FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Advances in the Multimodal 3D Reconstruction and Modeling of Buildings

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Dipl.-Ing. Mag.rer.soc.oec. Michael Schwärzler, Bakk. techn.
Matrikelnummer 00325222

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer

Diese Dissertation haben begutachtet:

Prof. Dr. Elmar Eisemann

Associate Prof. Mag. Dr. Hannes Kaufmann

Wien, 27. Juni 2018

Michael Schwärzler

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Advances in the Multimodal Reconstruction and Modeling of 3D Buildings

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dipl.-Ing. Mag.rer.soc.oec. Michael Schwärzler, Bakk. techn.
Registration Number 00325222

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer

The dissertation has been reviewed by:

Prof. Dr. Elmar Eisemann	Associate Prof. Mag. Dr. Hannes Kaufmann

Vienna, 27th June, 2018

Michael Schwärzler

Erklärung zur Verfassung der Arbeit

Dipl.-Ing. Mag.rer.soc.oec. Michael Schwärzler, Bakk. techn.
Mexikoplatz 24/23
1020 Wien
Österreich

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. Juni 2018

Michael Schwärzler

Acknowledgements

This thesis would not have been possible without the support of many people. First, I would like to thank my thesis advisor Michael Wimmer for all the input, helpful discussions, and the collaboration in various research projects. It is an honor for me to have Elmar Eisemann and Hannes Kaufmann as reviewers for this thesis.

I would also like to thank all my current and former colleagues at the VRVis Research Center, especially the members of the *Semantic Modelling and Acquisition* group, Stefan Maierhofer, Christian Luksch, Harald Steinlechner, Lisa Kellner, Georg Haaser, Andreas Walch, Attila Szabo, and Janne Mihola – but also all the other friends I found here who helped me a lot with valuable input, inspiring brainstorming sessions, an extraordinary working environment and emotional support when it was necessary. Moreover, I would like to thank my co-authors Murat Arikan, Daniel Scherzer, Oliver Mattausch, and Simon Flöry, without whom it would not have been possible to complete my research.

For the finalization of this thesis, I would especially like to thank Helfried Gschwandtner, who pushed me to “finally get this work done”, and of course Theresia Gschwandtner, Maria Wimmer, Matthias Schlachter, Andreas Reichinger and Thomas Ortner for their reviews, corrections, suggestions and help.

Finally, I want to thank my parents Karin and Ronald Schwärzler, as well as my brother Alexander and my sister Marion for supporting me throughout the creation of this thesis.

This work was enabled by the Competence Centre VRVis. VRVis is funded by BMVIT, BMWFW, Styria, SFG and Vienna Business Agency in the scope of COMET - Competence Centers for Excellent Technologies (854174), which is managed by FFG.

Kurzfassung

Der Wunsch nach schnelleren und effizienteren Methoden zur Digitalisierung städtischer Umgebungen hat die Verfügbarkeit von erschwinglichen Systemen zur 3D-Datenakquise für Gebäude in den letzten Jahren rasch ansteigen lassen: Sowohl Laserscanner als auch photogrammetrische Methoden erzeugen Millionen an 3D-Punkten innerhalb weniger Minuten Aufnahmezeit. Sie werden sowohl vom Boden aus als auch mit der Hilfe von Drohnen aus der Luft angewandt, und zur Ergänzung traditioneller tachymetrischer Punkte im Vermessungswesen verwendet. Doch diese Datenpunkte sind nicht die einzigen Informationsquellen: Extrahierte Metadaten aus Bildern, Simulationsresultate (z.B. aus Lichtsimulationen), 2D Grundrisspläne und semantische Annotationen – speziell aus Building Information Modeling (BIM) Systemen – werden immer häufiger verfügbar.

Die Herausforderungen, welche diese *Multimodalität* während der Rekonstruktion von CAD-fähigen 3D-Gebäuden mit sich bringt, sind vielfältig: Neben der enormen Größe der Daten ist auch deren korrekte Registrierung in einem gemeinsamen Kontext für die weitere Verarbeitung eine große Herausforderung – speziell aufgrund der Tatsache, dass Daten oft unvollständig oder fehlerhaft sind. Nichtsdestotrotz sind die Potenziale für eine Verbesserung der Effizienz bei der Verarbeitung als auch bei der Qualität der Resultate enorm: Fehlende Information kann durch Daten aus anderen Quellen ersetzt werden, die Ableitung räumlicher oder semantischer Relationen hilft, Einschränkungen zu umgehen, und die Komplexität der interaktiven Modellierung kann reduziert werden (z.B. durch die Limitierung der Interaktionen auf einen zweidimensionalen Raum).

Im Rahmen dieser Dissertation werden vier Publikationen präsentiert, welche jeweils ein Teilproblem aus dem Bereich der *multimodalen Städterekonstruktion* behandeln. Die entstandenen Ergebnisse stellen modulare “Bausteine” dar, die als Werkzeuge in diesem Anwendungsbereich frei kombiniert werden können. Zuerst werden effiziente Methoden zur Schattenberechnung von Flächenlichtquellen vorgestellt – zum einen mit einem Fokus auf der effizienten Generierung physikalisch korrekter Halbschatten, und zum anderen mit dem Ziel, berechnete Schatten in Folgebildern zur Vermeidung kostspieliger Neuberechnungen wiederzuverwenden. In weiterer Folge wird ein optimierungsunterstütztes Rekonstruktions- und Modellierungswerkzeug vorgestellt, welches skizzenhafte Interaktionen und Fangtechniken einsetzt, um 3D-Gebäude zu erstellen. Anschließend wird eine Erweiterung dieses Systems demonstriert, welche 2D-Fotos als einzige Interaktionsebene für eine simple, skizzenhafte Generierung texturierter Gebäudegeometrie verwendet.

Abstract

Driven by the need for faster and more efficient workflows in the digitization of urban environments, the availability of affordable 3D data-acquisition systems for buildings has drastically increased in the last years: Laser scanners and photogrammetric methods both produce millions of 3D points within minutes of acquisition time. They are applied both on street-level as well as from above using drones, and are used to enhance traditional tachymetric measurements in surveying. However, these 3D data points are not the only available information: Extracted meta data from images, simulation results (e.g., from light simulations), 2D floor plans, and semantic tags – especially from the upcoming Building Information Modeling (BIM) systems – are becoming increasingly important.

The challenges this *multimodality* poses during the reconstruction of CAD-ready 3D buildings are manifold: Apart from handling the enormous size of the data that is collected during the acquisition steps, the different data sources must also be registered to each other in order to be applicable in a common context – which can be difficult in case of missing or erroneous information. Nevertheless, the potential for improving both the workflow efficiency as well as the quality of the reconstruction results is huge: Missing information can be substituted by data from other sources, information about spatial or semantic relations can be utilized to overcome limitations, and interactive modeling complexity can be reduced (e.g., by limiting interactions to a two-dimensional space).

In this thesis, four publications are presented which aim at providing freely combinable "building blocks" for the creation of helpful methods and tools for advancing the field of *Multimodal Urban Reconstruction*. First, efficient methods for the calculation of shadows cast by area light sources are presented – one with a focus on the most efficient generation of physically accurate penumbras, and the other one with the goal of reusing soft shadow information in consecutive frames to avoid costly recalculations. Then, a novel, optimization-supported reconstruction and modeling tool is presented, which employs sketch-based interactions and snapping techniques to create water-tight 3D building models. An extension to this system is demonstrated consecutively: There, 2D photos act as the only interaction canvas for the simple, sketch-based creation of building geometry and the corresponding textures. Together, these methods form a solid foundation for the creation of common, multimodal environments targeted at the reconstruction of 3D building models.

Contents

CHAPTER 1

Introduction

1.1 Overview

Reconstructing and modeling urban scenarios has become an important task in today's industry. The area of urban reconstruction typically includes the generation of three-dimensional geometry and corresponding textures based on input data that was captured in an urban environment. It is currently a very actively investigated research field affecting not only the areas of computer graphics, computer vision, and computational geometry, but also several application-oriented domains such as urban planning, lighting design, or surveying.

Through technical advances combined with a drastic reduction in prices, acquisition techniques, as for instance laser scanners, photogrammetry, or infrared-based range scanners, have turned into highly sophisticated, but also widely available consumer-level tools. With them, enormous amounts of point-based data (i.e., a large number of three-dimensional coordinates of captured surfaces, sometimes with meta data such as color, normal vectors, etc.) of all different kinds of objects are generated. This "traditional" kind of data acquisition is more and more enhanced by additional data sources, such as photographs, user-generated annotations, or other semantic sources like 2D maps or data captured from user behavior (e.g., trajectories from cellphone GPS data). Moreover, simulations are used to immediately evaluate the reconstructed and modeled scenarios in their destined problem domain: The optimal placement of protection measures in case of flooding, the impact of a new building on the landscape, or – as treated in this thesis – the distribution of light in a 3D scenario are examples for a simulation-based method of data generation affecting the final reconstruction result. This has led to new challenges in terms of *processing* the acquired and simulated data, as the ensuing modelling and post-processing steps are by far not as sophisticated and unproblematic to handle as the acquisition.

Driven by the differing needs of the corresponding fields, the requirements for any urban reconstruction depend on the context it will later be used in. Apart from a pure reconstruction of the geometric structure and the texture of objects, the demand for additional contextual information and the inclusion of simulation data is rising continuously. This is caused by the increasing complexity of the available data and the increasingly complex use-cases. *Multimodal* information is not only important in further application steps (e.g., manipulable items, labeling, geo-referencing, etc.), but can also provide valuable help during the acquisition or modeling process itself (see Figure 1.1 for an example). The embedding of such additional information into the reconstruction process of 3D buildings is not only an optional improvement, but a necessity in order to improve today's modeling workflows in terms of speed and feasibility. This is also reflected in the ideas and concepts of the so-called *Building Information Modeling (BIM)*[1] process, where 3D data is enriched by a fourth (time) and a fifth (costs) dimension. The *German Federal Ministry of Transport and Digital Infrastructure (BMVI)* has released a plan that targets the mandatory application of BIM processes in building construction

[1]https://en.wikipedia.org/wiki/Building_information_modeling

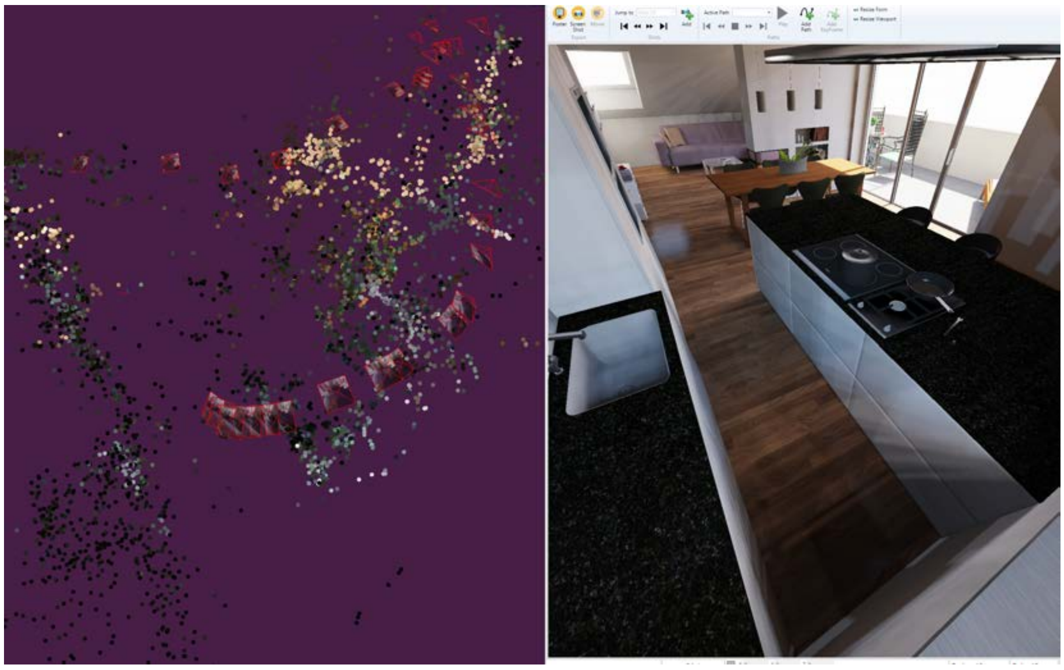

Figure 1.1: The connections between reconstruction and light transport simulation are manifold: In this example, a lighting design tool is used to create realistic “virtual photos” of a scene (right). These photos act as ground-truth example during the development and adjustment of a novel photogrammetric reconstruction pipeline (left). By simulating nearly any possible artifact that can occur in real photos, the robustness of the photogrammetric solution can be greatly increased, and the accuracy can be tracked exactly. In this thesis, both photogrammetric data (for urban reconstruction) as well as light transport simulation (for the fast calculation of soft shadows) play an important role. The screenshots were taken from prototypes [Sza18, LTH+13] developed at the *VRVis Research Center* in the context of the SHARC project (Section 7.3.1.)

from 2020 onwards[2]. Ideally, BIM should depict the hole life cycle of a building, and covers (apart from geometric data) manufacturer’s annotations, spatial relations, room usage, lighting information, and more. Using multimodal data during the reconstruction process therefore fits closely to the BIM system idea, and could help in making this transition towards BIM processes in building construction possible.

This thesis presents work from peer-reviewed publications that were created in the context of multimodal urban reconstruction, modeling, and simulation. Even though different aspects are treated – from an optimization-based, interactive creation of low-polygonal 3D buildings using novel snapping methods over photo-based 2D sketching tools up to efficient, physically accurate direct light transport – every part of it acts as a freely combinable “building block” for the creation of methods and tools for advancing the field.

[2]https://www.bmvi.de/blaetterkatalog/index.html?catalog=319312

The publications have emerged from the work in research projects in the *Semantic Modeling and Acquisition Group*[3] at the *VRVis Research Center* in Vienna, Austria[4]. One of the ultimate long-term goals of the research group is to create means to develop workflows *combining the two worlds of light transport and urban reconstruction.* A tight integration of these aspects in a common environment does not only increase the respective achievable accuracy, but also induces completely novel workflows and approaches in these interactive domains. Considering this strategic context the publications are embedded in, it becomes clear how the topic of the chapters 3 and 4 (casting soft shadows in real time) are connected to the goal of the chapters 5 and 6 (interactively reconstructing 3D buildings using suggestions and guidance), and how further research in this direction could benefit from these approaches.

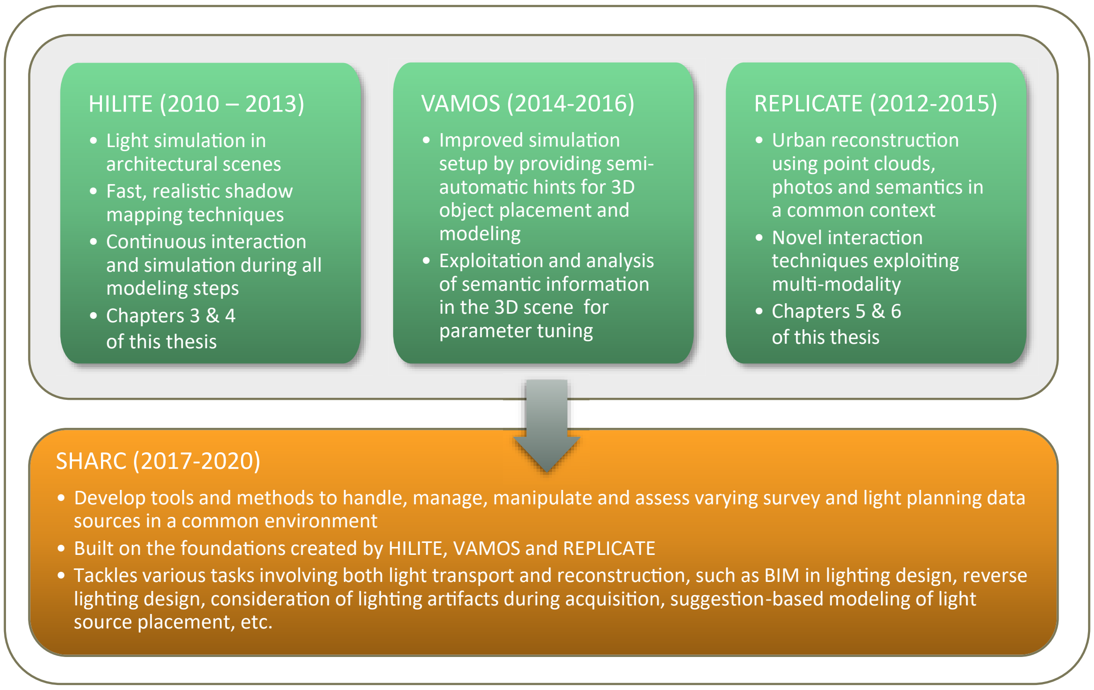
HILITE (2010 – 2013)
• Light simulation in architectural scenes
• Fast, realistic shadow mapping techniques
• Continuous interaction and simulation during all modeling steps
• Chapters 3 & 4 of this thesis
VAMOS (2014-2016)
• Improved simulation setup by providing semi-automatic hints for 3D object placement and modeling
• Exploitation and analysis of semantic information in the 3D scene for parameter tuning
REPLICATE (2012-2015)
• Urban reconstruction using point clouds, photos and semantics in a common context
• Novel interaction techniques exploiting multi-modality
• Chapters 5 & 6 of this thesis
SHARC (2017-2020)
• Develop tools and methods to handle, manage, manipulate and assess varying survey and light planning data sources in a common environment
• Built on the foundations created by HILITE, VAMOS and REPLICATE
• Tackles various tasks involving both light transport and reconstruction, such as BIM in lighting design, reverse lighting design, consideration of lighting artifacts during acquisition, suggestion-based modeling of light source placement, etc.

Figure 1.2: Overview on the research setting this thesis was created in: The projects HILITE, VAMOS and REPLICATE treated various aspects relevant for the field of *Multimodal Urban Reconstruction and Modeling*. In their context, the publications that are the basis of this thesis were published. They act as "building blocks" for the creation of a common, overarching approach, where 3D reconstruction and light transport are tackled together. For this, the SHARC project (see Section 7.3.1) was started in 2017.

In both *applied and strategic research projects*, the author identified and tackled problems that posed a significant scientific challenge and led to novel findings in the field of

[3]https://www.vrvis.at/research/semantic-modelling-and-acquisition-group/
[4]https://www.vrvis.at/

Multimodal Urban Reconstruction and Modeling – but at the same time, they were also of great interest for our industry partners. The results were therefore published at international, peer-reviewed conferences or journals, but also further refined and implemented in industrial applications.

In the following sections of this introductory chapter, an insight into the problems (1.2) and targeted research goals in the field of *Multimodal Urban Reconstruction and Modeling* (1.3) are provided. In Chapter 2, the contributions and the research setting in which the work for the publications have been carried out, and in which these "building blocks" for such a common environment have been developed, are summarized and presented (see Figure 1.2 for an overview). The papers that build the foundation for this thesis, several further co-authored publications, and supervised diploma theses are briefly described to provide a better understanding of the integrated approach this thesis was created in. The work presented in Chapters 3 to 6 features the main contributions of this thesis.

Chapter 7 concludes this thesis. Apart from summarizing the advances in the scientific field of *Multimodal Urban Reconstruction and Modeling*, a perfect example of the success of the chosen strategy is given by the outlook on a recently started research project (Section 7.3): *SHARC* is the name of the first funded project at the *VRVis Research Center* in which the targeted strategical combination of light transport and reconstruction in a common solution takes place. Coarse ideas about planned research projects that are currently awaiting funding are presented subsequently.

1.2 Open Problems in the Research Field

The field of *Multimodal Urban Reconstruction* is confronted with the following challenges that are the driving factors for research and novel developments in this field:

- **Vast amounts of data:** point clouds with billions of points are stored in data sets with file sizes that exceed today's memory capacity by orders of magnitude.
- **Heterogeneity of data:** point clouds, photos, photogrammetry, tachymetric measurements or simulation data are just a few sources that are typically available – all with their own definitions, coordinate systems, accuracy, density, etc.
- **Erroneous data:** caused by occlusions, complex materials, capturing artifacts, sampling rates, lighting conditions, numerical issues, compression, etc.
- **Complex-to-use, unintuitive modeling and simulation tools**, which are hard to grasp and handle, and that do not allow reaching the targeted level of detail of the 3D model, or the proper simulation settings.
- **Insufficient visual quality** due to environmental lighting artifacts captured in the data that remain in any created textures.

- **Reduced interactivity**, caused by long calculations in the processing or simulation phases, or by algorithms not suitable for real-time scenarios.

There is no single or direct way for solving these problems in general, as the required outcomes of a building reconstruction vary extremely, depending on the application they are further used in. For example, architectural visualizations require 3D models with high-quality textures and materials, but don't have to be extremely accurate. In the case of facade restoration work, every small detail has to be captured, whereas usually no special demands for a visualization are stated. The given challenges and the differing requirements sum up to the fact that a fast, easy and inexpensive high-quality reconstruction of an urban environment that is generally applicable is currently not possible.

Hence, the requirements and constraints for the 3D buildings to reconstruct were set as follows to be suitable for further processing in our industry partners' applications:

- The 3D models of buildings in urban environments are reconstructed in order to generate accurate and convincing real-time simulation environments.
- The 3D content therefore has to be modelled as accurately as possible and has to be visually convincing.
- Textures are generated from photos taken on-site to maximize the recognition value, and should not contain artifacts from illumination or occluders.
- At the same time, the level of detail of the 3D building models must not be too high, i.e., the number of polygonal faces has to be low, since the resources that can be used in a real-time simulation are limited.
- Missing or erroneous data has to be compensated (i.e., holes in the geometry and in textures should be avoided by finding suitable substitutes).
- The reconstructed 3D buildings have to be CAD-ready, which means they have to be reconstructed in such a way that a user of a CAD software can immediately use and extend it in an arbitrary 3D modelling package. This induces representing the entities and hierarchies the building consists of in the data structures (instead of a simple accumulation of vertices and edges).

In the case of reconstructing 3D buildings for further use in real-time simulations, games, GIS or planning systems, accurate, low-polygonal, textured 3D models are required. Especially for these applications, both current manual and automatic approaches have severe practical limitations. They have to be tackled to satisfy the constraints mentioned above:

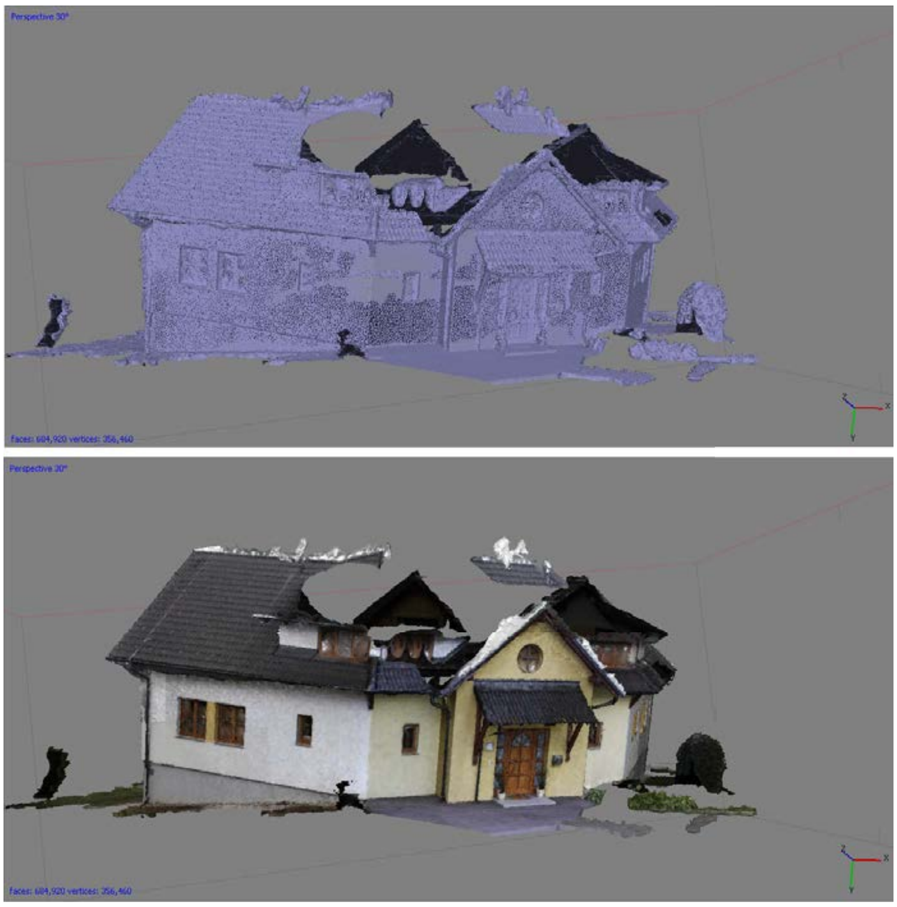

Figure 1.3: Automatic mesh reconstruction techniques generate noisy geometry consisting of several thousand or even millions of triangles, making them not suitable for efficient further processing in CAD tools, or for the direct use in real-time applications due to their complexity and hard-to-repair errors. In this figure, an automatically generated mesh from a data set of 100 photos is shown (top: wireframe mode, bottom: textured triangulated surface). It was created using the commercial software Agisoft PhotoScan [Agi18], and consists of 684.920 triangles. Note that the cloudy sky has been interpreted as part of the geometry as well.

Geometry creation, meshing and level of detail: The data acquired throughout the capturing process typically consists of hundreds of thousands or even millions of points, and can therefore not be used directly in real-time applications, where polygonal shapes with low face count are needed (see Figure 1.3). The final face count achievable with commercially available fully automatic approaches (e.g., Agisoft PhotoScan [Agi18]) is still significantly higher than in a model created manually by a skilled 3D artist. Furthermore, these meshing approaches rely heavily on the assumption that the target surface is closed (i.e., has no boundary), and can therefore not reconstruct arbitrary topologies. In case of data captured on street level, this condition can hardly be fulfilled due to various reasons (see next point).

Illumination-free texturing: Photos taken on-site are either manually "placed" onto the modelled geometry, or in case of a reconstruction from photogrammetric data sets, the image information of the positioned and oriented photos can be automatically reprojected onto the building geometry. While this seems to be a promising approach for solving the texturing issue at first sight, the amount of manual post-processing that has to be applied in order to achieve high-quality textures is still very large (see Figure 1.4): Photos contain the illumination that was prevalent in the scene during the acquisition process. Apart from the fact that the amount of stored light depends on various factors (angle between camera and light source / surfaces, camera settings, post production, etc.) for each photo, the changing position of the sun over time is an aspect that has to be considered. In order to generate textures for all surfaces of a building, image information from different photos has to be "stitched" together. Since lighting conditions and viewpoints between different photos change, clearly visible stitching artifacts are likely to occur. Another issue to consider is that occluding objects (i.e., objects that lie between the camera and the object to texture) are present nearby nearly every building, causing them to occur on the photos as well as on the reprojected textures. Sometimes, certain parts of the building can simply not be captured since the area is not accessible. Thus, texturing areas that are not or only partly visible on photos in an easy and fast way can only be done by cleverly reusing existing data, which induces the need to "understand" what kind of object has to be textured synthetically.

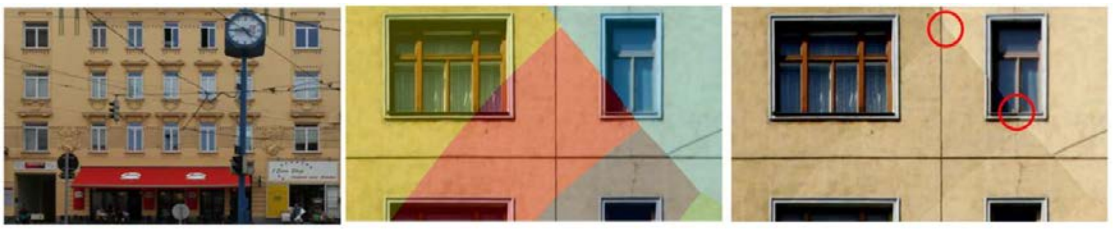

Figure 1.4: Texturing problems (screenshots created with a research prototype developed at the *VRVis Research Center*): Occlusions (left) and the combination of multiple photos (indicated by colors, middle) lead to artifacts (right). The so-called stitching artifacts when using multiple photos are caused by different lighting conditions and camera settings in the individual photos.

Noise, missing data, visibility: All point data acquired by current capturing techniques is erroneous and noisy: Apart from the device-dependent scan error rate, point data captured on street level often suffers from influences such as illumination problems (insufficient light, shadows, blending, reflections, lens flare, ...), shiny object materials (glass windows/facades, metal, mirrors, ...), weather conditions (rain, objects moving in the wind, ...), and occluders (trees, cars, persons, other buildings, ...). A further disturbing factor is the limited area from which the acquisition process can be accomplished: Even if a building is freely accessible from all sides, the capturing process can only take place from street level in nearly all cases, thus missing balcony windows or other details on higher floors, the roof, chimneys, etc. to be fully depicted in the data.

