayfinding Choremes*. PhD Thesis, Universität Bremen, 2003.

A. Klippel, K.-F. Richter, T. Barkowsky, and C. Freksa. The Cognitive Reality of Schematic Maps. In L. Meng, K.-F. Richter, T. Barkowsky, and C. Freksa, Editors, *Map-based Mobile Services*, pp. 55–71. Springer, Berlin, Heidelberg, 2005.

A. Klippel, K. Richter, and S. Hansen. Wayfinding Choreme Maps. In *Visual Information and Information Systems*, pp. 94–108, Berlin, Heidelberg, Germany, 2006. Springer.

S. Lamy, A. Ruas, Y. Demazeau, M. Jackson, W. Mackaness, and R. Weibel. The Application of Agents in Automated Map Generalisation. In *Proc. of the Int. Cartographic Conference (ICC)*, pp. 160–169. International Cartographic Association (ICA), 1999.

Y. C. Lee, A. Kwong, L. Pun, and A. Mack. Multi-Media Map for Visual Navigation. *Journal of Geospatial Engineering*, 3(2):87–96, 2001.

A. M. Leonowicz, B. Jenny, and L. Hurni. Automated Reduction of Visual Complexity in Small-Scale Relief Shading. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 45(1):64–74, 2010.

M. Lonergan and C. B. Jones. An Iterative Displacement Method for Conflict Resolution in Map Generalization. *Algorithmica*, 30(2):287–301, 2001.

H. Lorenz. *Texturierung und Visualisierung virtueller 3D-Stadtmodelle*. Ph.D.Thesis, Universität Potsdam, 2010.

H. Lorenz, M. Trapp, M. Jobst, and J. Döllner. Interactive Multi-Perspective Views of Virtual 3D Landscape and City Models. In L. Bernard, A. Friis-Christensen, and H. Pundt, Editors, *Proc. of AGILE Int. Conference on GI Science*, Lecture Notes in Geoinformation and Cartography, pp. 301–321, Berlin, Heidelberg, Germany, 2008. Springer.

F. Losasso and H. Hoppe. Geometry Clipmaps: Terrain Rendering using Nested Tegular Grids. In *Proc. of SIGGRAPH*, pp. 769–776, New York, NY, USA, 2004. ACM Press.

K. Lynch. *The Image of the City*. MIT Press, 1960.

S. Maass. *Techniken zur automatisierten Annotation interaktiver geovirtueller 3D-Umgebungen*. Ph.D.Thesis, HPI, Universität Potsdam, 2009.

A. MacEachren. *How Maps Work: Representation, Visualization and Design*. The Guilford Press, New York, NY, USA, 1st Ed., 1995.

A. MacEachren, R. Edsall, D. Haug, R. Baxter, G. Otto, R. Masters, S. Fuhrmann, and L. Quian. Virtual Environments for Geographic Visualization: Potentials and Challenges. In *ACM Workshop on New Paradigms for Information Visualization and Manipulation, Kansas City, USA, ACM Press*, 1999.

A. M. MacEachren and M. J. Kraak. Research Challenges in Geovisualization. *Cartography and Geographic Information Science*, 28(1):3–12, 2001.

W. Mackaness. Understanding Geographic Space. In W. Mackaness, A. Ruas, and L. Sarjakoski, Editors, *Generalisation of Geographic Information: Cartographic Modelling and Applications*, pp. 1–10. Elsevier Science, 2007.

W. Mackaness and M. Steven. An Algorithm for Localised Contour Removal over Steep Terrain. *The Cartographic Journal*, 43(2):144–156, 2006.

B. Mao, Y. Ban, and L. Harrie. A Multiple Representation Data Structure for Dynamic Visualisation of Generalised 3D City Models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(2): 198–208, 2011.

H. Mayer. Model-Generalization of Building Outlines Based on Scale-Space and Scale-Space Events. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXII-3:531–537, 1998a.

H. Mayer. Technical report, Chair for Photogrammetry and Remote Sensing, TU München, Germany, 1998b.

E. McKean, Editor. *The New Oxford American Dictionary*. Oxford University Press, USA, 2nd Ed., 2005.

R. McMaster and K. Shea. Generalization in Digital Cartography. Technical report, Association of American Geographers, 1992.

L. Meng and A. Forberg. 3D Building Generalisation. In W. Mackaness, A. Ruas, and T. Sarjakoski, Editors, *Challenges in the Portrayal of Geographic Information: Issues of Generalisation and Multi Scale Representation*, Ch. 11, pp. 211–232. Elsevier Science, 2006.

P. Müller, G. Zeng, P. Wonka, and L. Van Gool. Image-Based Procedural Modeling of Facades. *ACM Transactions on Graphics*, 26(3):85, 2007.

Navteq. *NAVTEQ's NAVSTREETS Street Data Reference Manual v2.8*. NAVTEQ, Chicago, IL, USA, 2008.