Modeling process, effort and accuracy: Today's 3D modeling tools are highly optimized in terms of functionality and user interfaces – but only for the creation of 3D content "from scratch", i.e., they do not integrate the reconstruction input data into the modeling process. While some of them allow point clouds to be imported, this is hardly true for complete photogrammetric data sets, and the points are used only as a visual orientation help for the artist. Hence, 3D artists have to rely on their skills and patience in order to model a polygonal building that represents the real-world object as accurately as possible, and have no verification method except their visual perception.

Re-usability and further editing: 3D models reconstructed from real-world data sets of buildings are a "static snapshot" of the scene at a given point of time. Modifications and changes in both the geometry and especially the texture pose a big problem: Since there is no reconstruction data available for modified areas, it is a hard task even for skilled 3D artists to create artificial textures that equal the quality of the reprojected photos. Moreover, geometric modifications are purely based on polygonal editing operations, and do not consider the building's structure, topology, or semantic layout.

Assignment and use of semantic information: Typically, semantic information plays a role whenever actions or interactions with 3D models are required in the simulation or the game they are used in: It is then important to display further information to a user of the system, or to know how objects can be manipulated, combined, triggered, exchanged, referenced, etc. Such behavior is usually linked to semantically annotated parts of the model in the program logic. Instead of using the inherent information during the generation of the 3D model to simplify and shorten the process, semantic annotations are currently defined manually after the geometric modelling step, therefore even extending the overall processing time.

1.3 Research Goals

The main goal of this dissertation is to contribute to improving today's workflows in the field of *Multimodal Urban Reconstruction* by proposing novel algorithms that can be employed in such a context. In particular, suggestion-based 3D interaction methods for working on point-cloud data and real-time shadow casting algorithms for occluder detection and visibility calculations are presented. They act as "building blocks" to be used in multimodal reconstruction and modeling systems taking all kinds of input data, suggestion-based interaction concepts, performance enhancements and high-quality, accurate, CAD-ready results into account. The most important aspects, and therefore the overarching goals of the multimodal research concept, are:

The combination of various types of input data into a common context: Throughout the modeling process, the system combines geometric, image-based and other data (simulation output, 2D plans from GIS systems, tachymetric measurements, or other semantic information) to enhance operations of all kinds (see Figure 1.5). Information that is missing in one type of data can possibly be substituted or enhanced by using the other data sources. Examples for such approaches are presented in the Chapters 5 and 6.

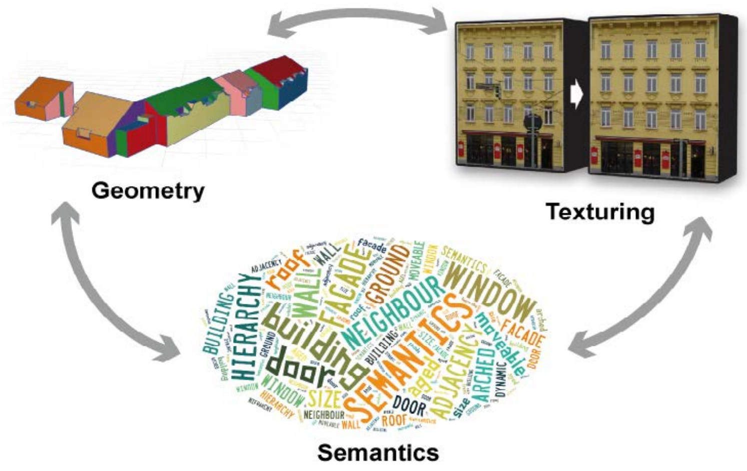

Figure 1.5: The simultaneous and interactive handling of geometry, textures and semantic information (which can be simulation output, 2D plans from GIS systems, tachymetric measurements, extracted relations, etc.) in a common framework. (1) Semantics are extracted from geometrical relations and spatial configurations as well as from image data. In return, this semantic information can be used to repair and complete missing data, and to simplify modeling operations. (2) An accurate geometric representation is necessary for an automatic texture reprojection and helps to detect occlusions in the texture synthesis process. (3) Texture analysis helps to increase the geometric accuracy by providing additional information (edges, corners, etc.), and allows the generation of more detailed building surfaces.

The utilization of information and spatial relations: Information in both geometric and image data can be sparse and incomplete – but if the context is known, repairing such regions can still be accomplished automatically by re-using data from similar entities, by information on the light distribution (to remove illumination artifacts), or through template-based proposals. In this thesis, we address this especially in Chapter 5 and Chapter 6, where structural analysis, optimization-based snapping and the use of multimodal data is used to account for missing data, but also in Chapter 3 and Chapter 4, where efficient light transport simulation methods are presented.

Specialized, intelligent user interfaces with interactive performance: Dividing the model to reconstruct into individual structural parts, segments or entities allows offering specialized, simple tools to operate on the 3D data. While the user only has to give approximate, sketch-based instructions, the optimization-based techniques (that rely on the underlying information) automatically provide for an exact solution. Examples for this are shown in the Chapters 5 and 6, where extracted polygons are automatically snapped together to close holes, polygon edges are snapped to edges found in photos, or where similar entities are found in neighboring images. In terms of the needed interactive performance for such approaches, the light simulation approaches presented in the Chapters 3 and 4 provide the required computational performance combined with the needed physical accuracy.

Closing the gap between current automatic and manual modeling paradigms: By applying scene analysis and reconstruction techniques in an interactive modeling and reconstruction package, the advantages of both automatic and manual approaches are combined: The artist can control every aspect and detail of the 3D model, while giving only a minimal set of constraining instructions for the automatic procedures in the background. By providing sketch-based 2D interactions in 3D space for reconstructing and modeling buildings in the Chapters 5 and 6 and a visual guidance indicator helping the user to estimate the currently achieved accuracy, we demonstrate how such approaches can help in the semi-automatic generation of textured, CAD-ready models within a few minutes – even on touch-based devices.

Creating a connection between reconstruction techniques and further modeling: Our proposed approach of combining different types of input data to generate a consistent model in terms of geometry, texture and annotations opens the door towards the use in further modeling tools: The division into entities and the definition of hierarchical relations allows deriving semantic information for the use in BIM systems, providing a new way to use the semantically enriched results without further processing. This is again treated in the Chapters 5 and 6, where CAD-ready models with hierarchical information, occluder-free textures and accuracy information can be exported to arbitrary modelling formats without further processing.

CHAPTER 2

Contributions

The vision of reconstructing buildings by exploiting multiple data sources has existed for several years. It has been approached in the *Semantic Modeling and Acquisition Group* at *VRVis* step-by-step through the execution of several research projects focusing on different aspects. In this chapter, the projects with the largest impact on current and future developments regarding the creation of a multimodal urban reconstruction environment (Section 2.1) as well as the research setting (Section 2.2) are briefly presented. Furthermore, the papers which emerged from these projects, and which act as the basis for the chapters 3 to 6 of this thesis, are summarized.

2.1 Contributions by Research Project

An overview of the collaborative research projects in whose context this thesis was created in, as well as of the generated scientific results, are given in this section.

2.1.1 High Quality Lighting Simulation – HILITE and VAMOS

HILITE[1] and the consecutive project *VAMOS*[2] were research projects within the scope of the *COMET*[3] funding scheme carried out at the *VRVis Research Center* from 2010 until 2016. They were executed in collaboration with the industry partners *Zumtobel Group*[4], *Hefel Wohnbau AG*[5], and *Witsch Visuals GmbH*[6]. The main focus lay on the development of an advanced lighting simulation system, allowing real-time interactions in terms of movement and scene modification. The goal was to provide a fast, dynamic,

[1]https://www.vrvis.at/research/projects/hilite/
[2]https://www.vrvis.at/research/projects/vamos/
[3]https://www.ffg.at/programme/comet-competence-centers-excellent-technologies
[4]http://www.zumtobel.com
[5]http://www.hefel.at
[6]http://www.witsch.net

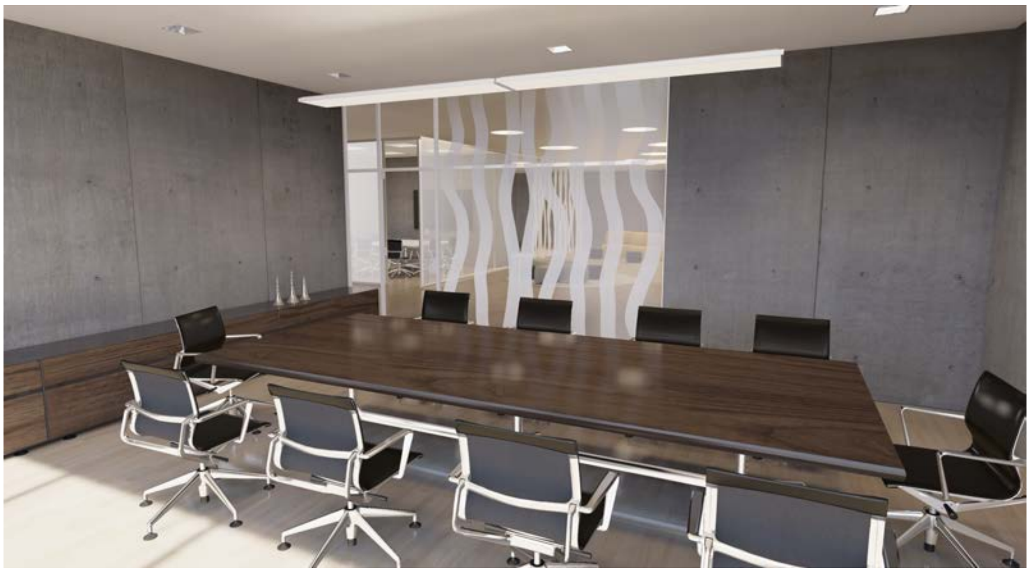

Figure 2.1: A recent screenshot of an office scenario designed and illuminated using the *HILITE* system [LTH^{+}13] with complex material system extensions [LTM^{+}14]. The actual light distribution is performed on the GPU using *Shadow Mapping* techniques. The accumulated light energy is iteratively accumulated in light-map textures. They are constantly updated in the visualization while the simulation is running, providing immediate feedback in interactive modeling sessions.

and easy-to-use way to visualize new lighting concepts in architectural scenarios. The resulting framework makes it possible to give users (and customers) a highly realistic and interactively modifiable preview of the illumination inside and outside of a building.

A novel approach to achieve the required interactivity was to provide means to immediately display a plausible visualization of the light distribution after every modification in the scene (which makes a recalculation necessary, and restarts the simulation). We tackled this problem by exploiting real-time GPU techniques originating from the gaming industry (e.g., *Instant Radiosity* and *Shadow Mapping*), and delved deeply into optimizing the visibility calculation for area or volumetric light sources, i.e., the physically accurate distribution of light from luminaires prevalent in our system with the highest possible speed. This led to the ideas that are presented in the chapters 3 and 4:

Summaries of the published methods this thesis is based on (part 1 of 2):

Fast Accurate Sampling of Area Lights: In Chapter 3, an optimal way (in terms of visual quality and speed) to calculate physically accurate soft shadows is shown. A computationally slow, but correct way of generating soft shadows is to take multiple samples from all over the area light source and to accumulate them. The process can be optimized by taking only as few samples as possible. This

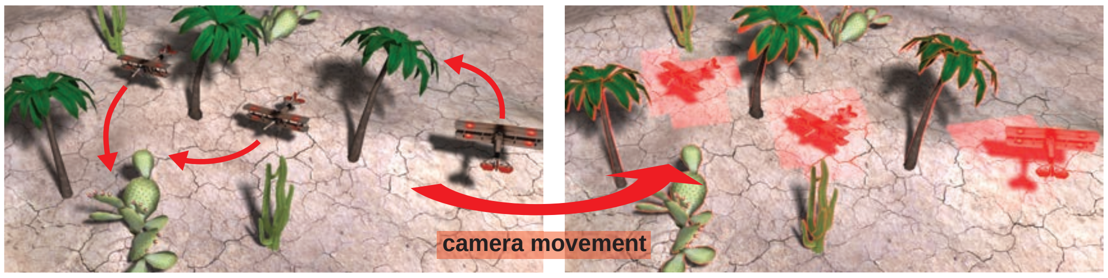

Figure 2.2: These screenshots demonstrate the principle of one of our novel soft shadow algorithms (presented in detail in Chapter 4) developed in the context of the *HILITE* project: The rendering performance of the *Percentage Closer Soft Shadows* [Fer05] method is significantly increased by exploiting the temporal coherence between individual frames. Only the shadows in the areas marked red in the right image have to be re-evaluated. This saves rendering time and doubles the soft shadow rendering performance in real-time 3D scenes with both static and dynamic objects.

number of needed samples depends on the size of the penumbra in screen space. Therefore, we propose a novel adaptive subdivision scheme to divide a rectangular area light source into sub parts, and use hardware occlusion queries to evaluate the need for further samples.

Chapter 3 is based on the publication:

> Michael Schwärzler, Oliver Mattausch, Daniel Scherzer, and Michael Wimmer. Fast Accurate Soft Shadows with Adaptive Light Source Sampling. In *Vision, Modeling and Visualization 2012*, pages 39–46. Eurographics Association, November 2012.

Reusing Soft Shadows in Consecutive Frames: In Chapter 4, the idea of reusing calculated area light source visibility over multiple frames in dynamic scenes is presented. We exploit the temporal coherence prevalent in typical scene movement, making the estimation of a new shadow value necessary only when regions are newly disoccluded. This can be due to camera adjustments, or when the shadow situation changes due to object movements. Through these optimization, we achieve a significant performance increase in typical 3D game scenarios.

Chapter 4 is based on the publication:

> Michael Schwärzler, Christian Luksch, Daniel Scherzer, and Michael Wimmer. Fast Percentage Closer Soft Shadows using Temporal Coherence. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2013 (i3D 2013)*, pages 79–86, New York, NY, USA, March 2013. ACM.

Within the scope of projects *HILITE* and *VAMOS*, the author has also contributed to works that are not part of this thesis: The proposed light transport methods were extended to be used in a newly developed, interactive lighting design prototype, published by Luksch et al. [LTH+13]. The system was later enhanced by a complex material system [MÏ2, LTM+14] (see Figure 2.1). The intermediate illumination results were stored in additional texture layers, and it was assured that the accuracy increased from frame to frame until converging to a physically accurate solution. This way, scene interactions (such as object transformations or movement of luminaries) are always possible during these costly computations — even in case of large numbers of physically complex light sources. The developed system makes it possible to visualize the simulation results on both regular monitors as well as on 3D stereo setups with multiple screens. Hence, the system is capable of providing an immersive impression of the illuminated buildings, while allowing on-the-fly changes in scene and light simulation configurations.

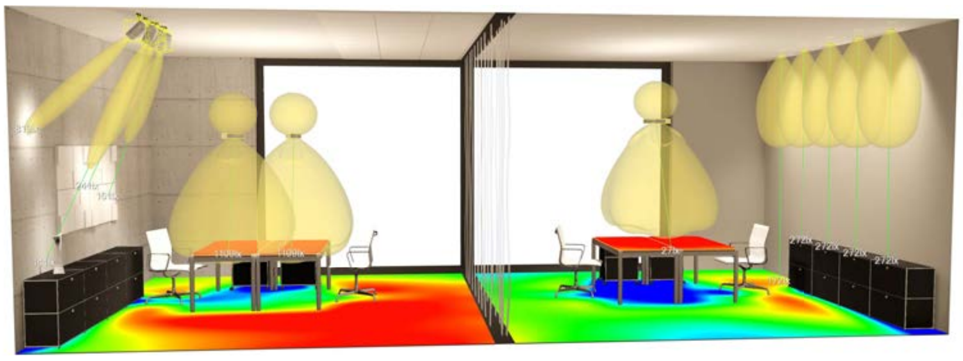

Figure 2.3: The *HILITE* system was extended by tools and methods that help to quantify the quality of lighting solutions, such as the visualization of light source emission profiles, or the placement of measurement surfaces. Especially the use of measurement surfaces increases the designer's productivity by providing direct visual feedback and by allowing an easy observation of industry norms.

Sorger et al. [SOL+16] used the proposed system to evaluate how 3D modeling and simulation could best be employed to optimally support decision finding processes in lighting design. Using tools and methods from Visual Analytics, they enabled weighting and ranking of created lighting scenarios, and made it possible to provide feedback and suggestions concerning light distributions in a 3D modeling tool – a big step towards the linking of the two aforementioned fields.

Several diploma theses (of which some have been further enhanced and published as papers) were supervised. The theses deal with GPU-based ray tracing as alternative approach [Vog13], high-level shader languages [May15, HSM+14, HSM+15], sampling strategies [Cor14, CSLW17], the interactive modeling of light sources themselves [Krö16, KLSW17], and a novel reconstruction workflow for capturing and reproducing lens-flare artifacts in real-time applications [Wal17, WLS+18].

2.1.2 Semantic Modeling and Acquisition for Urban Safety Simulations – REPLICATE

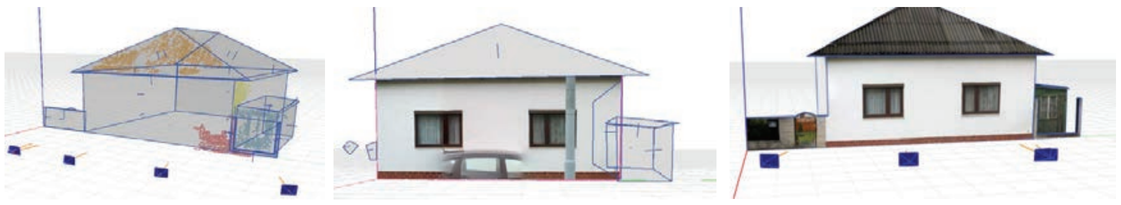

Figure 2.4: The reconstruction and modeling approach in the *REPLICATE* project (the screenshots have been created using the prototype developed in the project): Based on a computed photogrammetric data set consisting of oriented photos and a corresponding sparse, erroneous point cloud, a low-polygonal, CAD-ready geometric 3D model is created by exploiting detected planar segments, computed adjacency relations, extracted image edges and interactive snapping processes (left). The photos are first simply reprojected onto the geometry (middle). Through application of interactive brushing and/or texture synthesis methods, building surfaces free from stitching artifacts and occluders are created (right).

REPLICATE was carried out from 2012 until 2015 in parallel to *HILITE* and *VAMOS* as a collaborative *FIT-IT*[7] research project with the *Institute for Computer Graphics and Algorithms*[8] at *TU Wien* and the industry partners *ViewApp*[9] and *VCE Vienna Consulting Engineers ZT GmbH*[10], both specialized in security- and safety-relevant real-time simulations. The project was driven by the fact that state-of-the-art algorithms and tools were not capable of creating faithful replicas of real-world urban environments in an economically viable way – a fact that remains true until today, even though the situation has already improved since. *REPLICATE* helped to make huge steps towards the efficient acquisition of such models at economically relevant scales, and to provide content used in safety- and security-relevant real-time applications – such as the bus traffic or earthquake simulations our industry partners are carrying out (see Figures 2.5 and 2.6).

The chosen approach was to unify different categories of input data (measured point clouds, geometry, and image data) into a common context, which allowed exploitation and cross-correlation of semantic information and spatial relations (see Figure 2.4). The demand for additional contextual information continues to rise due to increasing complexity of available data and increasingly complex use-cases built on top of these data. Semantic information and annotations are not only important in further application

[7]http://www.fit-it.at/
[8]http://www.cg.tuwien.ac.at
[9]http://www.viewapp.at/
[10]http://www.vce.at
[11]http://www.omnibussimulator.de/

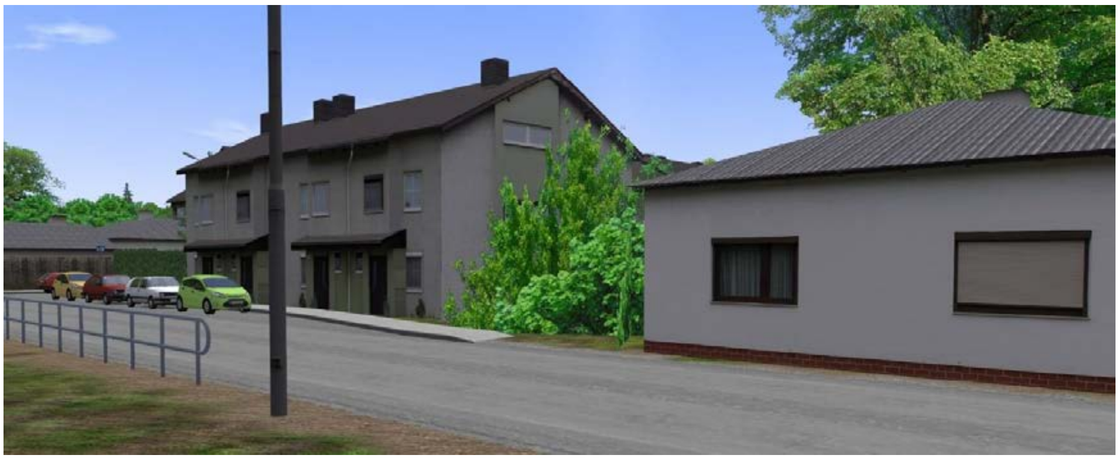

Figure 2.5: Buildings created with the prototypes developed in the *REPLICATE* project are used in a bus simulation software (OMSI[11]). Notice the low-polygonal geometry, perfectly suited for real-time applications, and the textures generated from the photos: Even though they don't have any seams, they still suffer from shadowing artifacts, as the illumination contained in the photos has not been removed.

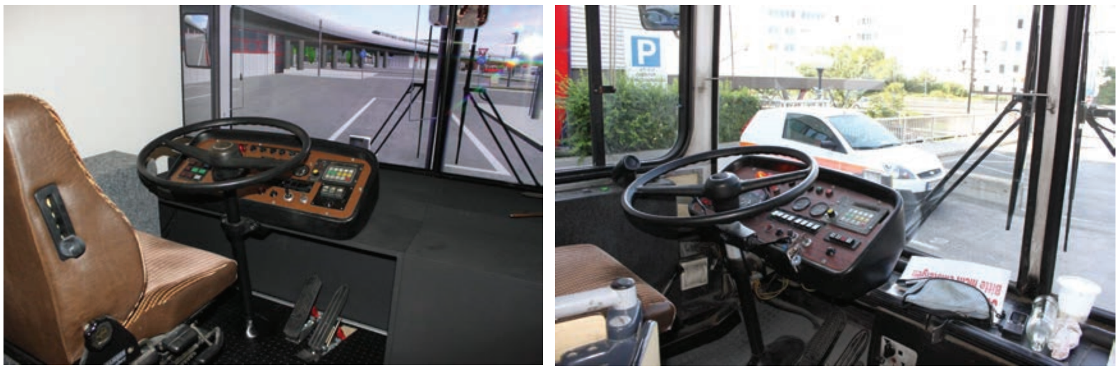

Figure 2.6: During the project, a hardware bus simulator (left) was built by our company partner *ViewApp* in order to increase both the immersion as well as the training effect. The bus simulation software including the reconstructed 3D buildings was coupled with the simulator hardware, perfectly replicating the behavior of driving a real bus through Vienna (right).

or simulation steps, but can also provide valuable help during the modeling process. Exactly this additional extracted semantic information, combined with novel, suggestion-based user-interaction methods tailored towards these reconstruction tasks, led to the development of prototypes that are presented in the chapters 5 and 6 of this thesis:

Summaries of the published methods this thesis is based on (part 2 of 2):

Interactive Polygon Snapping for 3D Building Reconstruction: In Chapter 5, a new 3D reconstruction and modeling paradigm called *O-Snap* is presented, which aims at semi-automatically creating low-polygonal, CAD-ready 3D building models from point clouds. This is achieved by first analyzing and segmenting the data into planar shapes, for which coarse polygons are extracted. This polygon soup is then automatically closed by an optimization-based snapping process wherever possible. In all other cases, the previously performed segmentation is exploited interactively: All manual modeling operations (that are necessary due to erroneous data) can be performed in simple 2D on the planes, while the snapping process further supports the alignment of the geometry.

Chapter 5 is based on the publication:

> Murat Arikan, Michael Schwärzler, Simon Flöry, Michael Wimmer, and Stefan Maierhofer. O-Snap: Optimization-Based Snapping for Modeling Architecture. *ACM Transactions on Graphics*, 32:6:1–6:15, January 2013.

Please note that this publication has a comparatively large scope and extent, and the work for the main contributions has therefore been split between the first and second author. While Murat Arikan focused on the development of the optimization-based snapping process, the application of these methods as a novel, guided 2D tool in a 3D modeling context was for the most parts proposed, realized and evaluated by the author of this thesis, and has to be seen as the contribution for this work.

Sketching 3D Buildings using Oriented Photos: In Chapter 6, the ideas of the previous chapter are extended to photogrammetric data sets and the exploitation of the corresponding image data: Again, all interactions are performed in 2D, but this time directly on photos. The underlying extracted geometric relations and the planar segmentation create a 3D representation of the building to reconstruct, while snapping and accuracy indicators guide the user through the modeling process. This easy-to-use user interaction was demonstrated to be applicable even on touch-based interfaces.

Chapter 6 is based on the publication:

> Michael Schwärzler, Lisa-Maria Kellner, Stefan Maierhofer, and Michael Wimmer. Sketch-based Guided Modeling of 3D Buildings from Oriented Photos. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D 2017)*, pages 9:1–9:8. ACM, February 2017.

2.2 Research Setting

In the *Semantic Modeling and Acquisition Group*, special emphasis is put on the following aspects during the execution of collaborative research projects:

Tight coupling with scientific partners: As a research center oriented towards the industry, it is of utter importance to not lose the focus on participating in the latest ground-breaking findings in basic research topics. In order to achieve this, we rely on our collaborations with various universities worldwide, making it possible to have access to state-of-the-art methods, and bring them to the application-oriented fields. In particular, several cooperative research projects and publications with the *Institute for Computer Graphics and Algorithms* at the TU Wien have proven to be a guarantee for success.

Tight coupling with industry partners: Apart from being driven by actual requirements that arise in industrial applications, one major aspect that has a remarkable benefit on the research is the availability of real-world data that our industry partners provide: The access to data sets from actual application scenarios – may it be 3D point cloud data, measured light sources, accurately measured reference geometry or materials, measurements of illumination results, photos, etc. – are used to test and verify research results (see Figure 2.7). Furthermore, industry experts and domain users provide valuable feedback and assure that the conducted research is actually applicable and suitable for solving the industry's problems.

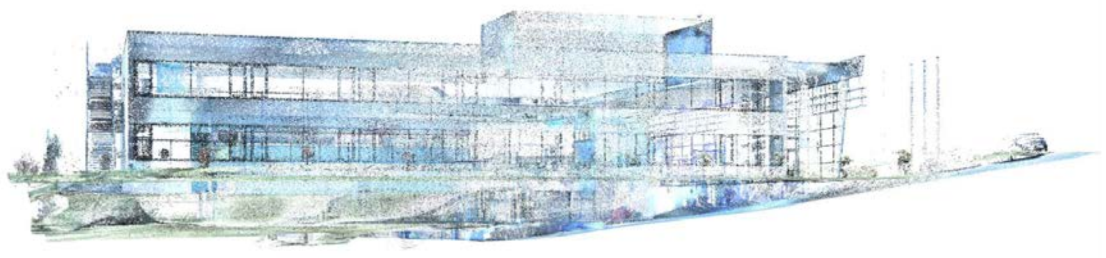

Figure 2.7: Screenshot of a visualized point cloud data set representing the Technologiezentrum in Pinkafeld (Austria), acquired using laser scanning technologies. It consists of approximately 500.000.000 points and has a file size of more than 40GBs. This data set resembles the typical data that our industry partners are confronted with. The researchers at *VRVis Research Center* are provided with such data sets throughout the projects.

Professional software development environment: In order to be able to bridge this gap between conducting and integrating novel research while at the same time delivering results in the form of software prototypes to the industry, it is a necessity to reach an excellent balance between rapid prototyping and high software quality. We therefore strive for agile, iterative, short development cycles to be able to integrate new algorithms and methods, while at the same time we are able to react to our industry partner's feedback and requests within a minimal time span. It is mandatory to make use of software platforms whenever possible to exploit project-overarching synergies, reuse results and enable efficient teamwork: A key asset in this attempt is the use of the Aardvark platform[12] – an open source framework for scientific purposes that has been used by over 60 projects in the past. It offers both the possibility to develop and use robust software libraries, may it be for commercial purposes or for the release of state-of-the-art algorithms that are part of scientific publications.

2.3 List of Publications

In summary, the publications that build the foundation for this thesis and that emerged from the aforementioned projects are:

- Michael Schwärzler, Oliver Mattausch, Daniel Scherzer, and Michael Wimmer. Fast Accurate Soft Shadows with Adaptive Light Source Sampling. In *Vision, Modeling and Visualization 2012*, pages 39–46. Eurographics Association, November 2012
- Michael Schwärzler, Christian Luksch, Daniel Scherzer, and Michael Wimmer. Fast Percentage Closer Soft Shadows using Temporal Coherence. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2013 (i3D 2013)*, pages 79–86, New York, NY, USA, March 2013. ACM
- Murat Arikan, Michael Schwärzler, Simon Flöry, Michael Wimmer, and Stefan Maierhofer. O-Snap: Optimization-Based Snapping for Modeling Architecture. *ACM Transactions on Graphics*, 32:6:1–6:15, January 2013
- Michael Schwärzler, Lisa-Maria Kellner, Stefan Maierhofer, and Michael Wimmer. Sketch-based Guided Modeling of 3D Buildings from Oriented Photos. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D 2017)*, pages 9:1–9:8. ACM, February 2017

Furthermore, the author co-authored the following peer-reviewed publications in the context of light-transport, urban reconstruction and modeling:

- Daniel Scherzer, Michael Schwärzler, Oliver Mattausch, and Michael Wimmer. Real-Time Soft Shadows Using Temporal Coherence. In *Advances in Visual Computing: 5th International Symposium on Visual Computing (ISVC 2009)*, Lecture Notes in Computer Science, pages 13–24. Springer, 2009

[12]https://github.com/aardvark-platform

- Przemyslaw Musialski, Christian Luksch, Michael Schwärzler, Matthias Buchetics, Stefan Maierhofer, and Werner Purgathofer. Interactive Multi-View Façade Image Editing. In *Vision, Modeling and Visualization 2010*, pages 131–138, November 2010

- Irene Reisner-Kollmann, Christian Luksch, and Michael Schwärzler. Reconstructing Buildings as Textured Low Poly Meshes from Point Clouds and Images. In *Eurographics 2011 - Short Papers*, pages 17–20, April 2011

- Christian Luksch, Robert F. Tobler, Ralf Habel, Michael Schwärzler, and Michael Wimmer. Fast Light-Map Computation with Virtual Polygon Lights. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2013*, pages 87–94. ACM, March 2013

- Christian Luksch, Robert F. Tobler, Thomas Mühlbacher, Michael Schwärzler, and Michael Wimmer. Real-Time Rendering of Glossy Materials with Regular Sampling. *The Visual Computer*, 30(6-8):717–727, June 2014

- Johannes Sorger, Thomas Ortner, Christian Luksch, Michael Schwärzler, Meister Eduard Gröller, and Harald Piringer. LiteVis: Integrated Visualization for Simulation-Based Decision Support in Lighting Design. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):290–299, January 2016

- Katharina Krösl, Christian Luksch, Michael Schwärzler, and Michael Wimmer. LiteMaker: Interactive Luminaire Development using Progressive Photon Tracing and Multi-Resolution Upsampling. In *Vision, Modeling and Visualization 2017*. The Eurographics Association, 2017

- Andreas Walch, Katharina Krösl, Christian Luksch, David Pichler, Thomas Pipp, and Michael Schwärzler. An Automated Verification Workflow for Planned Lighting Setups using BIM. In *REAL CORP 2018, Proceedings*, REAL CORP, pages 55–65, 2018

- Andreas Walch, Christian Luksch, Attila Szabo, Harald Steinlechner, Georg Haaser, Michael Schwärzler, and Stefan Maierhofer. Lens Flare Prediction based on Measurements with Real-time Visualization. *The Visual Computer*, May 2018

- Katharina Krösl, Dominik Bauer, Michael Schwärzler, Henry Fuchs, Michael Wimmer, and Georg Suter. A VR-based User Study on the Effects of Vision Impairments on Recognition Distances of Escape-route Signs in Buildings. *The Visual Computer*, 34(6):911–923, Jun 2018

CHAPTER 3

Fast Accurate Sampling of Area Lights

This chapter is based on the publication:

Michael Schwärzler, Oliver Mattausch, Daniel Scherzer, and Michael Wimmer. Fast Accurate Soft Shadows with Adaptive Light Source Sampling. In *Vision, Modeling and Visualization 2012*, pages 39–46. Eurographics Association, November 2012.

The original paper was adapted in terms of formatting and type-setting to fit this template and to increase readability. The introduction was adjusted to fit the topic of this thesis, and the abstract was removed. Minor corrections, such as fixing typos or unclear wording, were applied. The original version is available at

https://diglib.eg.org/handle/10.2312/PE.VMV.VMV12.039-046.

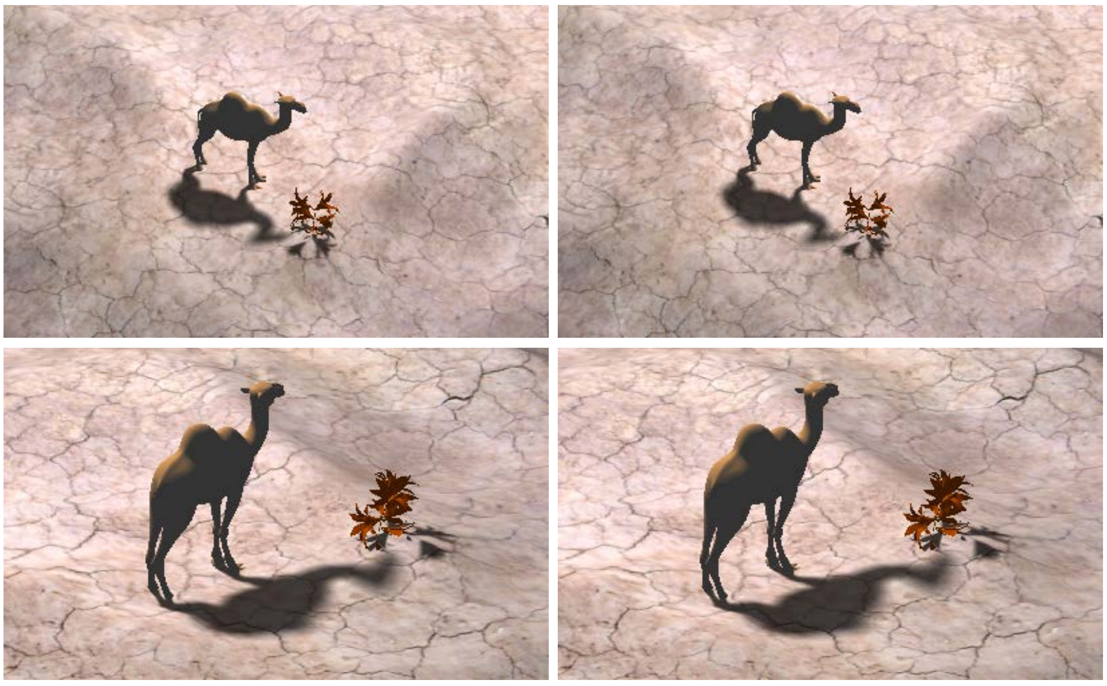

Figure 3.1: Our proposed method is capable of selecting and rendering a significantly reduced amount of shadow maps needed for a physically correct soft shadow solution using an adaptive light source subdivision. *Top Left:* Scene rendered from far with 289 fixed samples (10 FPS). Top Right: The same view point rendered with our method with only 25 samples (67 FPS). *Bottom Left:* The same scene, rendered from a closer view point with 289 fixed samples (10 FPS). *Bottom Right:* Our method reduces the number of needed samples to 105 (18 FPS).

3.1 Introduction

In 3D modeling scenarios, shadows provide an improved depth perception and increase the realism, which makes it easier and faster for an artist to accurately create the desired model. Therefore, algorithms for hard shadow rendering are widely used in today's games and applications. Nevertheless, hard shadows – which are cast by an infelicity small point light source – hardly appear in reality, as nearly every light source (including the sun) has a certain extent, leading to the generation of *soft shadows*. They consist of umbra (areas where the light source is completely blocked) and penumbra (areas where the light source is partly visible) regions. In the case of *Multimodal Urban Reconstruction and Modeling*, an accurate calculation of the distributed light energy plays an important role, for example during the removal of shadows in textures generated from photos of buildings under certain lighting conditions. Moreover, using soft shadows in rendering applications significantly increases the realism of the generated images (see Figure 3.2), and inherent shadow map artifacts like aliasing at the shadow boarders are often hidden through the low frequency soft shadows.

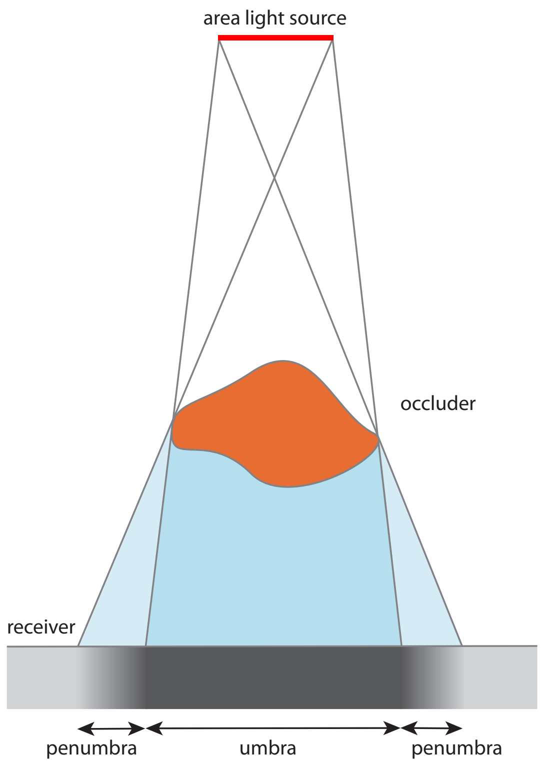

Figure 3.2: An area light source leads to a soft shadow, which consists of umbra and penumbra regions.

In contrast to hard shadows, the fast and correct calculation of soft shadows is a complex task and still an area of active research. Especially the increased computational costs that are involved in the calculation of physically correct soft shadows often prevent their use in real-time applications. In this work, we suggest a novel approach (presented in Section 3.3) based on the idea of sampling the light source several times in order to obtain *physically correct* soft shadows: We optimize the number of sampling points needed for satisfying results by starting with only very few sampling points and adaptively adding more and more of them, depending on whether the sampling density is already high enough or not. This decision is made by projecting the shadow maps from four sample points forming a quad on the area light to the view point of the camera, and by comparing there how much they differ using hardware occlusion queries. Only if the space between the individual shadow boundaries is too large (i.e., banding artifacts are visible), the quad on the light source is subdivided, and new sampling points are added on the next level(s).

After creating n shadow maps by applying our adaptive sampling strategy and the corresponding weights, we discuss how they can be used to render physically accurate soft shadows at interactive or even real-time frame rates using both *deferred rendering* as well as *texture arrays* in our rendering framework (see Section 3.3.4).

3.2 Related Work

A vast number of real-time soft shadow algorithms has been published during the last few years, most of them based on extensions to the shadow mapping algorithm (see Section 3.3.1) or the shadow volumes algorithm introduced by [Cro77]. We will therefore focus on the most relevant publications for our work (see [ESAW11] for an extensive overview).

Since the calculation of physically correct soft shadows is generally considered too costly for real-time application, most soft shadow approaches for interactive or real-time applications estimate the complex area visibility (i.e., the amount of the area light source that is visible from a point on a surface) by calculating a single hard shadow from the center of the area light source, and simulate the penumbra using approximative heuristics. The simplifications used in these so-called *single sample approaches* will in general not result in physically correct soft shadows.

In [WH03], not only a shadow map, but also a so-called *Penumbra Map* is generated by analyzing the objects silhouettes from the position of the light source, allowing a penumbra region to be estimated in the illumination pass. [Fer05] suggests using a technique called *Percentage Closer Soft Shadows (PCSS)*, where *Percentage Closer Filtering (PCF)* by [RSC87] is applied and combined with a blocker search: PCF softens hard shadow boundaries by not only comparing the current depth to a single value in the shadow map, but by doing so with the neighboring pixels in the shadow map as well. The percentage of successful shadow tests specifies the shadow intensity. It helps to reduce aliasing artifacts at the softened shadow boundaries, but the penumbra is far from being accurate, as it always has the same size. PCSS therefore uses an additional blocker search in the shadow map, so the filter kernel can be adjusted according to the relation between light, blocker and receiver. To avoid the vast number of shadow map lookups for PCF several pre-filtering methods have been proposed [DL06, AMB$^+$07] that allow real-time frame-rates.

Several papers [GBP06, GBP07, AHL$^+$06, ASK06, SS07] have recently been published, which propose variants of a technique called *backprojection*. The idea is to use a single shadow map not only for depth comparison, but to employ it as a discretized representation of the scene. In order to calculate the visibility factor v for a screen-space pixel p, the shadow map texels are backprojected from p onto the light source, where the amount of occlusion is estimated. These approaches can produce more accurate results than PCSS and variations thereof, but are prone to artifacts (e.g., in cases when occluders overlap, when the light source is too close, or when the penumbra is extremely large) and one may have to backproject a huge number of shadow map texels, which is costly.

The most intuitive, but also slowest approach to generate *physically correct* soft shadows is to generate hard shadows from several sampling points on the area light source and accumulate this information (see Section 3.3.2). In order to minimize computation time, [HH97] suggest using only a few regularly distributed samples for the calculation. For each shadow receiver, a so-called *attenuation map* is computed by summing up the individual shadows, which is then used to modify the illumination of the object. So, for n sampling points and m receivers, $m \times n$ shadow maps are required. An improvement of this idea has been suggested by [ARHM00]: Instead of calculating and using an attenuation map for each receiver, a *single layered attenuation map* for the whole scene is created, which allows interactive frame rates on modern graphics hardware. In the method proposed by [SAPP05], the visibility information of many shadow maps is combined into a precomputed compressed 3d visibility structure, which is then used for rendering. Employing CUDA support for irregular data structures, [SEA08] compute accurate soft shadows by evaluating the shadow solution for each visible pixel in screen. [SSMW09] sample the light source over multiple frames exploiting temporal coherence. Although they show cases where they converge to the physical correct result, they have problems with quickly moving objects and can therefore not guarantee correct results in all scene configurations.

Real-Time soft shadows can also be simulated with modified versions of the shadow volumes algorithm, in particular methods based on the Penumbra Wedges algorithm [AAM03, FBP06].

Our algorithm is based on the approaches which use multiple shadow maps per light, but we propose a novel adaptive sampling strategy in order to minimize both the number of shadow maps needed to obtain high quality soft shadows and the rendering time per frame.

3.3 The Algorithm

In this section, we introduce our adaptive refinement strategy for the sampling of area light sources, and most importantly, our GPU-based subdivision evaluation criterion. Additionally, we discuss possible ways to render the (potentially) large amounts of generated shadow maps.

3.3.1 The Shadow Mapping Algorithm

Shadow mapping is an image-based algorithm first introduced by [Wil78]. Its basic idea is to view the scene from the position of the light source in a first pass, and store the depth values of the fragments in a texture (called the *shadow map*). The shadow map therefore contains the distances to all sampled surface points which are illuminated by the light source.

In the second pass, the scene is rendered from the camera's point of view. Every fragment is transformed into light space, where its distance to the light source is compared to the corresponding value in the shadow map. If the distance to the current fragment is larger

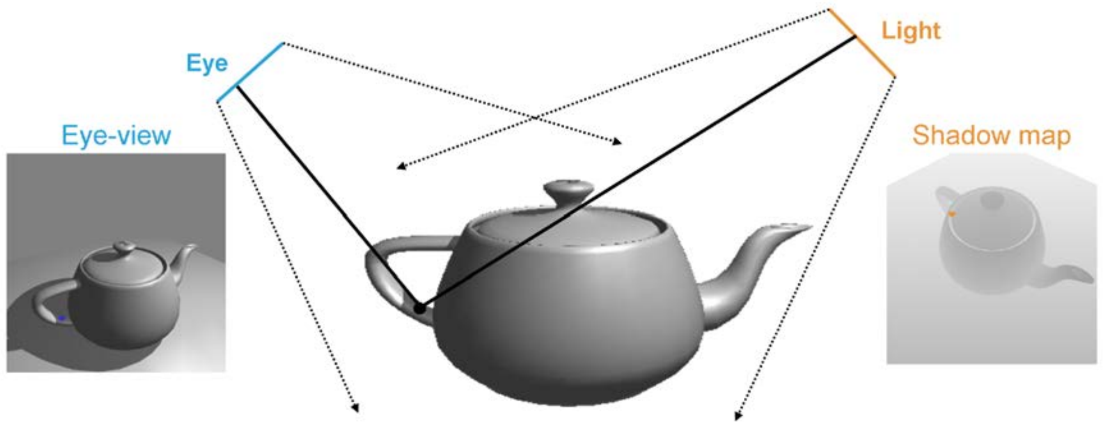

Figure 3.3: The shadow mapping algorithm: The depth values as seen from the light source are stored in a shadow map, and are then used in a second pass to generate shadows on the objects (image from [Sch05].

than the shadow map value, it lies in shadow; otherwise it has to be illuminated by this light source. Figure 3.3 illustrates the basics of the algorithm.

3.3.2 Estimating Soft Shadows with Light Source Sampling

An area light source can be approximated by n different point light source samples. A shadow map allows us to evaluate for every screen space fragment if it is illuminated by its associated point light.

$$\tau_i(x,y) = \begin{cases} 0 & \text{lit from point light } i \\ 1 & \text{in shadow of point light } i \end{cases} \tag{3.1}$$

$\tau_i(x,y)$ is the result of the hard shadow test for shadow map i for the screen space fragment at position (x,y). Under the assumption that the point sampling on the area light source is dense enough (i.e., n is high enough), the soft shadowing result ψ (i.e., the fractional light source area occluded from the fragment) can be estimated by the proportion $\hat{\psi}_n$ of shadowed samples

$$\hat{\psi}_n(x,y) = \frac{1}{n}\sum_{i=1}^{n} \tau_i(x,y). \tag{3.2}$$

3.3.3 Adaptive Refinement of the Sampling Density

Generating soft shadows with multiple shadow maps per light is computationally expensive due to the high sampling density which is required to render smooth, visually appealing penumbra regions. If the density is too low, banding artifacts are likely to appear, and the human visual system does not perceive a soft shadow anymore, but several hard shadows (see Figure 3.9).

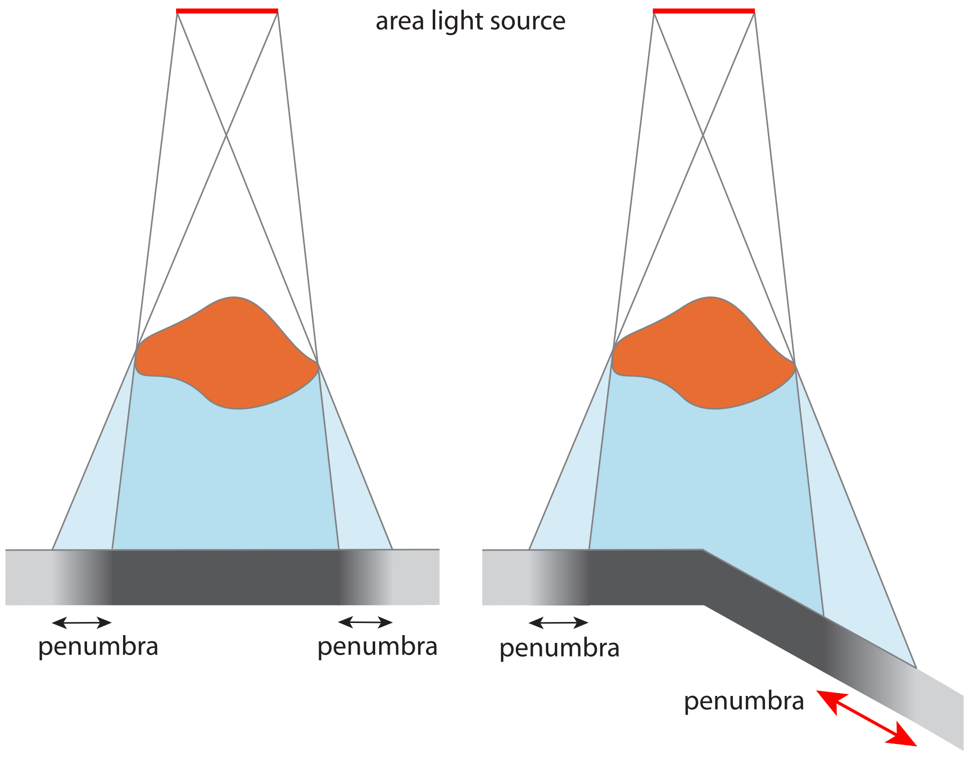

Figure 3.4: A slight change in the receiver geometry can cause a significant increase of the penumbra size.

The larger a penumbra is, the more samples are necessary to create smooth transitions between the individual hard shadows. The minimum required sampling density is not easy to predict, though: It depends on the relation between light, blocker and occluder. As can be seen in Figure 3.4, a slight rotation of the receiver geometry leads to a drastic increase of the penumbra size, making more samples necessary. Due to the perspective projection, the camera's point of view plays a major role here as well, as it determines the size of the penumbra in screen space: if the camera is very close to the shadow, the penumbra region can be as large as the whole frame buffer.

To avoid redundant shadow map computations caused by using a constant high sampling density only required in some worst cases, we suggest selecting the sampling points adaptively whenever the scene configuration or the camera position changes.

Generating Shadow Maps

The first step in our algorithm is to create the initial shadow maps at the corners of the area light source. These are the only shadow maps which are always generated; all the others are only computed if necessary (see Section 3.3.3). We assume a square area light source for an easier explanation in this work, but similar subdivision strategies can be found for other kinds of light sources, as our splitting criterion is independent of the actual subdivisions performed. The shadow maps are generated from the sampling points using standard uniform shadow mapping with a perspective projection.

Reprojection

After the creation of the initial sampling points, we project the shadow maps into the same space in order to compare them. It is important that the refinement is dependent on the observer's position and the view: For example, it makes no sense to refine a soft shadow which is far away and hardly visible, while for shadows very close to the camera, it is important to have more samples in order to obtain a smooth penumbra. We therefore project the shadow maps into camera space, where a comparison makes such a view-dependent refinement possible.

The reprojection step is done similar to the second step in the regular shadow mapping algorithm, but instead of using the shadow values from the shadow map for illumination, they are directly used for comparisons as described in Section 3.3.3. In order to generate the correct subdivision level needed for the current screen buffer size, the comparison render target extents must have the same dimensions. If a smaller amount of shadow maps is desired (at the cost of physical accuracy, leading to banding artifacts), the resolution of the comparison render targets can be lower (see Section 3.3.4).

Subdivision Evaluation

The comparison of four neighboring shadow maps in camera space is done in a pixel shader by applying a 2-pass strategy: In the first pass, the reprojected depth values of the four shadow maps are evaluated as in the original shadow map algorithm: For each screen space fragment, the 4 corresponding shadow values are calculated and summed up (i.e., each fragment obtains an integer value between 0 and 4), and stored in the comparison render target.

In the second pass, the stored accumulated shadow values are used to identify potential regions that produce banding artifacts: banding artifacts appear whenever the distances between the hard shadow borders are too large, so that the shadow is perceived as multiple hard shadows instead of a single soft shadow. We therefore investigate the 8-connected neighborhood of each penumbra texel (indicated by a texel with a value between 1 and 3) in the comparison render target texture, and check if there is at least one neighboring texel that has a different value. If this simple condition is fulfilled, the subdivision level is assumed to be sufficient for this texel; otherwise, the area light source has to be subdivided further.

In order to quickly evaluate the need for a subdivision, we exploit the functionality of *hardware occlusion queries* [BMH98, Ope07], which are usually used to evaluate visibility by counting the number of pixels drawn on the screen. By discarding all fragments for which the subdivision level is sufficient, the remaining fragments can efficiently be counted. If at least one pixel is output, the area light source needs further refinement in this frame. Note that similar to lowering the resolution of the comparison render target as explained in Section 3.3.3, increasing this threshold and tolerate a few fragments causing banding artifacts can also help to reduce the number of shadow maps.

Generating Additional Sampling Points

If the subdivision evaluation suggests creating a further refinement level on the area light source, new sampling points (and the corresponding shadow maps) have to be created. In the case of a two-dimensional rectangular area light source, we suggest using a quadtree-like structure: If the comparison step makes a subdivision necessary, the rectangle is split into 4 sub-quads, and new shadow maps are generated on all new corners (See Figure 3.5).

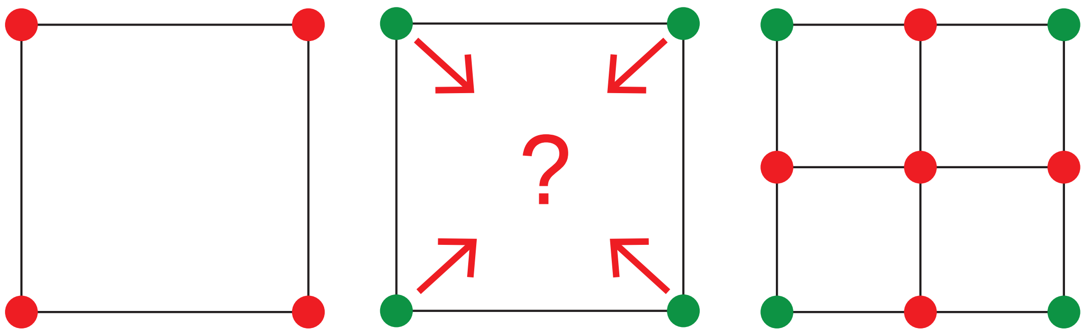

Figure 3.5: Subdividing a rectangular area light source: *Left:* Generate sampling points at the quad corners. *Middle:* Compare corresponding shadow maps in a common projection center (camera space). *Right:* If necessary, subdivide the quad into 4 sub-quads, repeat steps for each sub-quad.

For the new subdivision level, the whole procedure is repeated again: Shadow maps are generated from the new sampling points' positions, and are compared to their quad neighbors. This refinement process is repeated until either the sampling density is high enough in all areas to fulfill the condition defined in Section 3.3.3, or a predefined maximum number of shadow maps has been created.

3.3.4 Evaluating the Shadow Map Information

After the computation of the shadow maps, their contribution must be evaluated in an illumination render pass. This step is basically similar to the second render pass in the standard shadow mapping algorithm. Still, difficulties can arise due to differing subdivision depths (Section 3.3.4) and due to the large amount of depth textures which have to be sampled (Section 3.3.4).

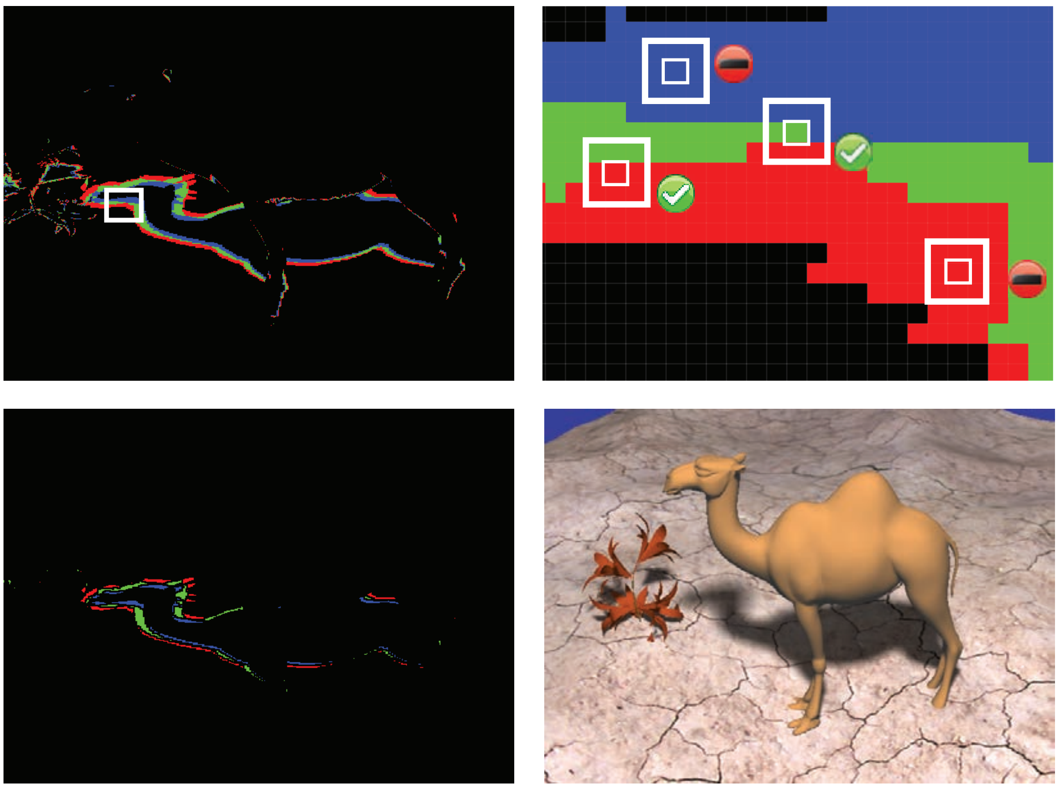

Figure 3.6: Test for further subdivision. *Top Left*: The shadow values of 4 shadow maps in a quad are projected to camera space and accumulated. For visualization purposes, the amount of received shadow has been color-coded: red = 1, green = 2, blue = 3, black = 0 or 4. *Top Right:* Close-up view of the marked region in the left image. For each fragment with a value from 1-3, the 8-connected neighborhood is tested for different values (green tick). If no different value is found, the distance between the shadow maps is too large, and the test fails (red symbol). *Bottom Left:* The fragments that failed the test are drawn in the second pass, and counted using a *hardware occlusion query*. If at least one pixel is drawn, the light source needs to be subdivided. *Bottom Right:* Final result after subdivision.