S. Neubauer and A. Zipf. Suggestions for Extending the OGC Styled Layer Descriptor (SLD) Specification into 3D - Towards Visualization Rules for 3D City Models. In M. Rumor, V. Coors, E. M. Fendel, and S. Zlatanova, Editors, *Proc. of Urban Data Management Symposium (UDMS)*, pp. 133–142. Taylor & Francis, 2007.

M. Neun, D. Burghardt, and R. Weibel. Automated Processing for Map Generalization Using Web Services. *GeoInformatica*, 13(4):425–452, 2008.

NVidia. GPU Programming Guide GeForce 8 and 9 Series. Technical report, 2008.

Object Management Group (OMG). *OMG Unified Modeling Language (OMG UML), Superstructure V.2.4.1*. Object Management Group (OMG), 2011a.

Object Management Group (OMG). *OMG Business Process Model and Notation (BPMN), V.2.0*. Object Management Group (OMG), 2011b.

OGC. *Symbology Encoding Implementation Specification*. Open Geospatial Consortium Inc. (OGC), 2006.

OpenStreetMap. OSM Mapnik StyleSheet, `http://svn.openstreetmap.org/applications/rendering/mapnik/osm.xml` (accessed 1.7.2012) 2012.

A. Oulasvirta, S. Estlander, and A. Nurminen. Embodied Interaction with a 3D versus 2D Mobile Map. *Personal and Ubiquitous Computing*, 13(4):303–320, 2009.

R. Pajarola and E. Gobbetti. Survey of Semi-Regular Multiresolution Models for Interactive Terrain Rendering. *The Visual Computer*, 23:583–605, 2007.

B. Pan, X. Chen, X. Guo, W. Chen, and Q. Peng. Interactive Expressive Illustration of 3D City Scenes. In *Proc. of Int Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics)*, pp. 406–410. IEEE Computer Society Press, 2011.

S. Pasewaldt, M. Trapp, and J. Döllner. Multiscale Visualization of 3D Geovirtual Environments Using View-Dependent Multi-Perspective Views. *Journal of WSCG*, 19(3):111–118, 2011.

M. Peter, N. Haala, and D. Fritsch. Preserving Ground Plan and Facade Lines For 3D Building Generalization. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVII-B2, 2007.

D. Peters, Y. Wu, and S. Winter. Testing Landmark Identification Theories in Virtual Environments. *Spatial Cognition VII*, 6222/2010(1):54–69, 2010.

R. J. Phillips. An Experiment with Contour Lines. *The Cartographic Journal*, 16(2):72–76, 1979.

H. Qu, H. Wang, W. Cui, Y. Wu, and M.-Y. Chan. Focus+Context Route Zooming and Information Overlay in 3D Urban Environments. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1547–54, 2009.

J. Quantz, J. Döllner, R. Fricke, R. Tolksdorf, T. Hoppe, and I. Jung. DigiPolis - An Approach for Interactive 3D Building & Interior Models as Communication Tools. In J. Sieck, Editor, *Multimediale Systeme*, Kultur und Informatik. Verlag Werner Hülsbusch, 2011.

J. L. Raheja. *Recognition of 3D Settlement Structure for Generalization*. Ph.D.Thesis, TU München, 2005.

D. Rainsford and W. Mackaness. Template Matching in Support of Generalisation of Rural Buildings. In *Advances in Spatial Data Handling*, pp. 137–151. Springer, 2002.

J. Rau, L. Chen, F. Tsai, K. Hsiao, and W. Hsu. LOD Generation for 3D Polyhedral Building Model. *Advances in Image and Video Technology*, 4319:44–53, 2006.

N. Regnauld and R. McMaster. A Synoptic View of Generalization Operators. In W. Mackaness, A. Ruas, and L. Sarjakoski, Editors, *Generalisation of Geographic Information: Cartographic Modelling and Applications*, Ch. 3, pp. 36–66. Elsevier Science, 2007.

T. Reichenbacher. Adaptive Egocentric Maps for Mobile Users. In L. Meng, T. Reichenbacher, and A. Zipf, Editors, *Map-based Mobile Services*, Ch. 10, pp. 141–158. Springer, 2005.

A. Reimer and J. Fohringer. Towards Constraint Formulation for Chorematic Schematisation Tasks - Work in Progress. In *Proc. of ICA Workshop on Generalization and Multiple Representation*. International Cartographic Association (ICA), 2010.

F. Remondino and S. El-Hakim. Image-Based 3D Modelling: A Review. *The Photogrammetric Record*, 21(115):269–291, 2006.

J. Ribelles, P. Heckbert, M. Garland, T. Stahovich, and V. Srivastava. Finding and Removing Features from Polyhedra. In *Proc. of Design Engineering Technical Conference (DETC)*, pp. 1–10, Pittsburgh, PA, USA, 2001. American Association of Mechanical Engineers (ASME).