Assigning Shadow Map Contribution Weights

If all shadow maps generated with our refinement strategy contribute to the final soft shadow solution with equal weight, the darkness of the penumbra can sometimes vary slightly from the exact solution, if the distribution of the adaptively selected sampling points varies significantly. We therefore apply weights on the sampling points: In areas of many subdivision, the individual samples are assigned a smaller weight, and will not contribute as much to the darkness of the penumbra as the ones with a large weight.

In case of a 2D area light source that is subdivided as proposed in Section 3.3.3, the weight ω_i assigned to the i^{th} shadow map is calculated with

$$\omega_i = \frac{1}{(2^d + 1)^2}, \tag{3.3}$$

where d is the subdivision depth. The sum of all weights is 1 if all samples reach the same subdivision depth d. Otherwise, the weights have to be normalized to make sure the final accumulated shadow values lie between 0 (fully lit) and 1 (fully shadowed).

Soft Shadow Visualization using n Shadow Maps

For the calculation of soft shadows, the information from all generated shadow maps has to be checked for each screen space pixel. The hard shadow test values (0 or 1) from all shadow maps i are multiplied with their weight ω_i and summed up, resulting in an estimate for the percentage of occlusion. If the number of shadow maps is high, this can lead to problems due to the limited amount of textures that can be sampled in a single rendering pass.

A way to solve this is to make use of a *deferred rendering* system introduced by [DWS$^+$88] as well as a so-called *accumulation buffer*, which is a screen-space buffer with a single data channel. For each shadow map, we render the scene in a separate render pass. Instead of using the obtained hard shadow value of a screen space fragment $f(x, y)$ directly for illumination, we multiply it with its weight and add it to the accumulation buffer at the position $f_{acc}(x, y)$. A preliminary depth pass helps to ensure that only shadow values from “valid” (i.e., visible) fragments contribute to the accumulation buffer.

After n render passes, all shadow maps have been evaluated, and the accumulation buffer is filled. Now, in a final rendering pass, the scene is illuminated: For each screen space fragment $f(x, y)$, the corresponding accumulation buffer value $f_{acc}(x, y)$ is sampled and used as the occlusion percentage. Note: Since current graphics hardware does not support read and write operations on render targets at the same time, two instances of the accumulation buffer have to be created and swapped each rendered frame, resulting in an additional need for memory on the GPU.

Alternatively, the introduction of so-called *Texture Arrays* in newer graphics APIs makes it possible to send up to 512 textures with the same size and format to the shader, where they can be sampled arbitrarily. This functionality is perfectly suited for our purposes, as it allows us to sample many shadow maps from within the same pixel shader instance. The current fragment’s occlusion value can therefore be obtained without the need for additional passes, saving n read/write operations as well as the memory previously consumed by the accumulation buffer.

Filtering

As already stated in Section 3.3.3, the resolution of the comparison render target can be defined to be smaller than the frame buffer resolution in order to trade physical accuracy for a lower number of sampling points (and therefore higher performance). Since fewer shadow maps are generated, banding artifacts are more likely to become visible. Similar problems occur if the camera is extremely close to a penumbra, so that the maximum number of shadow maps is not sufficient to generate an appealing penumbra region, or if a few pixels with banding artifacts are allowed during the GPU-based splitting evaluation (See Section 3.3.3).

In order to improve the smoothness of the transitions between the individual shadow maps, we therefore suggest sampling them using a small PCF kernel in such situations. PCF filtering softens the shadow boundaries, and a version with a 2x2 kernel can be used on modern graphics hardware without performance hit.

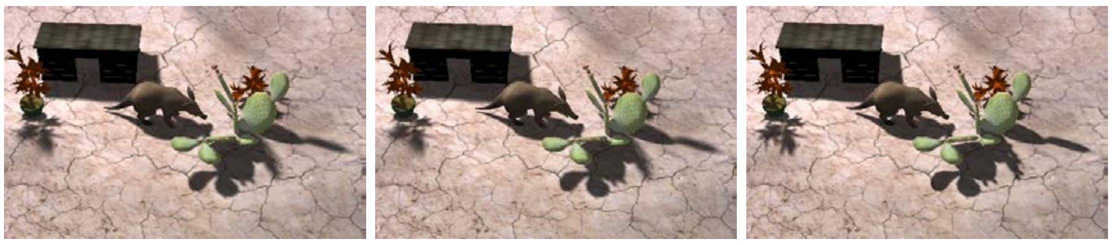
Figure 3.7: Visual Comparison of our approach. Left: Regular sampling with 289 shadow maps, acting as ground truth for our comparisons (8 FPS). Middle: Our method, 93 shadow maps (17 FPS). Right: PCSS soft shadow solution with visibility calculated from only a single shadow map (64/64 samples for blocker search/filtering step, 370 FPS). See Figure 3.8 for a visualization of the differences.

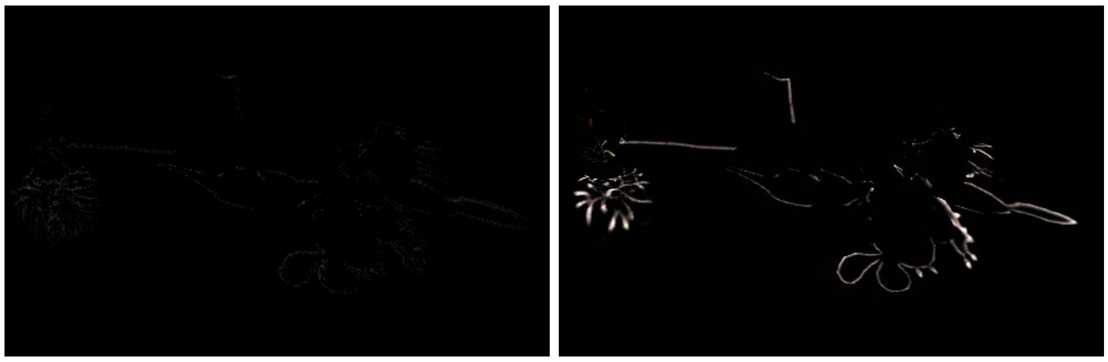
Figure 3.8: Left: Difference image between ground truth (Figure 3.7, Left) and our approach with 93 shadow maps (Figure 3.7, Middle), scaled by factor 5 for visualization purposes. Right: Difference image between ground truth (Figure 3.7, Left) and PCSS (Figure 3.7, Right), scaled by factor 5 for visualization purposes.

3.4 Results and Evaluation

All tests and images in this work were calculated with a comparison render target buffer size of 1024×768p, and a shadow map size of 512^2. The system on which we were testing our approach consisted of an Intel Core i7-920 Processor with 4 Cores, 6GB RAM, and a NVidia Geforce 580GTX with 1.5 GB Memory.

3.4.1 Implementation

We implemented both rendering methods described in Section 3.3.4 in a DirectX 10 rendering framework application using a two-dimensional rectangular light source. For the shadow maps, we use 32Bit floating point textures with a size of 512^2, and store the depth linearly. The maximum allowed number of shadow maps generated by our subdivision strategy is 289, representing a subdivision depth of 4 levels. For the deferred rendering implementation, we use 32Bit floating point textures with the same dimensions as the frame buffer for both the accumulation buffer as well as for the needed depth buffer.

The implementation using texture arrays to evaluate the shadow illumination performs slightly better (approximately 10% faster) than the deferred rendering solution, since the shadow map evaluation can be done in a single pass, and no additional read/write operations on the accumulation buffer are needed. Still, the deferred rendering solution seems to be an acceptable alternative for the application of our method in rendering systems using older APIs.

3.4.2 Visual Comparison

As can be seen in Figure 3.1, Figure 3.7, Figure 3.8, Figure 3.10 and Figure 3.11, the achievable visual quality of our proposed solution with only a few shadow maps is nearly identical to images with a significantly higher (fixed) amount of sampling points. In Figure 3.7, we also show the shadow solution computed by the PCSS method (with 64 samples for the blocker search and 64 samples for the filtering step). Since there the visibility is calculated using only a single shadow map from the center of the light source, the resulting shadow differs significantly from our solution computed with correct visibility (see Figure 3.8).

In Figure 3.9, we show the illumination results generated with a smaller comparison render target, leading to banding artifacts due to the lower number of shadow maps. These artifacts can easily be hidden by applying a simple PCF filter, but the physical accuracy is of course negatively affected by this approximation. Figure 3.10 and Figure 3.11 demonstrate the achievable speed-up that can be gained by reducing the comparison render target resolution as well as the introduced error.

3.4.3 Performance

The goal of our algorithm is to improve the generation of physically correct soft shadows by adaptively selecting only the light source samples which do really contribute to the visual quality of the penumbrae. The reduced number of needed shadow maps increases the overall rendering performance, but the subdivision evaluation produces an overhead of approximately 30% of the rendering time per frame. In scene configurations where the penumbra regions are comparatively small, and a significant reduction of shadow map samples is possible, even real-time performance can be achieved with our approach. Of course, whenever a penumbra is extremely large and fills a wide area of the frame buffer, and the system maximum number of samples has to be used, the method performs worse than sampling the light source with this fixed maximum number.

3.4.4 Limitations

Since the size of the penumbra regions can change drastically within a short time, the number of needed samples can vary widely as well, making our approach not suitable for applications where a guaranteed constant frame rate is necessary (like for example in real-time 3D games). We therefore see the use of this method in modeling and design scenarios (e.g., for lighting design purposes), where a fast real-time preview of a physically correct shadowing solution is necessary. In the worst case, when using this method in systems with a maximum number of usable shadow maps in combination with scenes in which large penumbras are prevalent leads to the situation that no performance gain can be achieved (see Section 3.4.3).

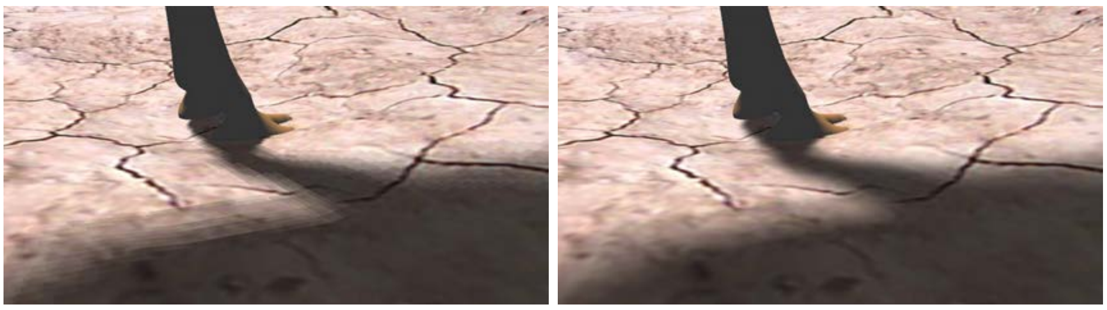

Figure 3.9: *Left:* Reducing the resolution of the comparison render target leads to the use of fewer shadow maps and therefore to banding artifacts. *Right:* By Applying a simple 3×3 PCF filter, the artifacts can be significantly reduced – but physical correctness is not guaranteed anymore.

3.5 Conclusions and Future Work

We presented an algorithm which is able to render physically accurate soft shadows that in most cases outperforms the regular light sampling method with a fixed sampling rate, since only the samples which contribute to the visual quality are computed and evaluated.

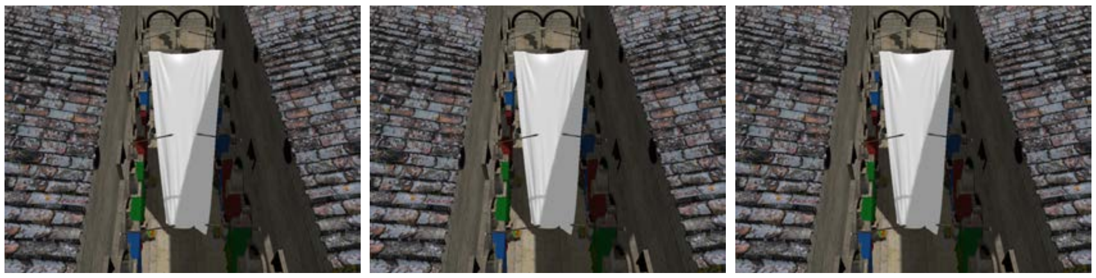

Figure 3.10: Visual Comparison using the complex *Sponza Atrium* scene: *Left:* Regular sampling with 289 shadow maps, acting as ground truth for our comparisons (2.5 FPS). *Middle:* Our method, 163 shadow maps (5 FPS). *Right:* Our method with only half the comparison render target size and a 3x3 PCF filtering requires only 14 shadow maps and is rendered at 40 FPS. See Figure 3.11 for difference images.

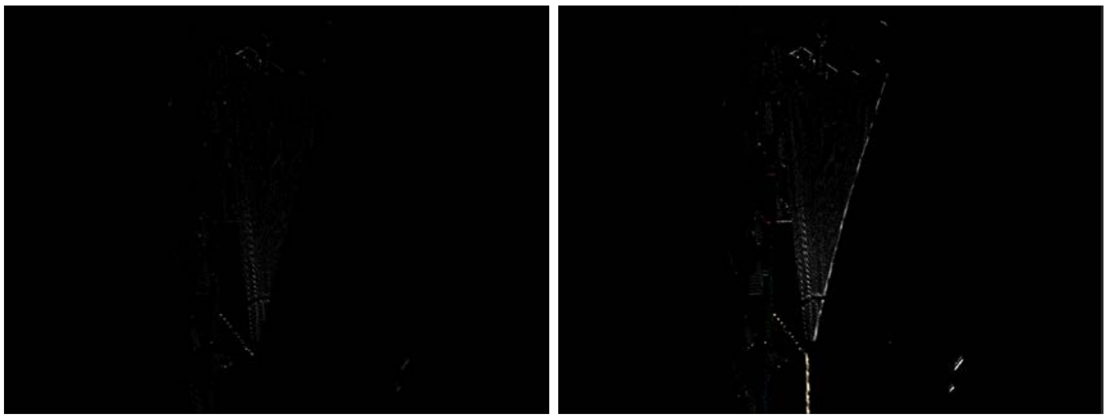

Figure 3.11: Visualized differences between the screenshots of the *Sponza Atrium* scene in Figure 3.10: *Left:* Difference image between ground truth (Figure 3.10, left) and our approach with 163 shadow maps (Figure 3.10, middle), scaled by factor 40 for visualization purposes. *Right*: Difference image between ground truth (Figure 3.10, left) and our method with reduced comparison render target size and PCF with 14 shadow maps (Figure 3.10, right), scaled by factor 40 for visualization purposes.

The decision whether another sampling point is needed in-between two neighboring ones is being reached by reprojecting the corresponding shadow maps to the camera's point of view and comparing them there using an occlusion query. The time needed for these checks is often more than compensated by the reduced number of shadow maps which have to be calculated.

In our test application, we were able to render soft shadows of a quality similar to the ones generated with 289 samples, but at interactive or even real-time frame rates. Performance can even be further increased by relaxing the subdivision criterion and using a simple PCF filter to hide potential banding artifacts.

As a future work, we want to reduce the computation time needed for the comparison step by finding better ways to handle the time-consuming occlusion queries and especially the corresponding GPU/CPU synchronization. This could for example be achieved by exploiting the temporal coherence between consecutive frames, so that the current subdivision state of the area light source is reused and only adapted when necessary in the next frame. Moreover, we plan to investigate the relation between the banding artifacts in case of a lower-resolution comparison render target and the necessary shadow filtering kernel sizes, so that self-regulating filtering mechanisms can be found. As a similar enhancement, filtering could be restricted to regions with banding artifacts only, further increasing the rendering performance. Further research effort could also be spent on finding techniques for a more randomized subdivision strategy, or on extending the algorithm to volumetric light sources.

CHAPTER 4

Reusing Soft Shadows in Consecutive Frames

This chapter is based on the publication:

Michael Schwärzler, Christian Luksch, Daniel Scherzer, and Michael Wimmer. Fast Percentage Closer Soft Shadows using Temporal Coherence. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2013 (i3D 2013)*, pages 79–86, New York, NY, USA, March 2013. ACM.

The original paper was adapted in terms of formatting and type-setting to fit this template and to increase readability. The introduction and the related work were adjusted to fit the topic of this thesis, and the abstract was removed. Minor corrections, such as fixing typos or unclear wording, were applied. The original version is available at

https://dl.acm.org/citation.cfm?id=2448209.

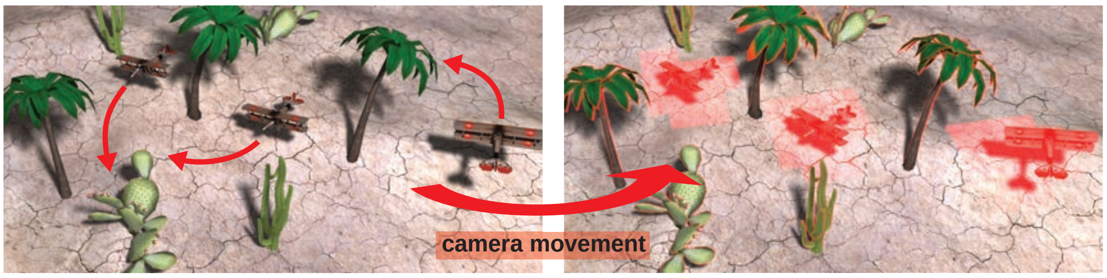

Figure 4.1: Our new method improves the rendering performance of the *Percentage Closer Soft Shadows* method by exploiting the temporal coherence between individual frames: The costly soft shadow recalculation is saved whenever possible by storing the old shadow values in a screen-space *History Buffer*. By extending the shadow map algorithm by a so-called *Movement Map*, we can not only identify regions disoccluded by camera movement, but also robustly detect and update shadows cast by moving objects: Only the shadows in the areas marked red in the right image have to be re-evaluated. This saves rendering time and doubles the soft shadow rendering performance in real-time 3D scenes with both static and dynamic objects.

4.1 Introduction

In the previous chapter, we proposed a way to reduce the computational effort that is needed for rendering physically correct soft shadows from an area light source by reducing the number of needed samples. Even though this approach makes the use of soft shadows at interactive frame rates possible, a general application in complex 3D modeling scenarios (e.g., with multiple light sources) is limited due to performance issues. We therefore investigate the idea to *reuse soft shadow information* whenever possible, i.e., we want to avoid a costly soft shadow recalculation in each frame for each pixel, reducing computation time and increasing the rendering and interaction performance.

Various publications tackled the problem of reducing the computational effort for expensive operations in the pixel shader by introducing a so-called *history buffer* to exploit temporal coherence [NSL$^+$07, SJW07, SaLY$^+$08a, SaLY$^+$08b]. The main idea is to reuse the calculated pixel values over several consecutive frames by storing them in a screen-space buffer and reprojecting them into the next frame. If the reprojected value is still valid (i.e., the fragment depths between the two frames is below a given threshold), it can be used to either omit a costly re-calculation or to improve the image quality.

Another way to speed up soft shadow calculation at the cost of physical accuracy is to use single-sample approaches. They try to approximate a computationally expensive, exact solution by calculating a hard shadow first, and apply a blur filter kernel that is based on an approximation of the occluders between the light source and the shadowed surface. While methods like *Percentage Closer Soft Shadows (PCSS)* [Fer05] or *Back-projection* techniques [GBP06, GBP07, AHL$^+$06, ASK06, SS07, BFGL09] achieve physical plausibility when putting high computational effort into the costly occluder analysis, they are still prohibitively expensive to use in real-time 3D applications.

In this work, we propose to combine the idea of reusing data from previously rendered frames with the expensive calculation of perceptually convincing soft shadows with varying penumbra size (see Figure 4.1): Soft shadow intensities calculated with the PCSS algorithm are stored in a history buffer and potentially reused in consecutive frames. In case of shadows generated by area light sources, this task cannot be fulfilled with a simple per-fragment depth comparison, as moving objects cast complex shadows on completely different regions in the scene. We therefore propose a simple and easy-to-implement enhancement to the shadow map generation step, allowing shadows that have become invalid to be detected with a single texture lookup. We further discuss the issues introduced during the buffer reprojection, the limitations concerning moving light sources, and give details on our implementation and the achieved results.

4.2 Related Work

We give a short overview on publications related to our work. We refer the interested reader to the book recently published by Eisemann et al. [ESAW11] and to [HLHS03] for an in-depth overview on real-time (soft) shadows, and to a state-of-the-art report on temporal coherence techniques in real-time applications by Scherzer et al. [SYM$^+$11].

4.2.1 Real-time Soft Shadow Mapping

Although the *Shadow Mapping* algorithm proposed by Reeves et al. [RSC87] (explained in Section 3.3.1) is fast and easy to use, this algorithm suffers from aliasing and undersampling artifacts. Shadow filtering methods like *Percentage Closer Filtering (PCF)*, where the distance is also compared to the neighboring values in the shadow map to generate a “shadow percentage”, or pre-filtering techniques like *Variance Shadow Maps* [DL06], *Convolution Shadow Maps* [AMB$^+$07] or *Exponential Shadow Maps* [AMS$^+$08] reduce this artifacts by blurring the shadow edges. Apart from reducing the artifacts, the introduced “softness” of the shadow is also perceptually more convincing than hard shadow borders, since nearly all light sources in reality have a certain extent. Still, these blurry shadows do not reflect the fact that the size of the penumbra (the “half-shadowed” area, from which the light source is partly visible) varies depending on the distance relations between the light source, occluders and receiver.

In order to simulate more accurate soft shadows with varying penumbra sizes, *Back-projection* techniques ([GBP06, GBP07, AHL$^+$06, ASK06, SS07, BFGL09]) use a single shadow map not only for depth comparison, but employ it as a discretized representation of the scene. The visibility factor for a screen-space pixel is calculated by back-projecting the shadow map texels onto the light source, where the amount of occlusion is estimated. These approaches produce perceptually convincing results in many cases, but are prone to artifacts and require a costly blocker search in the shadow map.

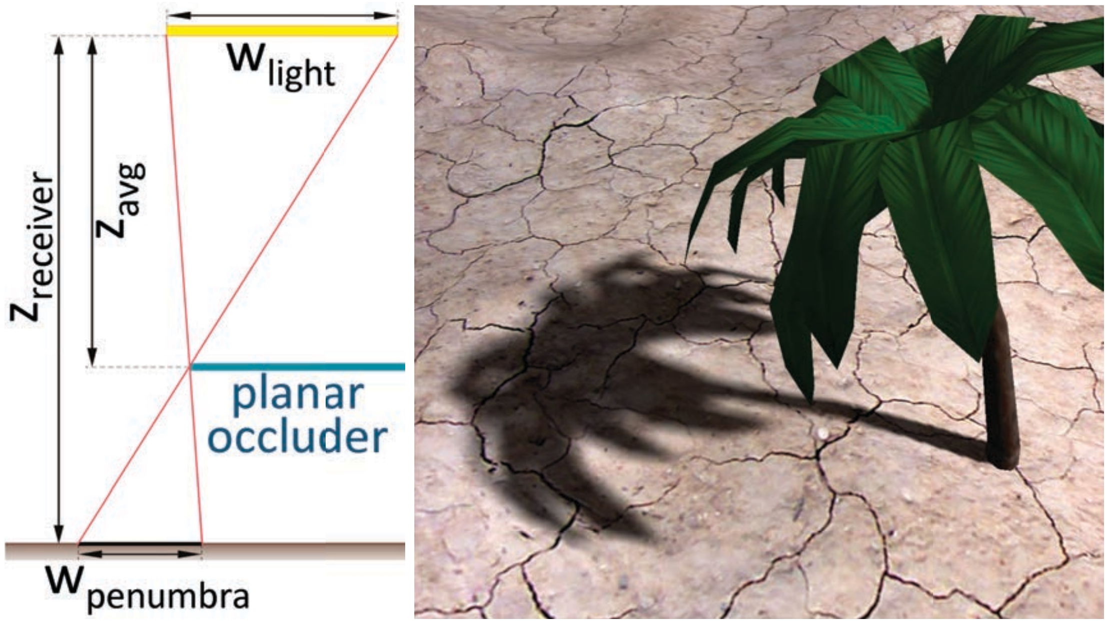

Figure 4.2: The PCSS algorithm: By estimating a penumbra width based on light size, average occluder distance and receiver distance (see Eq. 4.1), the filter kernel size is adapted, generating visually plausible varying penumbra sizes.

Percentage Closer Soft Shadows (PCSS) [Fer05] extend the *PCF* filtering method to support variable kernel sizes in order to simulate varying penumbra sizes (see Figure 4.2, right): An average blocker distance z_{avg} is first calculated by searching for values in the shadow map that are smaller than current pixel's depth within an initial kernel. Under the assumption that blockers and receiver are planar and in parallel, the penumbra size $w_{penumbra}$ is estimated based on similar triangles (see Figure 4.2, left) using the relations between pixel depth $z_{receiver}$ and light source size w_{light}, and the filter kernel is adjusted accordingly:

$$w_{penumbra} = w_{light} \frac{(z_{receiver} - z_{avg})}{z_{avg}}. \tag{4.1}$$

While the approach is comparably easy to implement and produces visually pleasing soft shadows, the vast amount of texture lookups (blocker search + large filter kernel) have a negative impact on rendering performance. We therefore propose a way to overcome this expensive calculation by exploiting temporal coherence techniques (see Section 4.2.2). Based on this idea of an additional blocker search for the estimation of the penumbra filter kernel size, adoptions of the prefiltering techniques mentioned above have been proposed to follow the PCSS pipeline [YDF$^+$10, ADM$^+$08]. While the achievable performance of prefiltering techniques is superior to a simple PCSS version, the implementation and handling is rather complex.

4.2.2 Data Caching / Temporal Coherence

Reusing data from previous frames by reprojecting it into the current frame has been independently proposed by Nehab et al. [NSL$^+$07] and Scherzer et al. [SJW07], and is referred to as *Reverse Reprojection.* The idea is to store per pixel information in an off-screen buffer - the so-called *history buffer*, *payload buffer* or simply *cache.* This buffer is viewport-sized and filled for all visible rasterized surface points. In consecutive frames, the cached data is reprojected according to the scene motion, and reused in the current frame whenever possible: By comparing the stored depth against the current depth (within a given tolerance), it is decided whether the pixel was visible in the previous frame, and the data stored in its buffer location can usually be safely reused (except for changes in the shading signals, e.g., a moving light source, specular highlights, etc.). In disoccluded regions or areas that lay outside the view frustum in the previous frame, the information has to be recalculated. It has to be pointed out that due to the lookup in the history buffer and the corresponding resampling, a reprojection error is introduced whenever the viewpoint changes, see Section 4.3.3.

Depending on the application, the reprojected data can be used to improve both image quality [SJW07, SW08, YNS$^+$09] and performance (by either reducing the need for expensive recalculations [NSL$^+$07, SaLY$^+$08a, SaLY$^+$08b], or by distributing expensive calculations into multiple frames [SSMW09]). Reiner et al. [RLD$^+$12] use a similar cache structure to increase the rendering performance in an interactive procedural modeling system.

4.2.3 Temporal Coherence and Shadows

Temporal Coherence in shadow calculation has been scientifically discussed twice: Scherzer et al. [SJW07] exploit temporal coherence to converge to pixel-correct hard shadows. In Scherzer et al. [SSMW09], the costly calculation of physically correct soft shadows is spread over multiple frames, so that the accumulated shadow values in the history buffer converge to the exact solution. In contrast to these publications, we concentrate on increasing rendering performance for plausible soft shadows, and suggest a stable solution for dynamic scene objects - an unsolved issue for both proposed techniques.

4.3 The Algorithm

We describe how to combine the idea of exploiting *Temporal Coherence* with the generation of soft shadows using the *PCSS* method (Section 4.3.1). Especially the robust updates of soft shadows cast by moving scene objects (Section 4.3.2) and the handling of the reconstruction error introduced during the reprojection (Section 4.3.3) are non-trivial issues and require special considerations.

4.3.1 Shadow Reprojection

We closely follow the *Reverse Reprojection* pipeline in order to reuse soft shadow information (see Figure 4.3): Each fragment's shadow value, computed via PCSS (implemented exactly as in the NVIDIA PCSS white paper [KFB08]), is not only used to illuminate the corresponding scene surface, but also stored together with the clip space scene depth in a 2-channel off-screen viewport-sized history buffer (which is set as a second render target). If the depth test as described in Section 4.2.2 passes, the old shadow value is reused. Since reading the old buffer information and writing the new data in the same rendering pass is not allowed on current GPUs, we use two buffer textures and switch them in ping-pong style. Assuming a static scene configuration, the costly PCSS evaluation has to be only performed in regions of disocclusions or in areas which have previously been outside the screen borders after camera movements, significantly reducing the rendering load: Instead of a minimum of 16 texture lookups for the blocker search and the filtering step each, only a single bilinear lookup in the history buffer has to be executed.

4.3.2 Detecting Moving Objects

Whenever an object in the scene moves, the shadows cast *on* this object as well as the shadow cast *by* it change, so that the corresponding fragments in screen space cannot be reconstructed from the history buffer and need to be recalculated. The invalidation of shadow information that is cast *on* a dynamic object is trivial, as these fragments automatically fail the depth test described in Section 4.3.1. For the shadows cast by dynamic objects onto static objects anywhere in the scene, this depth test is completely irrelevant (as of course no change in depth is induced by shadows), making the invalidation a much more complex task, especially in the case of soft shadows (see Figure 4.4).

For this reason, we extend the shadow mapping algorithm by storing not only the depth from the view of the light source, but also the information whether an object is currently moving, in a light-weight binary mask buffer with the same size as the shadow map. This buffer is set as a second render target, and can therefore be written to in parallel to the depth map. We use an 8-bit buffer and output a value of "1" (which is stored as "255") in the shader for a texel where a moving object is visible, and "0" otherwise, and refer to this buffer by calling it *Movement Map* (see Figure 4.5). By looking up this information during the shadowing pass with a preliminary texture fetch, it can immediately be decided whether a shadow recalculation is necessary for the fragment, or if the history buffer should be investigated. Additionally, shadow values in whose calculation moving objects have been involved are marked as such in the history buffer, so that they get updated when the dynamic object casting the shadow has "moved on", avoiding the shadow leaving a "trail".

A problem with this approach lies in the concept of using only a single hard shadow map for the generation of physically plausible soft shadows through filtering. The information regarding moving objects is only valid for the original hard shadow, and needs to be extended to reflect the penumbra region, i.e., the region that is partly visible from the

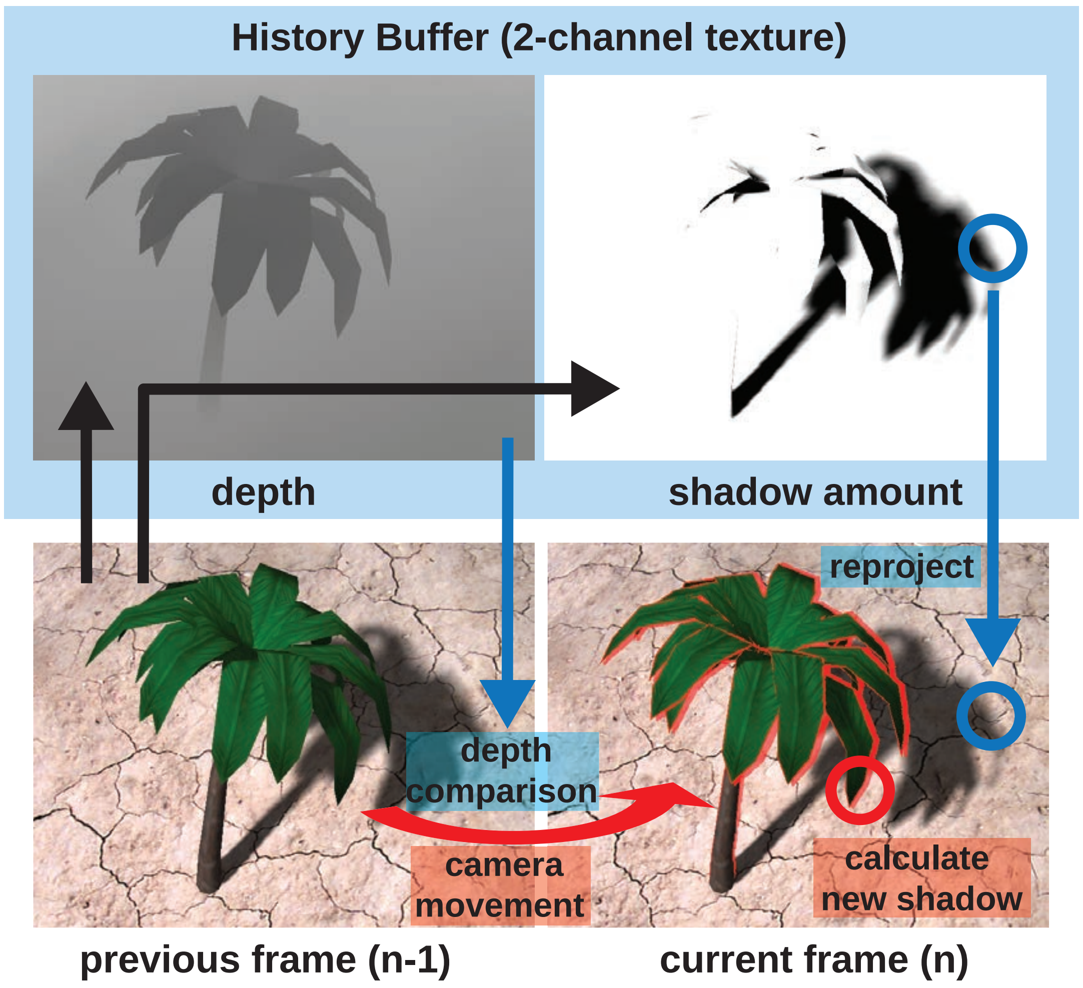

Figure 4.3: Shadow reprojection: Depth and soft shadow amount from frame $n - 1$ are stored in the *History Buffer*. In frame n, the buffer is reprojected, and the depth values are compared. If an occlusion (marked red) is detected, a new soft shadow value has to be estimated using the *PCSS* algorithm; otherwise, the stored shadow can be re-used.

area light source. We divide the penumbra itself into two regions that require special attention: the *inner penumbra*, representing the part that lies inside the hard shadow borders and is connected to the fully shadowed *umbra* region, and the *outer penumbra*, extending the hard shadow and fading out until the surface is fully lit. The problems associated with these regions are (see also Figure 4.6 and Figure 4.7):

1. *Inner penumbra of static objects*: The inner penumbra of a static object lying closer to the light source than a moving object on the same "light ray" will block an update of the area the moving object cast a shadow on, as the value in the

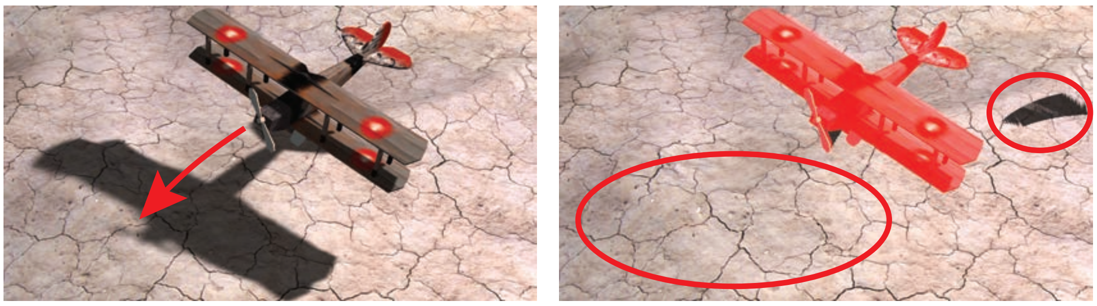

Figure 4.4: Shadows from a moving object. *Left:* Correctly shadowed scene of a moving toy airplane using PCSS. *Right:* Naive reprojection using depth comparison only recognizes that the shadow *on* the airplane needs to be updated, but does not indicate the need for a shadow recalculation in the regions marked with red ellipses.

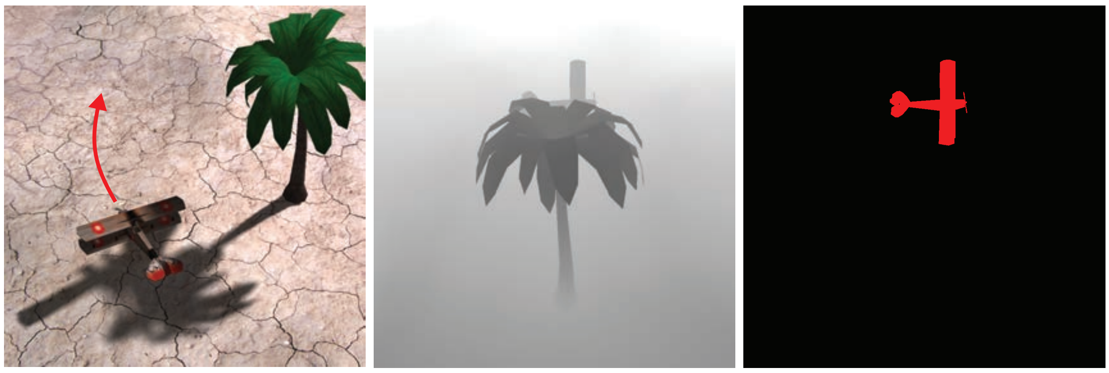

Figure 4.5: Shadow mapping extension for dynamic objects. *Left:* A scene with a static palm, casting a shadow onto the moving toy airplane. *Middle:* The depth map as known from the shadow mapping algorithm. *Right*: The corresponding *Movement Map*, indicating regions in which a moving object casts a shadow. Note that by rendering dynamic objects first, the parts of the airplane that are *occluded by the palm* can be stored as well. After the moving objects are stored, mipmaps are generated and used for an efficient lookup during the shadowing pass.

movement map is “0” (see blue ellipse in Figures 4.6 and 4.8).

2. *Outer penumbra of moving objects*: Whenever an object moves, the approaching soft shadow is *larger* than the “tagged” area in the movement map described above (see red ellipse in Figures 4.7 and 4.8). It would therefore be necessary to search for information in the map within a given radius through costly texture lookups, annihilating the algorithmic speedup gained by using the history buffer as described in Section 4.3.1.

We solve these two difficulties by refining our strategy on how the movement map is

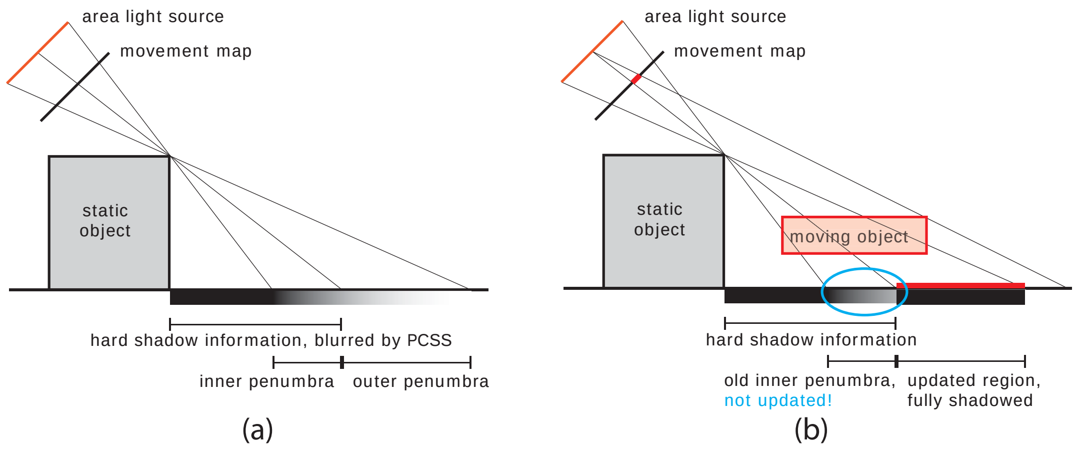

Figure 4.6: Problems with inner penumbras of static objects: In a scene with static objects only, the movement map stays empty (a). If a moving object enters the view frustum of the light source (b), and static objects are rendered prior to moving objects, the data in the movement map is not properly set as seen in Figure 4.5 due to z-buffering! This prevents a proper update and causes artifacts in inner penumbra regions of static objects (blue ellipse). See Figure 4.8 for a visualization of the artifacts caused by such special scene configurations as well as the results in Figure 4.11 for an example on how to correctly handle these cases.

filled and used: By first rendering all moving objects into both the depth map and the movement map, then releasing the movement map as a render target, and finally rendering the remaining objects, the fragments of the misleading static elements (problem *I*) are not depicted in the movement map anymore, ensuring that all inner penumbra regions are updated correctly. At the same time, the depth map itself represents the scene with the correct ordering of objects and depth values as usual.

In order to solve problem *II* with the outer penumbra of moving objects, we exploit the fact that (in contrast to the depth map) the movement map can be prefiltered, and use the hardware-accelerated mipmap generation procedure to efficiently create an image pyramid of the movement map. Since we use an 8-bit buffer, a value of “255” leaves a footprint over at least 5 mip levels using the pixel-averaging mipmap generation algorithm, which proved to be sufficient in all our tests (in case of a 1024^2 depth map, level 5 represents a search radius of 32 texels). This allows us to search for a moving object within a given radius by simply checking if the value of a texel in the desired mip layer is *larger than zero* with only a single texture lookup. The selection of the correct mip level is equivalent to the calculation of the initial occluder search radius in the PCSS algorithm (see Equation 4.2), and is estimated by taking the area light source size w_{light}

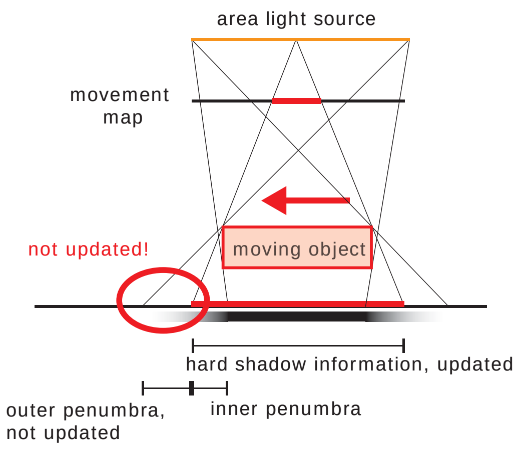

Figure 4.7: Problems with outer penumbras of moving objects: If only a single lookup in the highest mip level (i.e., the “hard shadow” boundaries) is used to evaluate shadow updates, the outer penumbra regions of moving objects are not properly updated (red ellipse).

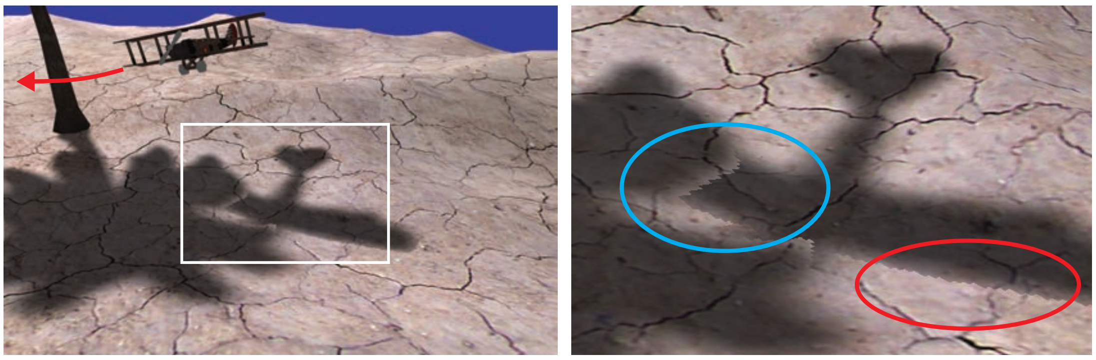

Figure 4.8: Artifacts in the penumbra regions (as explained in Figure 4.6 and Figure 4.7) that are removed by our movement map generation strategy (see the results in Figure 4.11 for a proper handling of these cases).

(in UV space) and the distance of the fragment $z_{receiver}$ to it into account [KFB08]:

$$r_{search} = \frac{w_{light} * (z_{receiver} - d_{Nearplane})}{z_{receiver}} \tag{4.2}$$

Note that the size of the light source in UV space w_{light} is a customizable parameter and has to be set by the programmer according to scene scale and is “something you can change with artistic preference” [KFB08]. The corresponding mip level of a shadow map with size w_{SM} can thus be found by evaluating

$$l_{mip} = \lceil \log_2 (2 * r_{search} * w_{SM}) \rceil, \tag{4.3}$$

where w_{SM} is the shadow-map resolution. With the movement map prepared this way, we can efficiently and robustly detect *soft* shadows of dynamic objects for any fragment in screen space, allowing us to quickly decide whether a new shadow value has to be calculated due to object movement, or if the history buffer should be checked for reusable data.

4.3.3 Reconstruction Error

Reprojecting the history buffer from the previous frame to the current one always comes at the cost of a certain reconstruction error in case of camera movement. This problem is equivalent to transforming a 2D image, and requires resampling of discrete data. Since the best available native reconstruction filter on today's graphics hardware is bilinear interpolation, state-of-the-art publications in the area of temporal coherence [SYM$^+$11] advise to sample the history buffer accordingly, as most of them reuse the stored data only for one or a few frames.

While this strategy may be sufficient in many use cases, we have made the observation that keeping and reprojecting shadow values for several hundreds or thousands of frames using bilinear interpolation introduces an amount of additional softness in the shadow that is noticeable in certain scene configurations. Interestingly, this yields both positive and negative aspects for the soft shadows (see Figure 4.9):

- The additional blur leads to a *reduction of banding artifacts* that can occur when the penumbra size is large and not enough samples are used during the PCF step. By applying our proposed reprojection strategy, these artifacts disappear automatically after a few frames.
- Unfortunately, the blurriness falsifies the penumbra size drastically in *contact areas*, where shadow casters and shadow receivers touch each other. In such regions, the shadow is nearly a hard shadow (i.e., there is hardly any penumbra visible), and the introduced softening is therefore visually disturbing.

Depending on the scene configuration it can therefore be necessary to make sure the accumulated reprojection error does not become too large in order to avoid "oversmoothing". We have evaluated two strategies to overcome the problem (see Section 4.5 for details):

Using third-order (bicubic) texture sampling by manually implementing it as a shader function, the amount of blur introduced during the reprojection is significantly smaller, retaining small penumbras. Note that the applied bicubic filter has to reconstruct the history buffer by *interpolation*, and not by *approximation*, as otherwise the shadow information "loses energy" and becomes unusable within a few frames. Unfortunately,

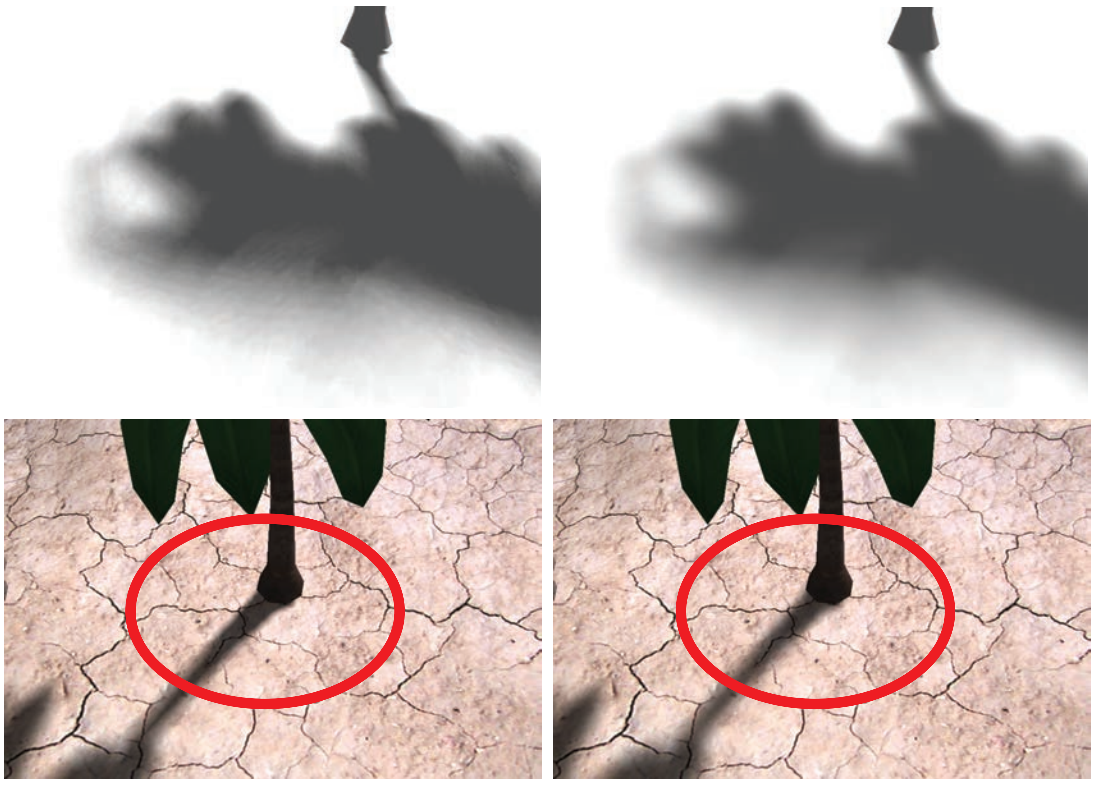

Figure 4.9: The blur introduced by bilinear reconstruction has both positive and negative effects. *Top Row:* Typical PCSS band artifacts (left) disappear after the camera has been moved and the shadow has been reprojected several frames (right). *Bottom Row:* Shadows in "contact regions" with a small penumbra (left) become too soft due to reprojection (right).

this makes the use of an optimized B-Splines filter with only 4 bilinear texture fetches as proposed by Sigg and Hadwiger [SH05] impossible. Instead, we implemented a Catmull-Rom interpolation with 16 nearest neighbor texture fetches for the history buffer lookup, which is used in frames whenever the camera has moved. While this conceptually solves the problems with the reprojection error, the application is only feasible if a high-quality PCSS version with more than 16 texture lookups is used (e.g., 64 for the PCF step) – or if future generations of GPUs support a corresponding bicubic texture sampling in hardware.

Alternatively, a refresh strategy as described in [SYM$^+$11] can be used to update the bilinearly reprojected texels before the accumulated error becomes noticeable. By dividing the screen into groups in a grid, tiled regions can be updated periodically using a global clock. An *amortized sampling* strategy ensures that newly calculated shadow values do not completely replace the old ones, but that they are gradually blended, avoiding visible transitions caused by the update pattern.

Although refreshing the fragment pixels reduces the achievable performance speed-up factor by approximately 10%, we opted for this solution in most of our application scenarios, as it is highly configurable, simple to implement, predictable and stable. Bicubic texture sampling has a larger performance impact and reduces the achievable speedup by about 20% on today's graphics hardware. Still, the decision on whether the reprojection error needs to be minimized and what strategy is applicable depends on the scene configuration, the area light source size, the amount of movement and the desired shadow quality.

4.4 Implementation

Our proposed technique to increase the rendering performance of the PCSS algorithm can be implemented on all current shader-based rendering frameworks. We have implemented the algorithm in a C++ framework using DirectX10 for testing and evaluation purposes. Based on the PCSS example provided in the NVidia white paper [KFB08], we render the depth map into a 32-bit render target with a size of 1024^2, and simultaneously use an 8-bit texture render target of the same size as the movement map. Note that *conceptually*, it would be sufficient to simply use a DirectX Depth-Stencil Resource to save both the depth and the movement information, but due to API limitations, we have to use two separate render targets: The first issue is that a depth resource can only store depth values between 0 and 1, but the original PCSS algorithm uses linear depth values. Secondly, no mipmaps can be generated for a DirectX stencil buffer resource.

For the history buffer, we use two screen-size texture resources with two 32-bit channels each, where one texture is set as a render target and stores the current information, and the other acts as the lookup buffer for the information from the previous frame. After each rendered frame, the two textures are swapped ("ping-pong"). While in the first channel of the history buffer the current shadow value is saved, the second channel stores the depth and (encoded by a negative sign) the information whether the shadow originates from a moving object. The usage of such screen size buffers is related to *deferred shading* approaches that have become a popular method in today's 3D games, and does therefore well integrate in such rendering systems.

4.5 Evaluation and Comparison

We have tested our algorithm in three different scenarios in terms of necessary shadow updates and visual quality (see Figures 4.10 and 4.11): First, we evaluated our method in a completely static scene, then replaced some objects by moving toy airplanes, and finally tried to challenge our algorithm with a scene where all shadow casters are moving. In these three scenes, we used exactly the same camera path for the scene walkthrough. The system used for the tests consists of an Intel Core i7 920 CPU with 6GB of RAM and a Geforce GTX 580 GPU.

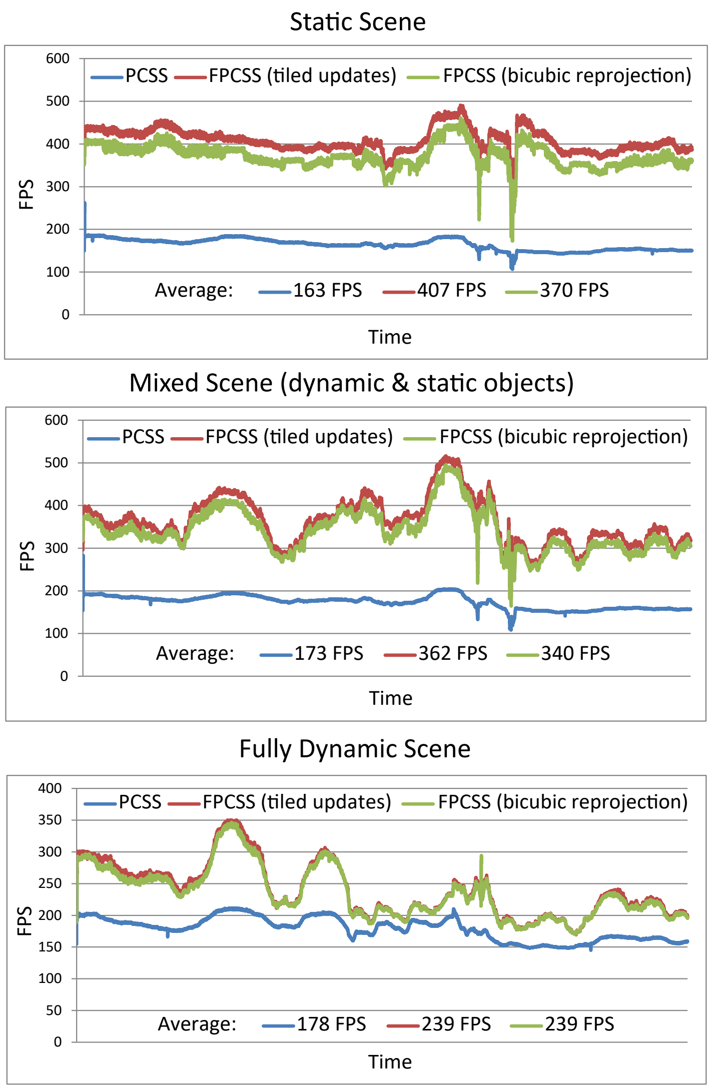

Figure 4.10: Benchmark comparison of different scenes. The standard PCSS algorithm, our new method with a tiled update strategy, and our new method with bicubic reprojection have been tested (FullHD resolution, 64 samples for both the blocker search and in the filtering step). In all three scenes, the same camera path has been used for the walkthrough. In the first frame of the recording, the history buffers have been filled with data from preceding frames. *Top:* In a static scene, where only the camera was moving, an average speedup factor of 2.5 could be achieved. *Middle:* In a scene with both static and moving objects (representing a typical game scene), the average performance was increased by factor 2. *Bottom:* Even in a scene with only dynamic shadow casters, the scene could be rendered at 130% of the speed of the PCSS version.

We compared three different algorithms per scene in our benchmarks: The standard PCSS algorithm, our method with a tiled region update strategy, and our method with bicubic reprojection (see Section 4.3.3). In the PCSS step, we used 64 samples for both the blocker search and the filtering step, allowing us to simulate a large area light source with visually plausible penumbras. The screen view port was set to a resolution of 1920x1080 (FullHD) for the benchmarks.