N. Ripperda. Determination of Facade Attributes for Facade Reconstruction. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVII-B3a: 285–290, 2008.

G. G. Robertson, S. K. Card, and J. D. Mackinlay. Information Visualization Using 3D Interactive Animation. *Communications of the ACM*, 36(4):57–71, 1993.

L. Rognant, J. Planes, M. Memier, and J. Chassery. Contour Lines and DEM: Generation and Extraction. In *Digital Earth Moving*, pp. 87–97. Springer, 2001.

J. Royan, C. Bouville, and P. Gioia. Pbtree: A New Progressive and Hierarchical Representation for Network-Based Navigation in Urban Environments. *Annales des Télécommunications*, 60 (11-12):1394–1421, 2005.

J. Royan, P. Gioia, R. Cavagna, and C. Bouville. Network-Based Visualization of 3D Landscapes and City Models. *Computer Graphics and Applications, IEEE*, 27(6):70–79, 2007.

A. Ruas. A Method for Building Displacement in Automated Map Generalisation. *International Journal of Geographical Information Science*, 12(8):789–803, 1998.

T. T. Sankey. Scale, Effects. In S. Shekhar and H. Xiong, Editors, *Encyclopedia of GIS*, pp. 1021–1026. Springer, 2008.

A. Schilling, M. Over, S. Neubauer, G. Walenciak, and P. Neis. Interoperable Location Based Services for 3D Cities on the Web using User Generated Content from OpenStreetMap. In *Proc. of Urban Data Management Symposium (UDMS)*. CRC Press, Taylor & Francis Group, 2009.

W. Schroeder, K. Martin, and B. Lorensen. *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition*. Kitware, 2006.

A. Semmo, J. E. Kyprianidis, and J. Döllner. Automated Image-Based Abstraction of Aerial Images. In M. Painho, M. Y. Santos, and H. Pundt, Editors, *Geospatial Thinking*, Lecture Notes in Geoinformation and Cartography, pp. 359–378. Springer, 2010.

A. Semmo, M. Trapp, J. E. Kyprianidis, and J. Döllner. Interactive Visualization of Generalized Virtual 3D City Models using Level-of-Abstraction Transitions. *Computer Graphics Forum*, 31 (3), 2012.

J. P. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.

M. Sester. Generalization Based on Least Squares Adjustment. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIII-4:931–938, 2000a.

M. Sester. *Maßstabsabhängige Darstellungen in digitalen räumlichen Datenbeständen*. Habilitation, Universität Stuttgart, 2000b.

M. Sester. Application Dependent Generalization - The Case of Pedestrian Navigation. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIV-4: 8–12, 2002a.

M. Sester. Typifizierung mittels Kohonen Merkmalskarten. *Mitteilungen des Bundesamtes für Kartographie und Geoinformation*, 22:161–169, 2002b.

M. Sester. 3D Visualization and Generalization. In D. Fritsch, Editor, *Proc. of Photogrammetric Week*, Vol. 7, pp. 285–295. Wichmann-Verlag, 2007.

M. Sester and C. Brenner. Continuous Generalization for Fast and Smooth Visualization on Small Displays. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXV-4:1293–1298, 2004.

M. Sester and A. Klein. Rule Based Generalization of Buildings for 3D-Visualization. In *Proc. of the Int. Cartographic Conference (ICC)*, pp. 214–224, Ottawa, Canada, 1999. International Cartographic Association (ICA).

C. Silva, Y. Chiang, J. El-Sana, and P. Lindstrom. Out-of-Core Algorithms for Scientific Visualization and Computer Graphics. In *Proc. of IEEE Visualization*, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

S. Smirnov, H. Reijers, and M. Weske. A Semantic Approach for Business Process Model Abstraction. *Advanced Information Systems Engineering*, pp. 497–511, 2011.

M. E. Sorrows and S. C. Hirtle. The Nature of Landmarks for Real and Electronic Spaces. In *Proc. of Int. Conf. on Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science*, pp. 37–50, London, UK, 1999. Springer.

W. Staufenbiel. *Zur Automation der Generalisierung topographischer Karten mit besonderer Berücksichtigung grossmaßstäbiger Gebäudedarstellungen*. Ph.D. Thesis, Universität Hannover, 1973.

J. Stoter, B. Baella, C. Blok, D. Burghardt, M. Pla, N. Regnauld, O. Survey, U. Kingdom, G. Touya, K.-h. Anders, J. Haunert, N. Bakker, A. Dortland, P. Lentjes, P. Rosenstand, and S. Schmid. *State-of-the-Art of Automated Generalisation in Commercial Software*. Gopher/ EuroSDR, Amsterdam, Netherlands, 2010.