As can be seen in Figure 4.10, our proposed algorithm outperforms the standard PCSS algorithm in all three scenarios. It is easily comprehensible that the greatest performance improvement (250% of the PCSS frame rate) can be achieved whenever most of the shadow values in the scene can be reused (i.e., the scene is static). Still, a significant performance boost (130%) can even be gained in a fully dynamic scene where all shadows cast by the moving objects have to be recalculated! This effect can be explained by the fact that shadowless regions (in contrast to the standard PCSS algorithm) do not need to perform the costly blocker search at all, but can rely on the information in the movement map that is fetched with a single texture lookup. In a scene with both static and dynamic objects, comparable to situations often found in 3D games, the average frame rate is doubled.

In general, our proposed method benefits from the computational complexity inherent in the chosen soft shadow algorithm: the more expensive shader instructions can be saved through the reprojection, the higher is the speed-up. The relative performance boost would therefore be even higher in a PCSS version with 128 texture lookups for the blocker search and 128 lookups for the PCF filtering step, but lower when using a version with only 32 lookups each. Note that the chosen PCSS light source size itself cannot be seen as a direct influence factor for the prospective performance, as the evaluation of pixels to recalculate takes place in screen-space only: The camera position and the scene configuration have a significantly larger impact (e.g., if the camera is very close to a penumbra region of a dynamic object, nearly the whole screen has to be updated in the next frame – even if the light source itself is comparatively small).

As demonstrated in the close-up images of Figure 4.11, the perceivable differences between the different algorithms are negligible and hardly noticeable. Even if shadows in contact areas become softer than the PCSS version due to bilinear reprojection, the tiled region update strategy combined with amortized sampling quickly covers up the introduced blur within a few frames.

4.6 Discussion and Conclusion

We have presented a new method to *accelerate the computationally expensive PCSS algorithm* by exploiting *temporal coherence* techniques and by extending the shadow-mapping algorithm by a so-called *movement map* – a light-weight 8-bit buffer storing the location of moving objects in light space. By pre-filtering this map through mipmap generation, it can be easily decided with a single texture lookup whether the soft shadow value stored in a screen-space history buffer can be reused or has to be re-estimated.

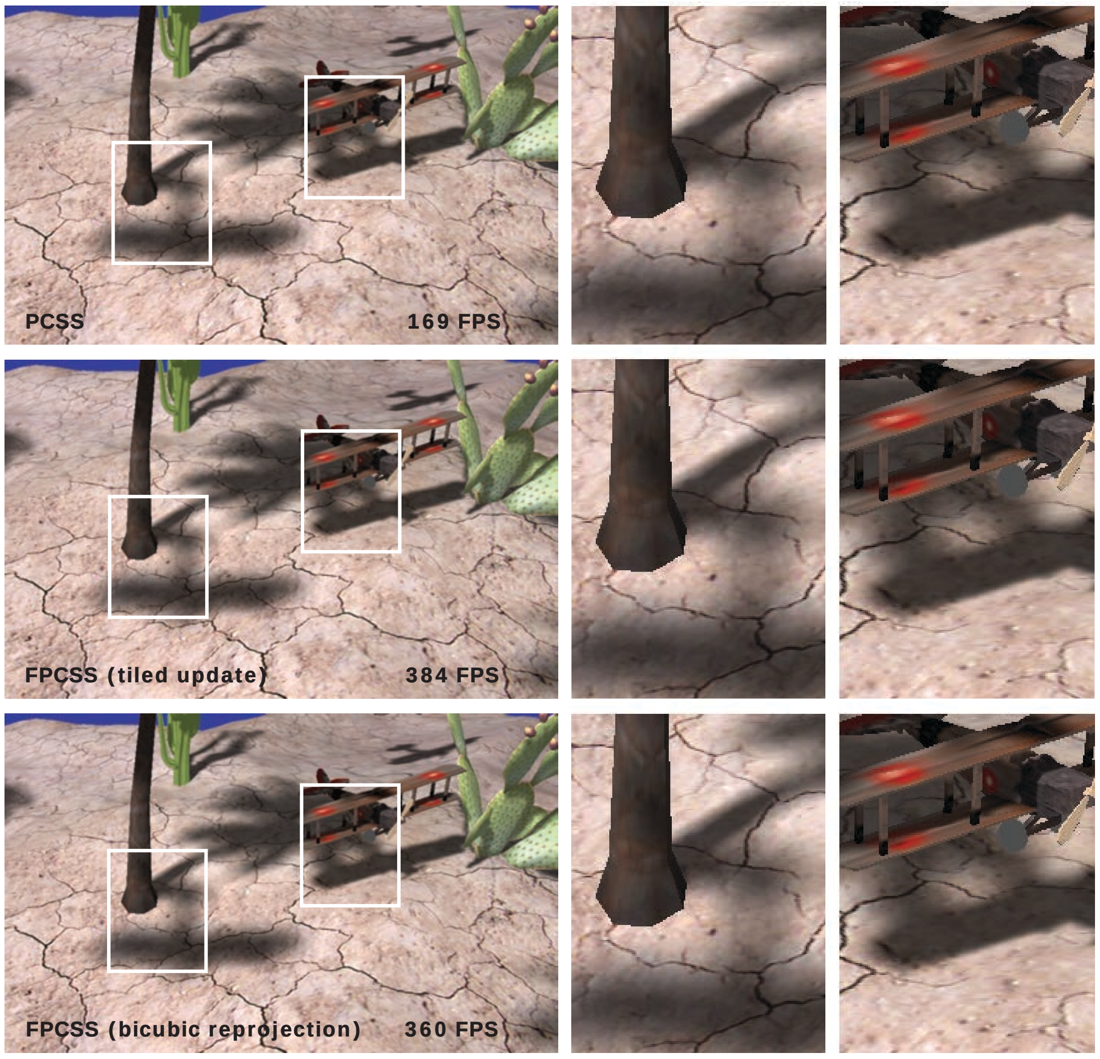

Figure 4.11: *Left Column:* Result screenshots from the benchmark scene with static and dynamic objects. *Top:* PCSS, 169 FPS. *Middle:* Fast PCSS with tiled updates and amortized sampling strategy, 384 FPS. *Bottom:* Fast PCSS with bicubic reprojection of the history buffer, 360 FPS. *Middle and Right Column:* Close-up views of critical areas (contact shadows, overlapping penumbras) show that the achievable visual quality of our new approach is nearly equal to the quality of the significantly slower PCSS version.

The algorithm is easy to integrate into an existing rendering framework, and can be robustly used for all kinds of different scenes. The achievable performance gain (between 250% in static scenes and 130% in fully dynamic scenes) comes at the cost of *memory consumption*, though: Apart from the 8-bit buffer for the movement map, two more 32-bit 2-channel screen-size buffers have to be allocated for the history buffer.

A limitation of our proposed algorithm is its application in scenes with *moving light sources*: While our algorithm can robustly handle such cases by setting all values in the movement map (i.e., in the “view frustum”) of the light source to “255”, obviously no speedup can be achieved. In this worst case, our algorithm is minimally slower (2-5%) than the standard PCSS algorithm, as it has to perform the mipmap generation and the additional movement map lookup. In scenarios with both static and moving light sources (e.g., with a static sun light source from above, and a headlight on a car driving around in the scene), using our method can still improve rendering performance: as long as the moving light source does not force an update of all fragments in screen space, at least the soft shadow from the static light source can be efficiently reused.

It has furthermore to be pointed out that the due to the varying number of pixels that need to be updated, the frame rate of our approach is not as constant as when using the original PCSS method (see Figure4.10). This may be of concern whenever the overall rendering performance of the application is close to a critical threshold, e.g., 30 FPS.

We hope to be able to extend our idea to further soft shadow algorithms in order to make their use feasible in 3D games and applications – especially the real-time calculation of physically correct soft shadows in dynamic scenes is still a challenge yet to be solved.

CHAPTER 5

Interactive Polygon Snapping for 3D Building Reconstruction

This chapter is based on the publication:

> Murat Arikan, Michael Schwärzler, Simon Flöry, Michael Wimmer, and Stefan Maierhofer. O-Snap: Optimization-Based Snapping for Modeling Architecture. *ACM Transactions on Graphics*, 32:6:1–6:15, January 2013.

The original paper was adapted in terms of formatting and type-setting to fit this template and to increase readability. The introduction was adjusted to fit the topic of this thesis, and the abstract was removed. Minor corrections, such as fixing typos or unclear wording, were applied. The original version is available at

> https://dl.acm.org/citation.cfm?id=2421642.

Please note that this publication has a comparatively large scope and extent, and the work for the main contributions has therefore been split between the first and second author. While Murat Arikan focused on the development of the optimization-based snapping process (Sections 5.4 and 5.5), the application of these methods as a novel, guided 2D tool in a 3D modeling context (Section 5.6) was for the most parts proposed, realized and evaluated by the author of this thesis, and has to be seen as the contribution for this work.

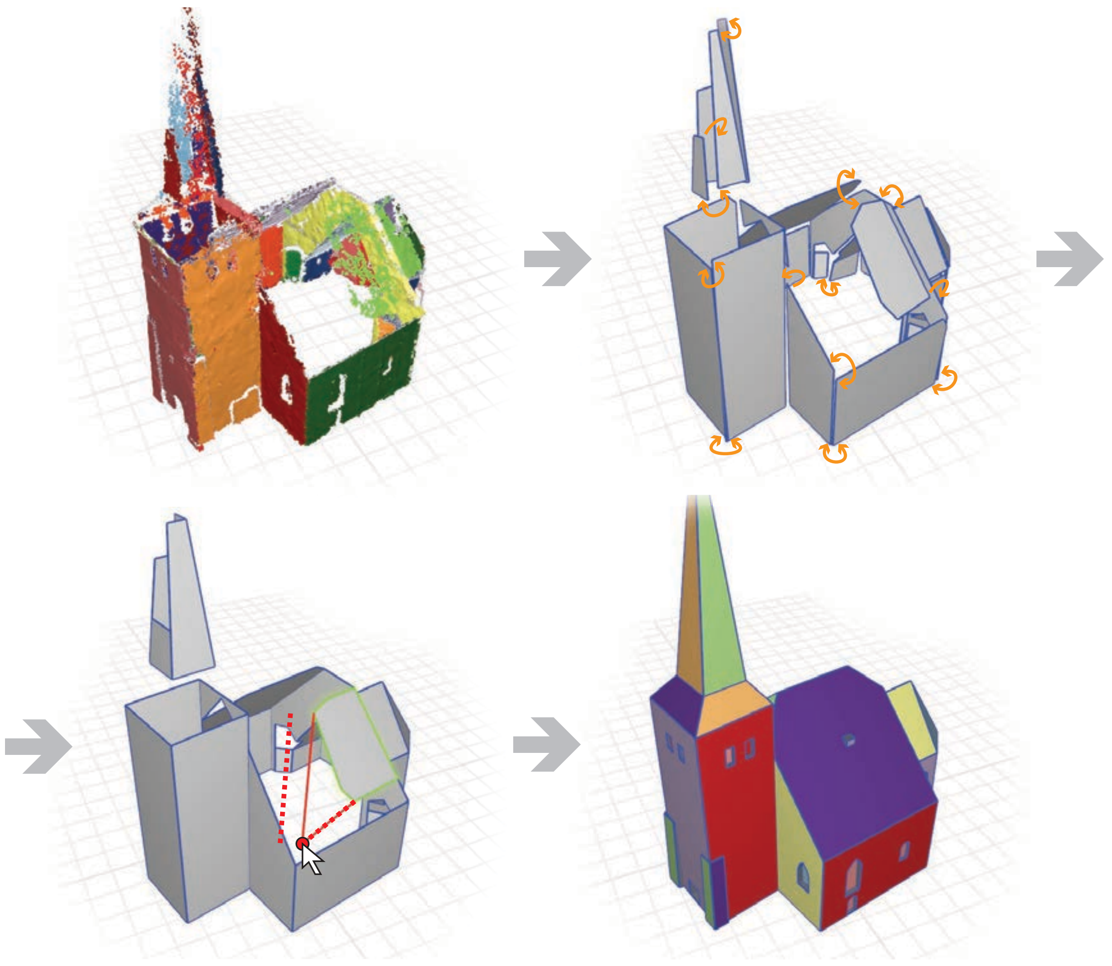

Figure 5.1: An overview of our reconstruction and modeling pipeline: Starting from a noisy and incomplete point cloud, we decompose the input data into subsets lying approximately on the same plane (top left). The boundary points of each subset are extracted and used to estimate coarse polygons (top right). Local adjacency relations are automatically discovered (top right) and enforced via a non-linear optimization to snap polygons together, providing an initial reconstruction (bottom left) that can then be interactively refined by our optimization-aided sketch-based interface within a few clicks (bottom left), yielding a coarse polygonal model that well approximates the input point cloud (bottom right).

5.1 Introduction

By putting the focus on geometric reconstruction and modeling of low-polygonal 3D buildings based on point clouds, this chapter presents another aspect relevant for the field of *Multimodal Urban Reconstruction and Modeling*. Converting raw point cloud data into 3D models suitable for applications like games, GIS systems, simulations, and virtual environments has so far been a complex and time-consuming task suitable for skilled 3D artists only. Even though modeling applications like Google SketchUp and its Pointools plugin [Poi11] address this issue by proposing simplified user interfaces and interoperability with other products like Street View (from where the artist can for example retrieve photogrammetric data), the modeling process remains cumbersome and time-consuming, as the accuracy of the reconstruction depends on the skills and patience of the user. We propose a system for modeling 3D buildings (from measured point cloud data) that analyzes and exploits the inherent structure of buildings to create a low-polygonal representation, simplifies user interactions during the modeling process, accounts for missing data, and always maintains the fitting to the input data.

Our approach is mainly based on the observation that many man-made objects, and especially buildings, can be approximated and modeled using planar surfaces with piecewise linear outlines as their primary elements, as long as the desired level of detail is not too high. There are two main insights that drove this research: (1) in order to create 3D models suitable for virtual environments, the system needs to propose an initial solution that already abstracts from the deficiencies of the input data, like noise, missing elements etc. And (2), modeling requires a tight coupling of interactive input and automatic optimization. We therefore propose a modeling pipeline that first creates a coarse polygonal model from an input point cloud, based on the decomposition of the points into subsets associated with fitted planes and subsequent polygon boundary extraction. The most important step is an optimization-based algorithm that snaps adjacent parts of the model together. This algorithm is then repeatedly carried out during the interactive modeling phase to facilitate modeling. Our work exploits certain characteristics in the input data to provide an optimized reconstruction of piecewise planar surfaces with arbitrary topologies, which is precise on the one hand, and comprises a very low number of faces on the other hand.

Our main contributions are

- a new polygonalization pipeline for point clouds that abstracts polygon outlines to reasonable shapes even in the presence of high amounts of noise and outliers, that is easy to implement and most importantly maintains interactivity during the modeling process,

- a new optimization-based snapping algorithm for polygon soups, where the novelty lies in a robust discovery of adjacency relationships,

- a new interactive modeling paradigm based on 2D sketching combined with interactive optimization-based snapping, allowing the user to model with coarse strokes.

5.2 Related Work

The challenge of quickly generating 3D models of architectural buildings from images, videos or sparse point clouds has received tremendous interest lately. Although significant success has been achieved with both semi- and fully automatic systems such as from Werner and Zisserman [WZ02], Schindler and Bauer [SB03], Chen and Chen [CC08], Furukawa et al. [FCSS09] and Vanegas et al. [VAB10], as well as interactive systems such as from Debevec et al. [DTM96], van den Hengel et al. [vdHDT$^+$07], Sinha et al. [SSS$^+$08] and Nan et al. [NSZ$^+$10], these systems either require a greater amount of manual intervention or have strict assumptions on the model to be reconstructed. We refer the reader to the recent survey by Musialski et al. [MWA$^+$13] for a comprehensive overview of urban reconstruction algorithms.

The automatic reconstruction work of Chen and Chen [CC08] introduces a method to reconstruct polygonal faces of the model by searching for Hamiltonian circuits in graphs. They assume the existence of the complete set of planes and their neighboring information. Two planes are assumed to be adjacent if the minimum distance between their corresponding point sets is within a distance parameter. Due to erroneous and missing data prevalent in real-world data sets, we believe that one of the most challenging problems in automatic reconstruction remains the determination of neighboring information and that more sophisticated algorithms are needed to solve this problem.

Image-based approaches [WZ02, SB03] generate coarse approximations consisting of mutually orthogonal planes. The coarse models are then refined with predefined shapes to add details such as windows, doors, and wedge blocks.

In their seminal work, Debevec et al. [DTM96] introduced a hybrid method that combines geometry-based modeling with image-based modeling into one pipeline, in which the user matches edges in the photographs to the edges in the model. Parameters and relative positions of model components as well as camera parameters are computed by minimizing a non-linear photogrammetric objective function.

Recently, Nan et al. [NSZ$^+$10] presented a system, the so-called *SmartBoxes*, to quickly model architectural buildings directly over 3D point clouds. SmartBoxes assumes Manhattan world scenes [FCSS09, VAB10], and is great to reconstruct facades with repetitive axis-aligned structures. Compared to SmartBoxes, our system doesn't make any assumptions on the shape of planar surfaces, their mutual alignments and orientations.

Sinha et al. [SSS$^+$08] introduce an interactive system to generate textured models of architectural buildings from a set of unordered photographs. The user sketches outlines of planar surfaces of the scene by directly drawing on top of photographs. The drawing process is made easier by snapping edges automatically to vanishing point directions and to previously sketched edges. Compared to our approach, their system still needs a precise drawing of polygons, since they do not automatically estimate polygon boundaries, and snapping is induced by a simple proximity criterion, while our system reliably extracts adjacency relations between elements of the polygons.

The VideoTrace system proposed by van den Hengel et al. [vdHDT+07] interactively generates 3D models of objects from video. The user traces polygon boundaries over video frames. While in the work of Sinha a single image is sufficient to accurately reconstruct polygonal faces, VideoTrace repeatedly re-estimates the 3D model by using other frames.

A large number of mesh reconstruction methods have been proposed over the years: popular approaches include *Extended marching cubes* by Kobbelt et al. [KBSS01], the *Delaunay refinement paradigm* [BO05] and *implicit* approaches [KBH06, ACSTD07, SDK09]. Typically, all these methods need to generate high resolution meshes in order to recover sharp edges. More recently, Salman et al. [SYM10] have improved the accuracy of reconstructions for a prescribed mesh size while ensuring a faithful representation of sharp edges. Still, their method does not yield models of low face count and visual quality as required in this work (cf. Figure 5.15). Instead of applying an extensive post-processing pipeline (e.g., centered around Cohen-Steiner et al.'s shape approximation [CSAD04]), we propose to integrate all requirements into a single optimized algorithm: our method fits median planes to the input data, accurately reconstructs sharp edges from the intersection of these planes and generates coarse polygonal models (with the complexity defined by the number of planes). Most importantly, our method is good at handling surfaces with boundaries: all our input data is incomplete, with missing parts and holes due to the acquisition process.

Recent commercial systems such as SketchUp have been designed to quickly create 3D models from users' sketches. The *Pointools* plugin for SketchUp [Poi11] allows users to model directly over 3D point clouds, but the accuracy of the reconstruction depends on the skills and patience of the user, since the sketched geometry has to be manually aligned to the point cloud by visual inspection. In addition, the plugin offers a simple snapping tool that allows snapping the endpoint of a sketched line to a nearby point in the point cloud. In practice, this feature can be used to coarsely sketch the floorplan of a building, but is impractical for more detailed modeling tasks due to the noise inherent in point clouds, as there is no way to align a primitive to a set of points. Furthermore, gaps that appear in the modeling process have to be closed manually by moving edges. In contrast, we optimally reconstruct planar primitives by least-median fitting to the point data. Moreover, we automatically discover adjacency relations, which allows us to run a planarity-preserving optimization-based snapping algorithm to close the model.

GlobFit, recently introduced by Li et al. [LWC+11], iteratively learns mutual relations among primitives obtained by the RANSAC algorithm [SWK07]. Their system seems to be complementary to ours: While they focus on discovering global relations (like orthogonality, coplanarity, etc.) among parts of the model to correct the primitives, our strength lies in the automatic discovery of local adjacency relations between polygon elements. To produce a final model (which is not their primary goal), Li et al. only extrapolate and compute pairwise primitive intersections. While it is relatively easy to locally extend individual polygons, intersections of multiple primitives can be highly complex and reconstruction becomes non-trivial.

5.3 Overview

Our system takes as input a set of 3D points, for example from a laser scanner, photogrammetric reconstruction or similar source. The goal is to create a polygonal model suitable for interactive applications and not influenced by the noise and holes inherent in the input data. The reconstructed polygonal model will be watertight wherever feasible – in this work we shall denote such a model a *closed model*. It is important to emphasize that we do not assume our target surfaces to be closed in a topological sense.

Modeling consists of two phases: an automatic phase that creates an initial model, and an interactive phase that is aided by optimization-based snapping.

5.3.1 Automatic Phase

In the automatic phase, the input data is decomposed into subsets lying approximately on the same plane (Figure 5.1, top left). Throughout the work, we refer to these subsets as *segments*. For each such segment, boundary polygons are estimated in the *polygonalization* step (Section 5.4, Figures 5.1 top right and 5.3). The resulting model still has holes and is not well aligned.

We therefore introduce an intelligent *snapping algorithm* (Section 5.5) that constrains and optimizes the locally fitted planes and their corresponding polygons. The local fit of the planes is determined by how well the planes approximate the observed point cloud data, while the mutual spatial relations, i.e., adjacency relations between polygon elements, are iteratively computed and enforced through a non-linear optimization. This "intelligent snapping" is a crucial part of our approach: Instead of simply snapping to existing geometry or features within a given distance [SSS$^+$08, vdHDT$^+$07], we define a feature-sensitive matching and pruning algorithm to discover a robust set of adjacency relations among parts of the polygons (Figure 5.1, top right). The polygons are then aligned by enforcing the extracted relations, while best fitting to the input data and maintaining the planarity of the polygons (Figure 5.1, bottom left).

Note that while a completely automatic reconstruction of a whole building can hardly be achieved due to erroneous and missing data, our automatic phase produces results that are comparable to previous automatic systems, for example Chen and Chen [CC08], who assume the existence of a complete set of planes.

5.3.2 Interactive Phase

Since the initial geometry proposal from the automatic phase cannot guarantee a perfect solution in all cases, the interactive phase provides a simple and intuitive *sketch-based user interface* (Section 5.6) that directly interoperates with the optimization routines. Even though such manual interventions cannot be completely omitted, the novel system differs significantly from other 3D modeling techniques by exploiting the previous analysis: Due to the known supporting planes, the modeling complexity is reduced from a 3D to a 2D problem. This allows the user to model the necessary changes with a few simple and loose strokes on a flat layer, as the exact alignment is performed interactively by the optimization-based snapping algorithm (Figure 5.1 bottom left).

5.4 Polygonalization

We use a local RANSAC-based method [SWK07] to decompose the input point cloud into subsets (referred to as *segments*), each lying approximately on a plane, and a set of unclaimed points. Even though the decomposition requires normal information, the correctness of normals close to sharp edges is not critical for the subsequent estimation of plane primitives (Section 5.4.1) and the rest of our pipeline. Hence, our implementation approximates 3D normal vectors from point positions by applying a local PCA [Jol02] with fixed size neighborhoods.

A segment may consist of multiple connected components (e.g., front faces of all individual balconies on a facade), which are later separated by the boundary extraction algorithm (Section 5.4.1).

The goal of the polygonalization step is to divide the segments from the RANSAC stage into connected components, and approximate their outlines by coarse polygons. In the first step, we divide each segment into one or several connected components, and extract their ordered boundary points (Figure 5.3 top left), which act as initial polygons. Since the extracted boundaries are generally noisy, we cannot assume that we have high-quality vertex normal orientations. In the second step, we compute a smooth region around each point (Figure 5.3 top right), to which we then apply a local PCA to estimate initial 2D vertex normals (Figure 5.3 middle left). Finally, we reconstruct 2D vertex normals based on their initial values and a neighborhood relationship derived from the smooth regions (Figure 5.3 middle right), and use the reconstructed normal vectors to compute consistent vertex positions (Figure 5.3 bottom left). This process straightens the initial boundaries and thus provides a polygonal approximation of the 2D components on a coarse scale (Figure 5.3 bottom right).

5.4.1 Initialization

Plane fitting

The polygonalization is computed in 2D space defined by the segment plane. The first step is therefore fitting a plane to all the points contained in a segment obtained by RANSAC. As depicted in Figure 5.2 (left), nearby parallel structures (e.g., main facade and windows) may have been detected as a single segment. We therefore apply a *Least Median of Squares* (LMS) estimator [RL87], which consistently finds the main structure (see Figure 5.2 middle left), as it is capable of fitting a model to data that contains up to 50% outliers.

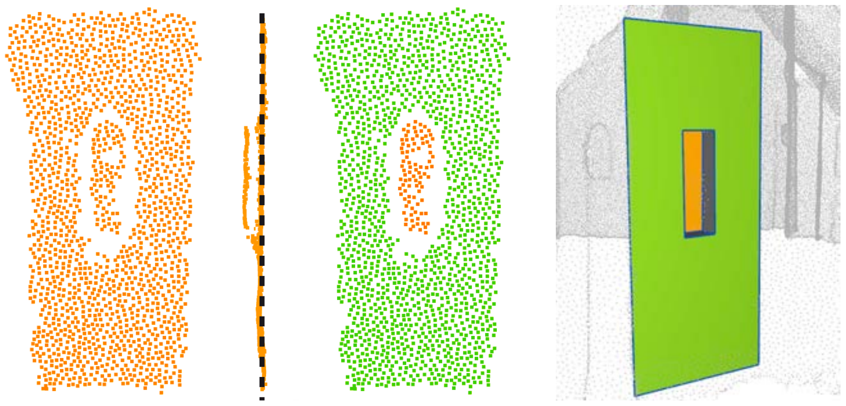

Figure 5.2: The first step of the automatic reconstruction pipeline, a local RANSAC-based method, may capture nearby parallel structures (e.g., windows and facade) as a single segment (left). The least-median of squares method is used to fit a plane to the points of the dominant structure (middle left). By applying k-means clustering (k = 2) (and subsequent automatic polygonalization and optimization) in the interactive modeling phase, a more detailed hierarchical reconstruction (middle right, right) is achieved.

Boundary extraction

In order to divide each segment into connected components and extract their ordered boundary points, we employ *2D α-shapes* [EM94] (with α controlling the number of connected components and the level of detail of their boundaries) on the segment points projected to the median plane. The family of 2D α-shapes of the set of projected segment points S is implicitly represented by the Delaunay triangulation of S. Each element (vertices, edges and faces) of the Delaunay triangulation is associated with an interval that specifies for which values of α the element belongs to the α-shape. The connectivity of the Delaunay triangulation and the classification of its elements with respect to the α-shape is then used to extract the connected components and their ordered boundary points. We estimate α using the average distance between neighboring points.

5.4.2 Polygon Straightening

The goal of this step is to robustly estimate for each boundary a coarse polygon that approximates the original shape's outline. Most surfaces used in architecture, especially at the level of detail relevant for polygonal modeling, are bounded by straight lines that meet in sharp corners. We also require from our method that it is fast, since we need interactivity during the modeling process (see Section 5.6.6).

Our method is inspired by the ℓ_1-sparse method [ASGCO10] for the reconstruction of piecewise smooth surfaces. However, we found the ℓ_1 formulation computationally too expensive, compared to our ℓ_2 approach counting only a few or even no re-weighted iterations (the ℓ_1 formulation's second-order cone programming solver needs to solve a series of linear systems of comparable dimension to our setting). We tackle stability

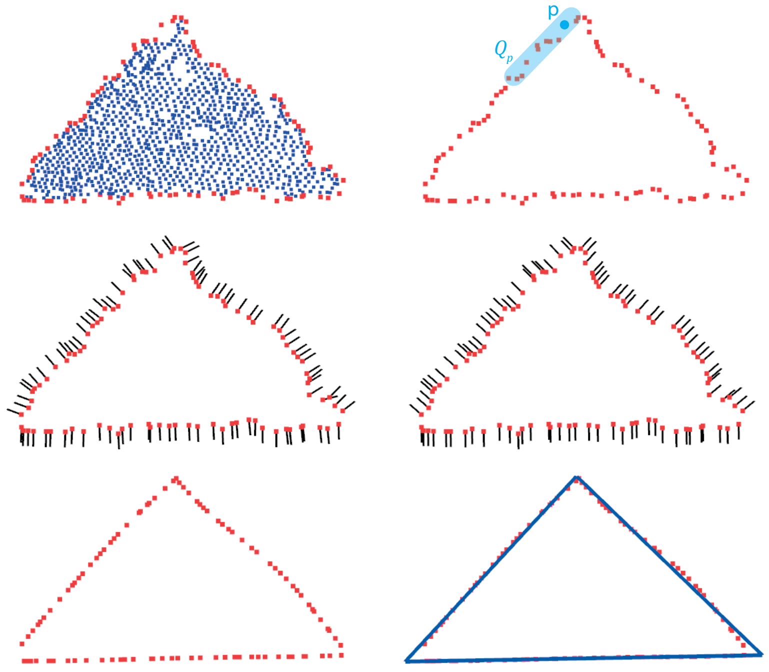

Figure 5.3: Overview of our polygonalization pipeline. Ordered boundary points (initial polygon) of a connected component are extracted (top left). By applying PCA to smooth regions Q (top right), vertex normals are initialized (middle left). Initial vertex normals are smoothed over neighboring vertices (with $\mathbf{p}$ and $\mathbf{q}$ neighboring if they are mutually contained in their respective smooth regions) (middle right) and used to compute consistent vertex positions (bottom left). Finally, a corner detection algorithm extracts the approximating polygon.

issues inherent to least-squares approaches (such as sensitivity to outliers) with statistical methods. In particular, we rely on the *forward search* method [AR00] and present a novel method combining the simplicity of least-squares minimization with the strength of robust statistics.

Neighborhood estimation

Similar to Fleishman et al.'s approach [FCOS05], we classify a locally smooth region around each vertex by applying the forward search method, which preserves sharp features and is robust to noise and outliers. The main idea in forward search is to start from a small outlier-free neighborhood Q and to iteratively extend the set Q until a termination criterion is met. Starting from Q and the model (in our case, a line) that is fitted to the points in Q, one iteratively adds one point to the set Q (the point with lowest residual) and updates the model at each iteration, until the diameter of Q exceeds a threshold d_{max}. Figure 5.3, top right, shows an example of a smooth region around a point $\mathbf{p}$ computed with forward search. Two polygon vertices $\mathbf{p}$ and $\mathbf{q}$ are said to be neighboring, if $\mathbf{q} \in Q_p$ and vice versa. The diameter threshold d_{max} is the only parameter that affects the output of our polygonalization method. Since d_{max} controls the local region sizes, we avoid high values of d_{max} (usually set to minimum expected feature size) to prevent oversmoothing of sharp features.

2D Normal estimation

We then estimate consistently oriented vertex normal vectors (Figure 5.3, middle right) based on the neighborhood relationship computed by the prior step. As in Avron et al. [ASGCO10], our least-squares minimization to reconstruct vertex normals consists of two terms and is formulated as:

$$E_1 = \sum_{(p,q)\in N} w_{p,q} \|\mathbf{n}_p - \mathbf{n}_q\|^2 + \lambda \sum_p \|\mathbf{n}_p - \mathbf{n}_p^0\|^2. \tag{5.1}$$

The first term minimizes the normal differences and extends over the set N of all neighboring vertices. The second term prevents the vertex normals $\mathbf{n}$ from deviating too much from their initial orientations $\mathbf{n}^0$. The weighting function $w_{p,q}$ in Equation 5.1 penalizes variations in normal directions, and is given by the Gaussian filter:

$$w_{p,q} = e^{-(\theta_{p,q}/\sigma)^2}, \tag{5.2}$$

where $\theta_{p,q}$ is the angle between the normal vectors $\mathbf{n}_p$ and $\mathbf{n}_q$ and σ is a parameter set to 20 degrees in all our examples.

The normal vectors are initialized (cf. Figure 5.3 middle left) by applying PCA to the local smooth regions computed by the prior step. The consistency of the initial normal orientations is provided by the order of the boundary points.

2D Polygon smoothing

We then compute consistent vertex positions (Figure 5.3, bottom left) by displacing polygon vertices in normal direction, i.e.,

$$\mathbf{p}' = \mathbf{p} + t_p \mathbf{n}_p.$$

Similar to Avron et al. [ASGCO10], the new vertex positions $\mathbf{p}'$ are computed as the minimizer of the energy function

$$E_2 = E_{2,s} + E_{2,init} \tag{5.3}$$

with

$$E_{2,s} = \sum_{(p,q)\in N} w_{p,q} \Big(\big|(\mathbf{p}' - \mathbf{q}') \cdot \mathbf{n}_q\big|^2 + \big|(\mathbf{q}' - \mathbf{p}') \cdot \mathbf{n}_p\big|^2 \Big)$$

and

$$E_{2,init} = \mu \sum_p t_p^2.$$

The first term smoothes (straightens) the polygon boundary by minimizing the deviation of $\mathbf{q}$ from the tangent through $\mathbf{p}$ weighted according to the confidence measure $w_{p,q}$ (Equation 5.2) and vice versa. The second term prevents the polygon vertices from deviating too much from their initial positions and as a consequence avoids shrinking of the polygon.

Both functionals E_1 and E_2 (Equations 5.1 and 5.3) are minimized using a Gauss-Newton method. Since we use forward search to reconstruct outlier-free regions, we require only three re-weighted iterations to minimize E_1 and a single iteration to minimize E_2. The weight parameters λ and μ are set to 0.1 in our data sets.

5.4.3 Polygon Extraction

The process described above provides us with a set of straightened boundary points with high-quality vertex normals, which are then used by a simple corner detection algorithm to extract approximating coarse polygons (Figure 5.3, bottom right).

We classify a vertex as an edge vertex if its normal vector is almost parallel to the normal vector of the succeeding and preceding vertex, respectively. Then we approximate sequences of edge vertices in a least-squares sense and obtain edge lines. A corner is detected at the intersection point of two successive lines. Note that vertices that are isolated in our neighborhood relationship graph are potential outliers or influenced by noise inherent in the initial boundaries, and thus not used by the corner detection algorithm.

5.5 Polygon Soup Snapping

The result of the polygonalization step is a soup of unconnected polygons $\mathcal{P} = \{P_1, \ldots, P_n\}$. The snapping process aims at closing the holes between the polygons of the polygon soup, and is used both in the initial automatic reconstruction and during interactive modeling. Snapping iteratively pulls polygon vertices towards other polygons, while simultaneously re-fitting $\mathcal{P}$ to the underlying point cloud and preserving the planarity of polygons. Each iteration consists of the following two steps:

- *Robust search for adjacencies*, which for each vertex identifies the possible matches to other vertices, edges or faces and discards false ones.
- *Optimization*, which enforces the set of discovered adjacency relations to snap the polygon soup together.

The process terminates when the polygon soup stabilizes, i.e., it becomes a closed model and satisfies the requirements given by the constraints. Our snapping process is related to Botsch et al. [BPGK06] and Kilian et al. [KFC$^+$08], however our system requires the optimization for various other constraints and in particular, the relations between the polygons are not known a priori. We now describe the steps in detail.

5.5.1 Robust Search for Adjacencies

The problem of matching the elements of the polygon soup to a closed model in a feature-aware manner is inherently ill-defined. The expected bad quality of the real-world data sets and the lack of any high-level input to the reconstruction pipeline (such as shape templates or semantic information) prevent a rigorous mathematical definition. Instead, we propose an automatic and robust algorithm based on stable vertex-vertex/edge/face and edge-edge matches.

Adjacencies in the model are discovered by searching for matches between polygon elements. There are five mechanisms that constrain the allowed matches: (1) An auxiliary *global parameter* r_{max} defines the maximal gap size to be closed in the model. (2) *Intrinsic stability* locally avoids self-intersections, flip-overs, edge and diagonal collapses. (3) An extended set of matching candidates allows more degrees of freedom (thus a more connected model) where (2) is too restrictive. (4) *Local pruning* fixes problems mostly introduced by (3), and (5) *global pruning* prevents degeneration of polygons (especially of thin features) by considering global issues of matches affecting more than two polygons. We first describe the different match types, constrained by (1-3), and then show local and global pruning. Please note that the choice of r_{max} doesn't influence the stability of the pruning algorithms, but only defines the maximal gap size.

Vertex-vertex matching

We define an adaptive search radius for each vertex of the model. The requirement of intrinsic stability bounds the search radius to half the minimal distance from $\mathbf{p}$ to the polygon's boundary, $d_\partial(\mathbf{p}) := \min_{\mathbf{e}\in\partial P\setminus\mathbf{p}} d(\mathbf{p},\mathbf{e})$, where $\partial P \setminus \mathbf{p}$ denotes the polygon boundary after removal of $\mathbf{p}$ and its incident edges. Half the distance d_∂ prevents vertices being matched across polygon edges (self-intersections, flip-overs) or vertices (edge or diagonal collapses). To respect the given upper bound, we define $r(\mathbf{p}) := \min(r_{max}, d_\partial(\mathbf{p})/2)$ as the *adaptive search radius* of $\mathbf{p}$.

The candidate set of matches for a vertex $\mathbf{p}$ comprises all vertices of $\mathcal{P} \setminus P$ within search distance $r(\mathbf{p})$. If this candidate set is empty, the closest vertex in $\mathcal{P} \setminus P$ (if not further than r_{max}) is included. By doing so, we may violate intrinsic stability intentionally to maintain sufficient degrees of freedom. A subsequent pruning step, described below, will restore validity at a later stage, if necessary. We define $r_c(\mathbf{p}) := \max(r(\mathbf{p}), \min(d_c, r_{max}))$ (with d_c being the distance of $\mathbf{p}$ to its closest vertex in $\mathcal{P} \setminus P$) as the *extended search radius* of $\mathbf{p}$.

A priori, two vertices $\mathbf{p}$ and $\mathbf{q}$ are considered matching, if they are mutually included in their respective extended search radii,

$$\|\mathbf{p} - \mathbf{q}\| \leq \min(r_c(\mathbf{p}), r_c(\mathbf{q})).$$

Finally, two matching vertices are supposed to collapse into a corner point at the later optimization stage. Such a corner point is incident to the intersection line l of the two corresponding supporting planes. Consequently, we further require a pair of matching vertices to be in feasible distance to l,

$$d(l, \mathbf{p}) \leq r_c(\mathbf{p}) \quad \text{and} \quad d(l, \mathbf{q}) \leq r_c(\mathbf{q}).$$

Please note that the computation of l is numerically stable, as the RANSAC stage gives a priori knowledge about polygons in the same supporting plane.

Vertex-edge matching

In a similar fashion to vertex-vertex matches, we establish correspondences between vertices and edges. We assign a search radius to each edge $\mathbf{e} = (\mathbf{p}_0, \mathbf{p}_1)$ as the minimal search radius of its end points, $r(\mathbf{e}) = \min(r(\mathbf{p}_0), r(\mathbf{p}_1))$. A vertex $\mathbf{p}$ is matched to an edge $\mathbf{e}$ if its orthogonal projection onto the line spanned by $\mathbf{e}$ is in the edge's interior, and – analogous to a vertex-vertex match – the two following expressions hold true:

$$d(\mathbf{p}, \mathbf{e}) \leq \min(r(\mathbf{p}), r(\mathbf{e})),$$

and

$$d(l, \mathbf{p}) \leq r(\mathbf{p}) \quad \text{and} \quad d(l, \mathbf{e}) \leq r(\mathbf{e}),$$

where l is the common intersection line of the corresponding supporting planes.

Other matches

To complete the survey of matches, a vertex $\mathbf{p}$ is paired with a face $\mathbf{f}$ if its orthogonal projection onto the plane spanned by $\mathbf{f}$ is in the face's interior and no further than search radius $r(\mathbf{p})$.

Based on the vertex-vertex and vertex-edge matches, we may further derive edge-edge matches: Two edges are said to match if their endpoints either induce two vertex-vertex matches, a vertex-vertex and a vertex-edge match or two vertex-edge matches. Edge-edge matches are used only in the matching and global pruning stage and not for optimization, as their contribution to reconstruction is implicitly included through vertex-vertex/edge matches. The same holds in an even stricter sense for edge-face and face-face matches, which are either implied by vertex-vertex/edge/face matches or are not present in the data due to occlusion in the acquisition process.

Local pruning

The vertex-vertex matching yields generally stable results. A few false matches, which result from the inclusion of closest vertices in candidate sets, are corrected in the following pruning step: Consider two or more vertices $\mathbf{q}_i$ of polygon Q being matched to a vertex $\mathbf{p} \in P$. This clearly violates the intrinsic stability requirement for Q and we remove all but the closest matching pair. This and all subsequent pruning steps are implemented on a graph representation $G = (V, E_M)$ of the matches, with all vertices and edges of $\mathcal{P}$ comprising V and the edge set E_M being given by the set of all matches obtained from

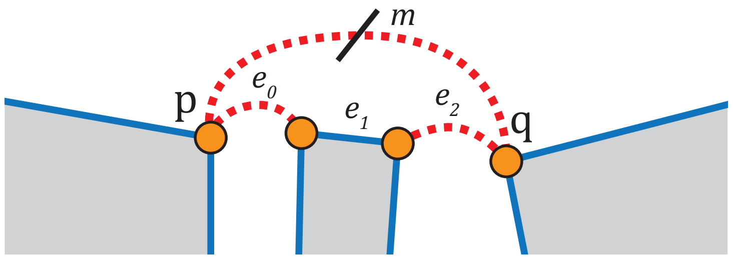

Figure 5.4: The false match $m = (\mathbf{p}, \mathbf{q})$, which forces the center polygon's edge e_1 to collapse and thus violates the intrinsic stability, is reliably detected by our global pruning strategy.

above (except vertex-face matches). Pruning at this point boils down to investigating all one-ring neighborhoods of G.

Similar to vertex-vertex matching, intrinsic stability demands the pruning of those vertex-edge matches where a vertex corresponds to multiple non-adjacent edges of a polygon. This can naturally happen due to overlapping search cylinders (with radii $r(\mathbf{e})$) around edges. Using the graph G, we compress this subset of matches to the closest vertex-edge match.

Global pruning

Up to now, the matches have been obtained on a local level only. They disregard any global issues affecting more than two polygons. Consider the situation in Figure 5.4, with three polygons stringing together and several of the corner points being matched. The vertex-vertex match between $\mathbf{p}$ and $\mathbf{q}$, jumping the center polygon, is not feasible, as it implies a degeneration of the center polygon's edge. Such a degeneration happens when certain polygon elements (vertices/edges) are connected through matches so that they form a *cycle*. The reason is that we have to assume that all polygon elements connected through matches might be joined to the same location during the optimization phase. In this section we therefore present our approach to define and find such cycles. We also show a second approach based on geometric tests for cases where the detection of cycles doesn't necessarily imply a degeneration.

For the example in Figure 5.4, to detect the contraction of the edge e_1, it would be sufficient to extend the edge set E_M in the matching graph G defined earlier by the actual polygon edges, and find the cycle (m, e_0, e_1, e_2) in the resulting graph. However, there are many other cases that would lead to a polygon degeneration, namely, whenever a match would cause two elements of a polygon to contract, see Figure 5.5 for several examples.

We therefore introduce an *extended matching graph* G_e that prevents all these internal contractions by representing them explicitly through so-called constraint edges. Formally, $G_e = (V, E_e)$ is a graph with V comprising the polygons' elements (vertices and edges,

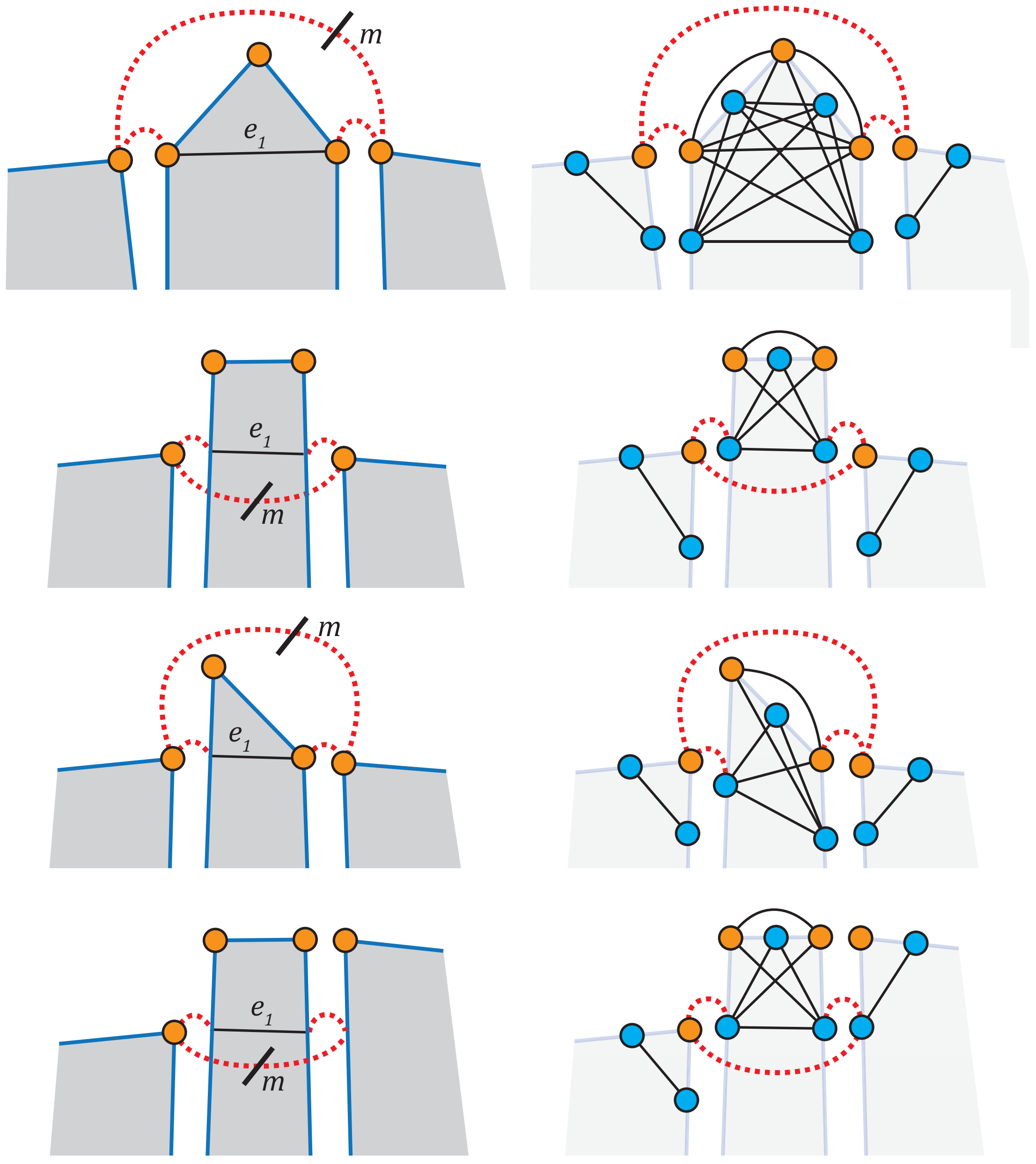

Figure 5.5: Left column: Examples of false matches m which would cause the constraint edges (denoted by e_1) to contract. Right column: We reliably avoid such false matches by searching for cycles in the extended matching graphs (only the matches of the cycles shown here).

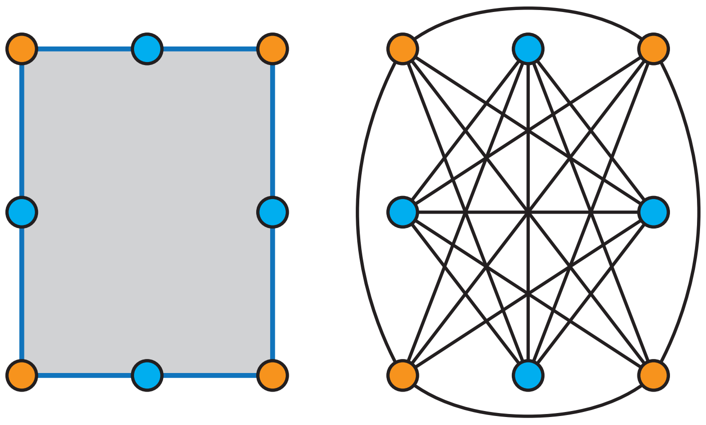

Figure 5.6: Construction of the extended matching graph $G_e = (V, E_e)$: Each polygon's vertices (orange) and edges (blue) constitute the node set V (left). The edge set E_e combines the set of all vertex-vertex/edge and edge-edge matches (not shown here), and the constraint set. The latter connects all pairs of elements in V of the same polygon, except polygon vertices with their incident polygon edges (right).

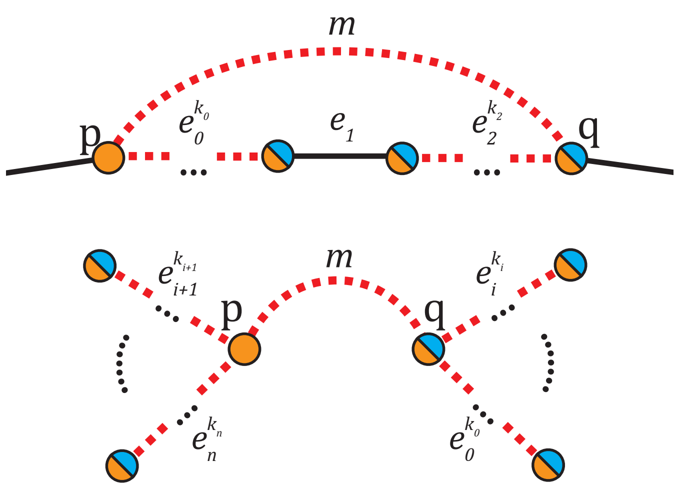

Figure 5.7: Illustration of our global pruning idea: We search for each vertex-vertex/edge match $m = (\mathbf{p}, \mathbf{q})$ an edge cycle with only one constraint edge (denoted by e_1) in the extended matching graph G_e (top). If such a cycle exists, we prune m (cf. Figures 5.4 and 5.5). Otherwise, we search for match-paths $(e_l^{k_l}, m, e_j^{k_j})$ in the actual matching graph G (bottom). Subsequent matches in the paths induce further matches (cf. Figure 5.8), which are then used to geometrically verify whether m leads to polygon degenerations.

see Figure 5.6 left). The edge set $E_e = E_M \cup E_C$ combines the set of matches E_M, and the *constraint set* E_C. The latter connects all pairs of elements in V of the same polygon, except polygon vertices with their incident polygon edges (see Figure 5.6 right). Figure 5.5 shows several examples where the detection of an appropriate cycle containing a constraint edge (denoted by e_1) in G_e causes a false match to be pruned.

Our strategy is now to determine for every vertex-vertex/edge match $m = (\mathbf{p}, \mathbf{q}) \in E_M$ whether it is part of a "harmful" cycle. We first note that we only look for cycles $c(m)$ where m is directly connected to another match on either side, i.e., $c(m) = (m, e_0, \ldots, e_n)$ with $e_0, e_n \in E_M$. The reason is that the degeneration of the constraint edges of $\mathbf{p}$ and $\mathbf{q}$'s polygons is already handled in the local pruning phase. Further, we ignore cycles containing more than one constraint edge, for reasons explained further below. Thus, we look for cycles $c(m) = (m, e_0^{k_0}, e_1, e_2^{k_2})$, with $e_1 \in E_C$ and $e_i^{k_i} = (e_{i,1}, \ldots, e_{i,k_i}) \in E_M^{k_i}$ ($e_i^{k_i}$ abbr. as e_i for $k_i = 1$), $k_i \geq 1$ (see Figure 5.7 top).

If we find such a cycle, the corresponding match can be pruned directly, because the cycle forces the constraint edge e_1 to contract (see Figures 5.4 and 5.5). However, in some rare cases (where polygons' elements meet at non-manifold vertices/edges of the final model) we also observed the pruning of a few "correct" matches. For the convergence of n polygon elements to the same location, $\binom{n}{2}$ connections (matches) are possible, but only $n-1$ involving all those n elements are sufficient. Thus, in practice, the pruning of a few "correct" matches doesn't indicate a problem.

Most of the degenerations in the model can already be avoided by pruning cycles with one constraint edge. However, there are also some cases involving several constraint edges, see Figure 5.8 right. Unfortunately, this situation cannot be detected unambiguously by searching for cycles, as can be seen in Figure 5.8 left. To solve this problem, we present a more general approach that is based on investigating sequences of matches. Such sequences *induce* further matches between the elements they connect, in the sense that in the optimization phase, these elements will also be joined. We thus need to verify whether the induced matches do not cause polygon degenerations.

Formally, for a match m, we search for paths $(e_l^{k_l}, m, e_j^{k_j})$ (with $k_l, k_j \geq 0$) in the actual matching graph G (see Figure 5.7 bottom). We then check whether all of the induced matches m_i are in E_M as well. For every match that is not in E_M, we need to verify geometrically whether it would lead to a polygon degeneration. Here we note that an induced match does not necessarily join the attached elements directly (e.g., two subsequent vertex-edge matches $m_1 = (\mathbf{p}, \mathbf{e})$ and $m_2 = (\mathbf{e}, \mathbf{q})$ do not induce the vertex-vertex match $m_3 = (\mathbf{p}, \mathbf{q})$, but both vertices project to the same edge). To geometrically verify whether an induced match leads to any degeneration, we project the vertices and/or edge endpoints of the match onto the common intersection line l of the polygons' supporting planes. We prune m if one of the thus-modified polygons (with projected vertices/edges) has a flipped normal vector orientation (flip-over) or has self-intersections. Note that the number of paths to investigate is typically low because paths containing only matches stay localized.

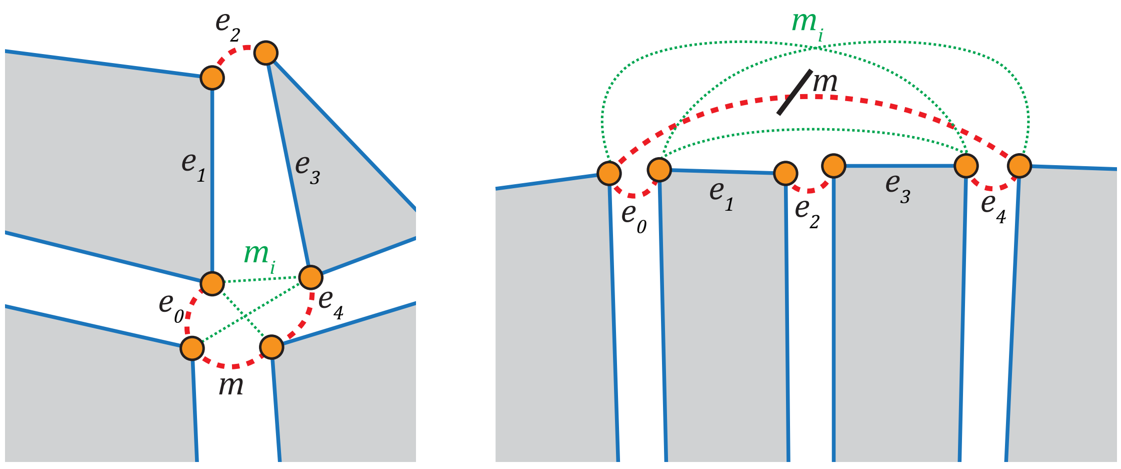

Figure 5.8: Cycles (here $(m, e_0, e_1, e_2, e_3, e_4)$) in the extended matching graphs G_e containing more than one constraint edge can be "harmful" (right) or not (left), and thus are not a good indicator for pruning. Instead we geometrically verify whether the induced matches (denoted in green by m_i) resulting from subsequent matches (e_0, m, e_4) in the matching graphs G cause polygon degenerations. Please note that in the right image, m is selected in the matching phase due to large search radii of the corresponding vertices (we show only a part of the polygons here).

5.5.2 Optimization

Based on the discovered adjacencies, we transform the polygons to optimally align with each other, while preserving their planarity and fitting to the input point cloud.

As in Kilian et al. [KFC$^+$08], we introduce a Cartesian coordinate system in the plane of each $P \in \mathcal{P}$, with origin $\mathbf{o}$ and basis vectors $\mathbf{f}_1$ and $\mathbf{f}_2$, and represent a point $\mathbf{p} \in P$ by the coordinates (p_x, p_y), so that $\mathbf{p} = \mathbf{o} + p_x\mathbf{f}_1 + p_y\mathbf{f}_2$. During the optimization, in order to reduce the spatial gaps between adjacent polygons, the coordinates (p_x, p_y) are displaced, while the Cartesian coordinate systems undergo a spatial motion. We linearize the spatial motion of each coordinate system by representing the displacement of each point through the velocity vector field of an instantaneous motion, given by $\mathbf{v}(\mathbf{x}) = \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}$. Thus, the position of a vertex $\mathbf{p} \in P_i$ during the optimization can be written as

$$\mathbf{p} = \mathbf{o}_i + \bar{\mathbf{c}}_i + \mathbf{c}_i \times \mathbf{o}_i + p_x(\mathbf{f}_{i_1} + \mathbf{c}_i \times \mathbf{f}_{i_1}) + p_y(\mathbf{f}_{i_2} + \mathbf{c}_i \times \mathbf{f}_{i_2}) \tag{5.4}$$

in the unknown parameters $\mathbf{c}_i$, $\bar{\mathbf{c}}_i \in \mathbb{R}^3$ of the velocity vector field attached to P_i and in the unknown coordinates (p_x, p_y) (this can be derived by applying the displacement $x' = x + v(x)$ for $x \in \{\mathbf{o}_i, \mathbf{o}_i + \mathbf{f}_{i_1}, \mathbf{o}_i + \mathbf{f}_{i_2}\}$).