J. E. Stoter. *3D Cadastre*. Ph.D. Thesis, TU Delft, Netherlands, 2004.

R. Stüber. *Generalisierung von Gebäudemodellen unter Wahrung der visuellen Richtigkeit*. Ph.D. Thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2005.

K. Tanaka. The Relief Contour Method of Representing Topography on Maps. *Geographical Review*, 40(3):444–456, 1950.

TeleAtlas. *MultiNet Germany 2006.10 Release Notes*. Teleatlas, 2006.

F. Thiemann. Generalization of 3D Building Data. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIV-4:286–290, 2002.

F. Thiemann. Interpretation of Building Parts from Boundary Representation. In *Proc. of ISPRS Workshop on Next Generation 3D City Models*, pp. 29–34, Bonn, Germany, 2005.

F. Thiemann and M. Sester. 3D-Symbolization using Adaptive Templates. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVI-2:49–54, 2006.

M. Trapp and J. Döllner. Real-Time Volumetric Tests Using Layered Depth Images. In K. Mania and E. Reinhard, Editors, *Proc. of Eurographics (Shortpapers)*, pp. 235–238. Eurographics, The Eurographics Association, 2008.

M. Trapp, T. Glander, H. Buchholz, and J. Döllner. 3D Generalization Lenses for Interactive Focus + Context Visualization of Virtual City Models. In *Proc. of Int. Conf. on Information Visualization (IV)*, pp. 356–361. IEEE Computer Society Press, 2008.

M. Trapp, A. Semmo, R. Pokorski, C.-D. Herrmann, J. Döllner, M. Eichhorn, and M. Heinzelmann. Communication of Digital Cultural Heritage in Public Spaces by the Example of Roman Cologne. In M. Ioannides, Editor, *Digital Heritage, Proceedings of 3rd EuroMed Conference*, Lecture Notes in Computer Science (LNCS), pp. 262–276. Springer-Verlag Berlin Heidelberg, 2010.

F. Tsai, W. Lin, and L. Chen. Multi-Resolution Representation of Digital Terrain and Building Models. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-2, 2008.

B. Tversky. Distortions in Cognitive Maps. *Geoforum*, 23(2):131–138, 1992.

M. van Kreveld. Smooth Generalization for Continuous Zooming. In *Proc. of the Int. Cartographic Conference (ICC)*, pp. 2180–2185. International Cartographic Association (ICA), 2001.

M. van Kreveld, R. V. Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour Trees and Small Seed Sets for Isosurface Traversal. In *Proc. of Symposium on Computational Geometry*, pp. 212–220, New York, NY, USA, 1997. ACM press.

J. van Wijk and A. Telea. Enridged Contour Maps. In *Proc. of IEEE Visualization*, pp. 69–74, Los Alamitos, CA, USA, 2001. IEEE Computer Society Press.

J. Viega, M. J. Conway, G. Williams, and R. Pausch. 3D Magic Lenses. In *Proc. of ACM Symp. on User Interface Software and Technology (UIST)*, pp. 51–58, New York, NY, USA, 1996. ACM Press.

N. G. Vinson. Design Guidelines for Landmarks to Support Navigation in Virtual Environments. In *Proc. of Computer Human Interface Conference (CHI)*, pp. 278–285, New York, NY, USA, 1999. ACM Press.

M. Visvalingam and P. Williamson. Simplification and Generalisation of Large-Scale Data for Roads : a Comparison of Two Filtering Algorithms. *Cartography and Geographical Information Systems*, 22(4):264–275, 1995.

C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2nd Ed., 2004.

J. Ware and C. Jones. Conflict Reduction in Map Generalization using Iterative Improvement. *GeoInformatica*, 2(4):383–407, 1998.

S. Winter, M. Tomko, B. Elias, and M. Sester. Landmark Hierarchies in Context. *Environment and Planning B: Planning and Design*, 35(3):381–398, 2008.

M. Wolff and H. Asche. Towards Geovisual Analysis of Crime Scenes - A 3D Crime Mapping Approach. In S. Monika, L. Bernard, and P. Volker, Editors, *Proc. of AGILE Int. Conference on GI Science*, LNGC, pp. 429–448, Berlin, Heidelberg, Germany, 2009. Springer.

X. Zhao, W. Zeng, D. Gu, A. Kaufman, W. Xu, and K. Mueller. Conformal Magnifier: A Focus+Context Technique with Local Shape Preservation. *IEEE Transactions on Visualization and Computer Graphics*, 2012.

C. Zhuo, Z. Yanqing, and Y. Chongjun. Real-time Contour Map Reconstruction with 3D Terrain on Modern Graphics Hardware. *International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-4W, 2009.

List of Figures

List of Tables

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Dissertation ohne Hilfe Dritter und ohne Zuhilfenahme anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

San Sebastian, den 1.7.2012

Tassilo Glander