Snapping

With the adjacency relations discovered by the prior step, we measure the *snapping error* as

$$E_{snap} = \sum_{i,j,k,l} d^2(\mathbf{p}_i, \mathbf{p}_j) + d^2(\mathbf{p}_i, \mathbf{e}_k) + d^2(\mathbf{p}_i, P_l),$$

where $d^2(\mathbf{p}_i, \cdot)$ denotes the distance of vertex $\mathbf{p}_i$ to the vertex $\mathbf{p}_j$, edge $\mathbf{e}_k$ and face P_l, respectively.

Point cloud deviation

For the polygon soup $\mathcal{P}$ not to deviate too much from the input point cloud, we use the *reference term*

$$E_{ref} = \sum_{l=1}^{|\mathcal{P}|} \sum_{i=1}^{|P_l|} d^2(\mathbf{p}_{i(l)}, P_l^{init}). \tag{5.5}$$

The above equation minimizes the sum of squared distances of vertices $\mathbf{p}_{i(l)} \in P_l$ to the initial planes P_l^{init} (see Section 5.4.1). Manually sketched polygons without underlying segments (Section 5.6.3) are excluded from Equation 5.5.

Orthogonality

In order to meet orthogonality constraints that naturally exist in urban environments, we include the following two terms

$$E_{\perp_1} = \sum_{i,j} w_{ij} (\mathbf{n}_i \cdot \mathbf{n}_j)^2 \tag{5.6}$$

and

$$E_{\perp_2} = \sum_{l=1}^{|\mathcal{P}|} \sum_{i=1}^{|P_l|} w_{i(l)} (\mathbf{e}_{i(l)} \cdot \mathbf{e}_{(i+1)(l) \bmod |P_l|})^2, \tag{5.7}$$

which measure the orthogonality of adjacent polygons and successive polygon edges, respectively. With the unit normal vector of P given by $\mathbf{n} = \mathbf{f}_1 \times \mathbf{f}_2$, Equation 5.6 extends over all pairs of polygons, with $w_{ij} = 1$ for adjacent polygons with normals deviating from orthogonality by less than $\frac{\pi}{9}$, and zero otherwise. Optimizing for orthogonality of polygon boundary edges $\mathbf{e}_{i(l)}$ in Equation 5.7 (with $w_{i(l)}$ defined similar to w_{ij} for polygons) might result in degenerating edges of vanishing length. This problem is in particular evident in case of missing geometry and is overcome by minimizing the sum of squared distances to current vertex positions $\mathbf{p}'_i$ as follows:

$$E_{cur} = \sum_i d^2(\mathbf{p}_i, \mathbf{p}'_i).$$

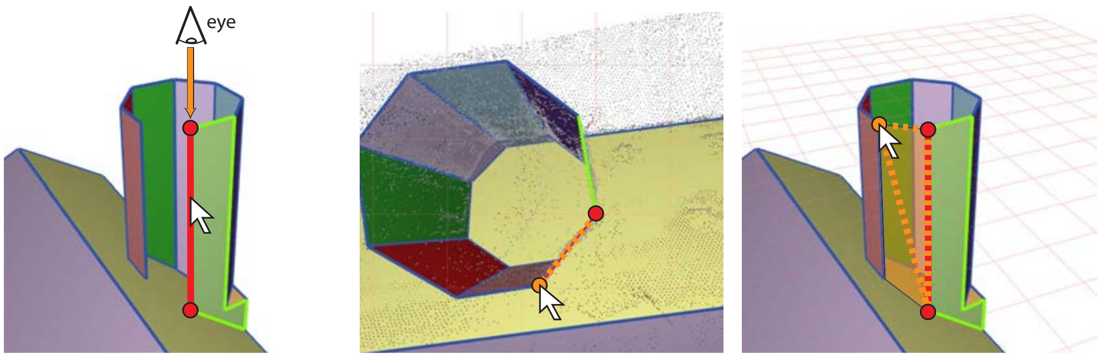

Figure 5.9: Sketching of a polygon in an area without an underlying segment: An edge of an adjacent polygon is selected by moving the mouse over it (left). The camera view direction aligns with the edge, and a perpendicular sketching plane is used to sketch a point (middle). The new plane is defined by the edge vertices and the sketched point, and initialized with a rectangular polygon (right).

Global energy and weights

The above energy terms are combined into the objective function

$$E = \lambda_{snap} E_{snap} + \lambda_{ref} E_{ref} + \lambda_{\perp}(E_{\perp_1} + E_{\perp_2}) + \lambda_{cur} E_{cur},$$

which is minimized using a Gauss-Newton method. Instead of decoupling the optimization as in Kilian et al. [KFC+08], we solve simultaneously for the parameters of the velocity vector fields attached to the polygons and the 2D coordinates of the vertices, resulting in a non-linear optimization problem due to the products $p_x \mathbf{c}_i$ and $p_y \mathbf{c}_i$ (Equation 5.4). Transforming the Cartesian coordinate system of P_i corresponding to the pair $(\mathbf{c}_i, \bar{\mathbf{c}}_i)$ would not yield a rigid body motion, but an affine one. Therefore, we use the underlying helical motion, which ensures rigidity, as described by Pottmann et al. [PHYH06]. The weights λ allow additional control of the optimization. We used $\lambda_{snap} = 1$, $\lambda_{ref} = 0.5$, $\lambda_{\perp} = 0.01$ and $\lambda_{cur} = 0.1$ for all the models shown in this chapter.

5.6 Interactive 2D Modeling

On top of automatic polygon creation and polygon soup snapping, we propose an interactive editing and modeling system that provides a novel way of user-guided 3D content creation and reconstruction. All user interaction is reduced to sketch-like approximate 2D operations by automatically choosing an appropriate 2D modeling space based on segment planes in the underlying point cloud. Implicitly dropping one dimension drastically reduces interaction complexity and thereby reduces overall modeling effort. At the same time, consistency and accuracy of the reconstructed model increase due to the interactive optimization performed after each modeling step.

5.6.1 Plane Selection

All modeling operations are based on and limited to planes. The active plane is chosen by selecting a polygon or a point cloud segment with a single mouse click. On demand, the camera's view direction aligns to the plane normal, allowing a “2D top-down view” onto it.

5.6.2 Polygon Editing

Polygons, which have either been created automatically or sketched by the user (see below), can be modified arbitrarily. Once in focus, the editing steps are comparable to a simple 2D vector graphics editing program: By moving the mouse cursor over the corresponding region, a vertex, an edge or the whole polygon is chosen for manipulation and can be dragged anywhere on the underlying plane. Vertices can be added by right-clicking on an edge, and removed by right-clicking on a vertex.

Individual polygons lying on the same segment plane, which may occur due to holes in the point data, can easily be merged by dragging polygons over each other, resulting in a single polygon consisting of their combined convex hull. We opted for this fast, interactive method of solving such cases instead of closing the holes automatically, as there are various situations in which such individual coplanar polygons are intended (e.g., front faces of balconies on a facade).

5.6.3 Polygon Sketching

In sketching mode, the selected plane also acts as a drawing area for new polygons. Manual sketching is applied whenever a segment has no polygons assigned (too few points), or if the existing polygon has been estimated wrongly due to noisy and missing data. In some cases, it can even be faster to replace the polygon by a new one instead of repairing it. Sketching is performed by approximately clicking the new vertex positions on the sketch plane. Existing polygons overlapping the new one are automatically removed.

Since we are dealing with noisy data, sparsely sampled parts of the model will most likely not be found in the RANSAC stage, excluding both automatic polygonalization as well as plane-based sketching in these areas. We therefore implemented an intuitive way to model arbitrary planes adjacent to existing polygons (see Figure 5.9): The user chooses an edge $\mathbf{e}$ of a polygon on which the new plane should attach to by hovering over it with the mouse cursor. By entering the sketching mode, the camera view direction aligns with the edge (i.e., $\mathbf{e}$ is only visible as a point then), and the user selects a point $\mathbf{p}$ on the plane perpendicular to the selected edge. The new plane is built using $\mathbf{p}$ and the vertices of $\mathbf{e}$, and initialized with a rectangular polygon.

5.6.4 Interactive Optimization

It is important to note that both editing and sketching operations only have to be performed very coarsely. As long as the approximate shape of the polygon is given,

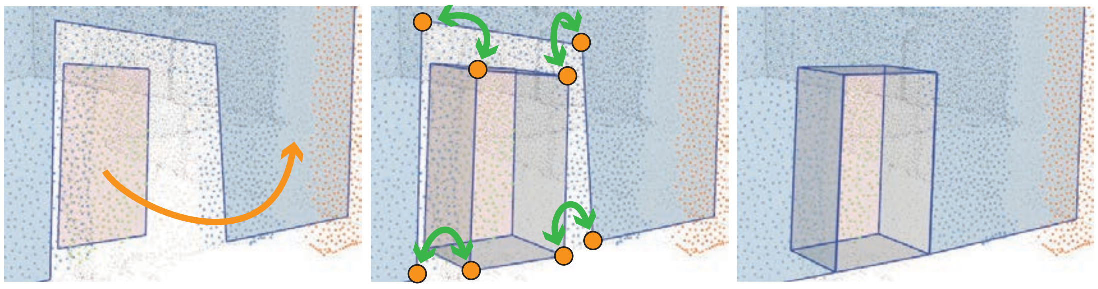

Figure 5.10: A hierarchy relation between a dominant facade plane and a door is defined with a single click (left). Side faces are automatically extruded (middle) and contribute to the snapping process: Due to the newly found matching pairs, a continuous surface is in this case generated within a few optimization iterations (right).

automatic snapping will align vertices with other parts of the model by favoring right angles while simultaneously re-fitting the polygon soup to the underlying point cloud. Optimization is therefore interleaved with each individual modeling step, providing the user with immediate feedback. When sketching completely new parts of a model, interactive optimization can also be switched off on-demand.

5.6.5 Hierarchies

The noisier and more sparsely sampled the faces to reconstruct are, the less likely it is that a suitable plane to sketch on can be found in the RANSAC stage. Depending on the chosen acquisition angle and technique, some faces may not even be depicted in the point cloud data at all. Despite the possibility to easily define arbitrary planes as described above, modeling the side faces of features like balconies, bays or windows remains a time-consuming and tedious task. We therefore allow the definition of hierarchy relations between the polygons by "connecting" a child polygon to a parent polygon with a single click: The corresponding side faces are then implicitly generated by extruding the child polygon's edges to its parent plane, reducing the modeling time to a few seconds (the corresponding holes in the parent polygons are created using a simple difference set operation $P_{Parent} \backslash P_{Child}$). This approach was used for example in Figure 5.2 right.

The generated side faces also contribute to the interactive optimization, which proves to be extremely useful in cases like doors situated at the bottom of a parent facade plane (see Figure 5.10): the vertices of the parent polygon lying in the front do not have to be manually edited, but are automatically attached to the side faces, generating a continuous surface with a single click.

Our hierarchy definition is extremely useful to model thin features (see Figure 5.11), as child polygons (belonging to different parent polygons) may snap to each other.

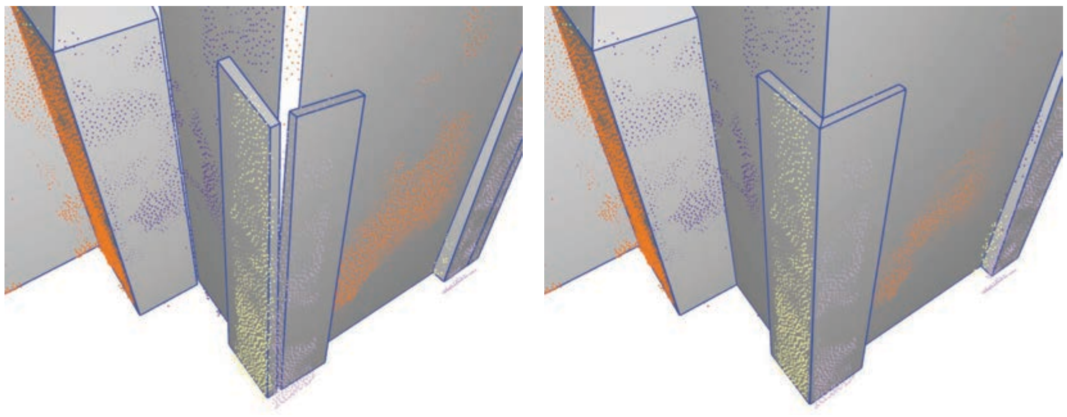

Figure 5.11: The snapping is not restricted to the reconstruction of the main structure and to child-parent relations, but it can also handle child-child relations to reconstruct detailed geometry: Two child polygons and their parent polygons after the manual segment division and subsequent polygon extraction (left). The child polygons snap to each other and the corresponding side polygons are projected correctly to the main structure to form the column (right).

5.6.6 Manual Segment Division

As discussed in Section 5.4.1 and shown in Figure 5.2 (left), nearby parallel structures may be captured as a single segment. Therefore, we offer the user the opportunity to manually divide (Figure 5.2 middle right) a selected segment by applying *k-means clustering* ($k = 2$). For the emerging segments, new polygons are automatically created (see Section 5.4, Figures 5.2 middle right and right). By combining the manual segment division with the hierarchy definition, complete facades including windows, balconies and doors can be modeled within seconds (see Figures 5.1 bottom right, 5.10 and 5.11).

5.7 Results

We have tested our reconstruction and modeling pipeline on a variety of data sets, including six point clouds obtained from photogrammetric methods (Figures 5.12 and 5.18), a laser scan (Figure 5.15), and a synthetic model (Figure 5.16). Figure 5.18 demonstrates the individual steps of our framework. Figures 5.12, 5.14 and 5.13 illustrate several applications. Figure 5.15 compares a model created using our system to the results of mesh reconstruction and decimation algorithms. The synthetic model, Figure 5.16, is used to validate the accuracy of our algorithm.

5.7.1 Performance and Scalability

In all our test scenes, interactivity could easily be maintained during the modeling sessions on a standard PC workstation (Intel i7 920 CPU with 2.67GHz, 4GB RAM): The computationally most expensive step after a modeling operation, matching and pruning, is computed within an average time of 0.2 seconds, while the optimization and the update of the rendering scene graph only take a few milliseconds to perform. We solve the sparse systems of linear equations at each Gauss-Newton iteration by a sparse QR factorization [Dav11].

Note that in our current implementation, the adjacency (matching) graph is completely rebuilt from scratch after each modeling step. In our test scenes, we experienced an adjacency graph rebuild time of one second as the worst case. By building the graph only once and updating it locally after each modification, the computational effort for the graph update could be decoupled from the geometric complexity, removing this potential bottleneck and keeping the modeling process interactive in larger-scale scenes.

5.7.2 Convergence

Solving the problem presented in Section 5.5 is a challenging task since we deal with an optimization problem that is

1. non-smooth: the set of computed adjacencies is a discrete variable,
2. non-linear: due to the simultaneous optimization of the parameters (cf. Section 5.5.2), and
3. constrained: the polygons shall remain planar.

To account for (1), we decouple the computation of the adjacencies from the rest of the optimization: at each iteration, we first fix the position of the polygons' vertices and search for adjacencies. Then, we fix the adjacencies and optimize vertex positions.

To account for (2), we choose a Gauss-Newton method (which approximates the distance function) for the smooth optimization and provide a good initialization of the problem (Section 5.4), which is known to be necessary in the solution of non-linear optimization problems (see e.g. [Kel99]).

To account for (3), we attach a Cartesian coordinate frame to each polygon and linearize its motion (which is again an approximation).

Due to the underlying characteristics of the given optimization problem, there is no guarantee that a global minimum can be found in an acceptable time. However, the results shown in this section indicate that we manage to find an aesthetically pleasing and accurate solution in a few iterations.

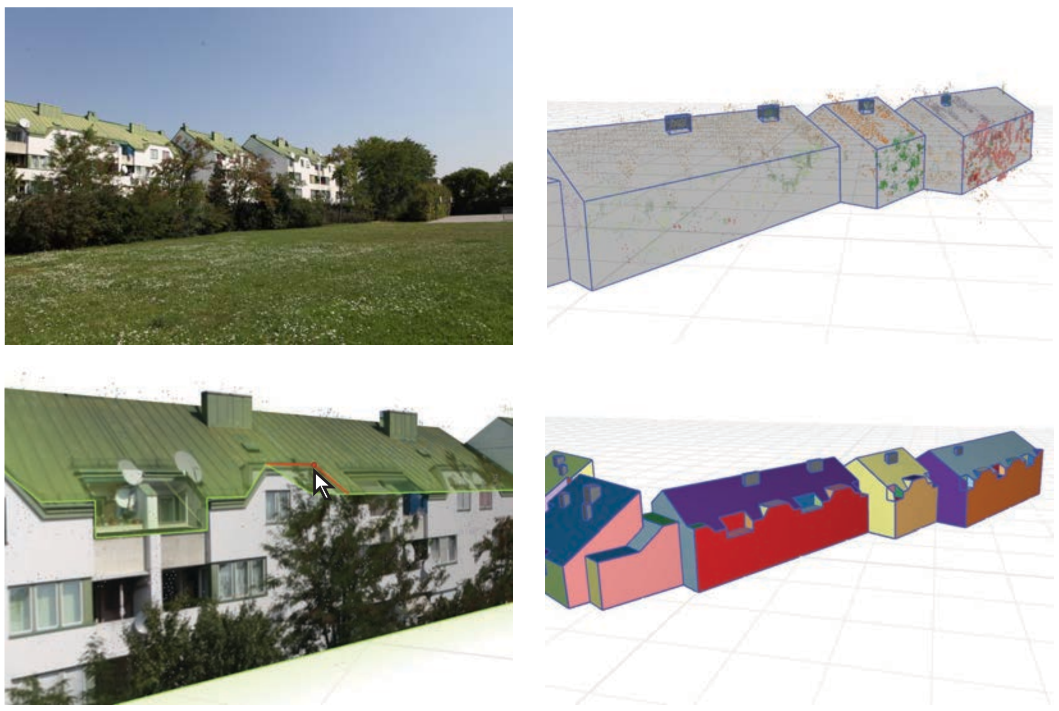

Figure 5.12: Reconstruction of a building complex occluded by trees and bushes. Top left: An example photo of the data set. Top right: The segments extracted from the sparse point cloud (obtained from a photogrammetric approach) and the main structures after approximately three minutes of optimization-aided modeling. Bottom left: The reprojected images help the user to modify the polygon boundaries and to sketch new polygons forming the balconies. Bottom right: The final model after 20 minutes using the additional image information.

5.7.3 Further Applications

Photo-guided modeling

During our tests with a wide range of different data sets, we have made the observation that in some cases parts of the model to reconstruct are not only sparsely sampled, but are not depicted in the point cloud data at all. This may be caused by occluders (e.g., lots of trees and bushes), highly reflective materials (e.g., glassy or metallic facades, which lead to problems for both laser scanners and photogrammetric approaches) or the viewing angle from which the building has been captured. Our modeling tools have still proven to be capable of reconstructing such areas, if the user is provided with photos of the object – in case of photogrammetric data, these can even be reprojected onto the existing geometry. The example in Figure 5.12 shows a complex of connected buildings that are highly occluded by trees, resulting in a noisy and sparse point cloud in which

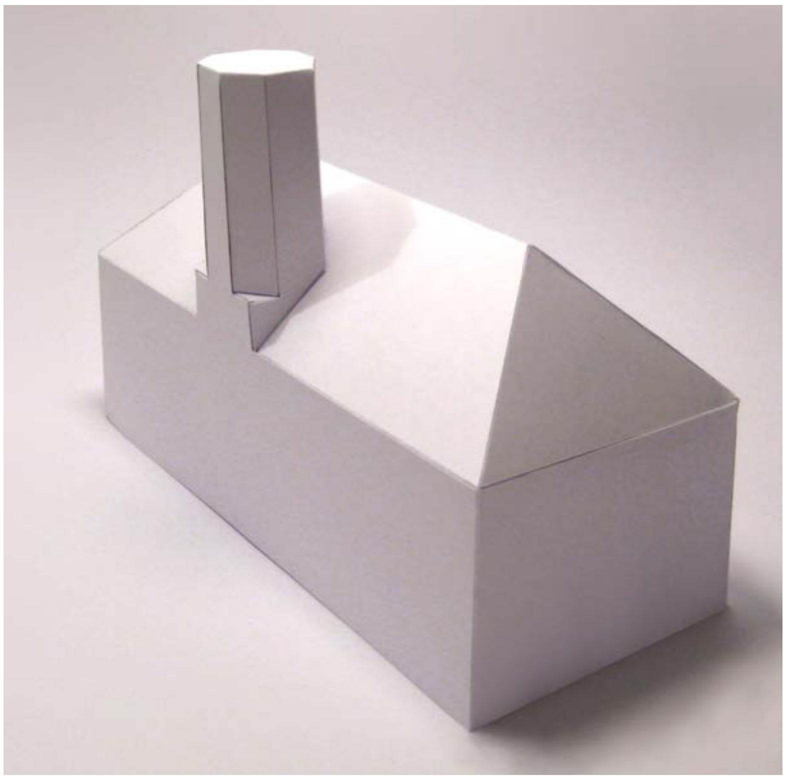

Figure 5.13: A paper model of *Town Hall*.

details like the balconies are not present. The image information reprojected on the basic shapes helps the user to modify the boundaries accordingly, and lets him or her accurately add any missing polygons as explained in Section 5.6.3. Please note that no reprojection of images was applied during the modeling process of the objects displayed in Figure 5.18.

Manufacturing

Precise reconstructions with low face count are of interest to applications beyond architecture, in particular to manufacturing. Simple production patterns are valuable, e.g., for upfolding planar cut patterns from paper or sheet metal. We shortly outline here how to implement the reverse operation to upfolding in our pipeline to generate production data.

Unfolding a polyhedron to a planar, connected shape without any self-intersections by only cutting along edges is a well surveyed research area (cf. [DO07]). Interestingly, it is still unknown if any convex polyhedron allows such an edge-unfolding, whereas it is known that there exist non-convex polyhedra where this is not possible. Basically, the solution space for a given mesh is given by all spanning trees of the mesh's face dual. By relaxing the constraints and requesting not a single but a small number of connected components, we determine a feasible spanning tree by heuristically searching

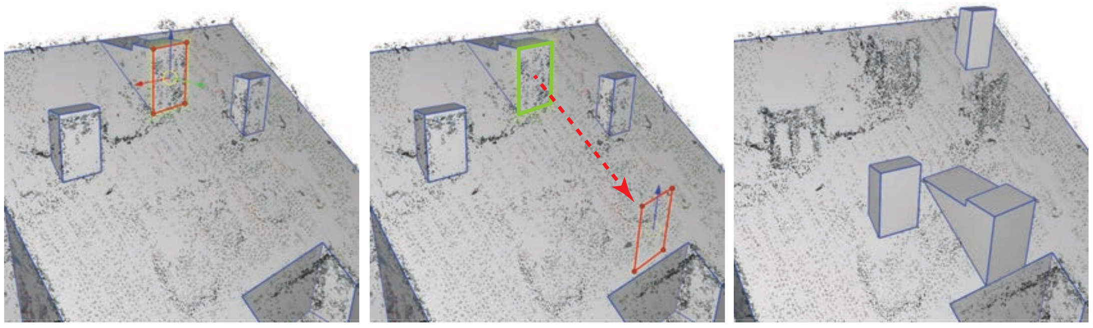

Figure 5.14: By design, the reconstructed models offer the generation of shape variations by exploiting the underlying adjacency graph.

the solution space. An example of a folded paper version of the *town hall* model is shown in Figure 5.13.

Advanced editing

Besides the modeling features introduced in Section 5.6, our models offer (by using the underlying adjacency graph) further editing possibilities to create different looks of reconstructed shapes: The user selects a single face of a chimney and applies an affine transformation to it. The connected component of the adjacency graph (containing the chimney's transformed face) undergoes the same affine transformation and the optimization is applied to reestablish a closed model (see Figure 5.14). While our main task is still reconstruction from point clouds, our pipeline leads to interesting ways for shape manipulation. Although having different objectives, the idea of optimization coupled editing has been extensively studied in the shape manipulation framework introduced by Gal et al. [GSMCO09].

5.7.4 Evaluation

To evaluate our method, we compared the visual quality of the models generated using our system and various other mesh reconstruction and decimation algorithms, performed a test to validate the accuracy of our results compared to using an existing interactive tool, and conducted a user study with non-expert users to show the ease of use of our method.

Comparison with meshing methods.

To evaluate the visual quality of our reconstructions, we applied our method and various other meshing techniques to the laser scan of the *Church of Lans le Villard* (Figure 5.15 top left). Figure 5.15 compares the different approaches: MeshLab's [CCR08] implementation of *Poisson Surface Reconstruction* [KBH06] (top right), Salman et al.'s feature-preserving mesh generation [SYM10] (middle left), the latter method followed by Graphite's [Gra10]

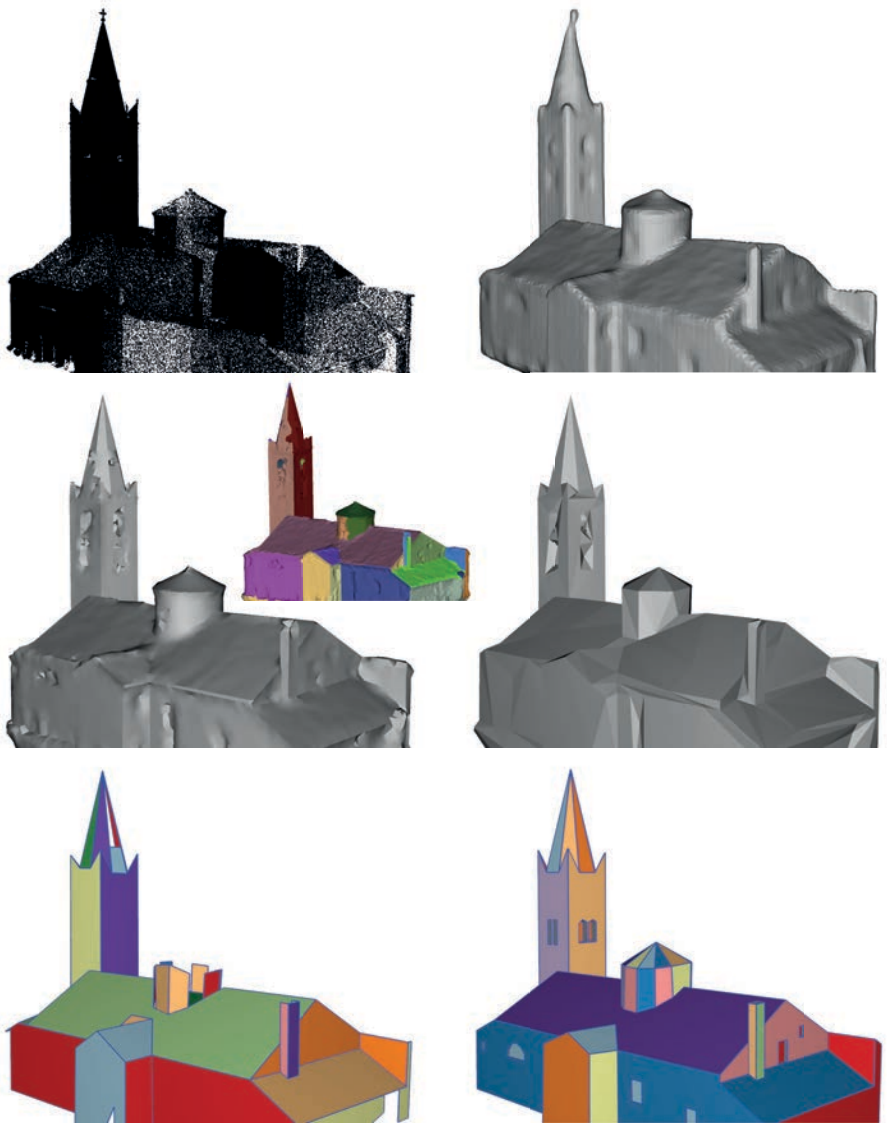

Figure 5.15: Top left: Input point cloud (provided courtesy of INPG by the AIM@SHAPE Shape Repository) downsampled to 10% of the original data set. Top right: Meshed Poisson implicit function [KBH06] (41k triangles). Middle left: Feature-preserving mesh [SYM10] provided courtesy of the authors (18k triangles). Middle left, small: Segmentation of the latter mesh [CSAD04]. Middle right: Mesh obtained by applying quadric-based simplification to the segments [GH97] (1k triangles). Bottom row: Initial automatic reconstruction of our method (43 polygons, 289 triangles used for rendering) and final refined model after additional 15 minutes of optimization-aided modeling (174 polygons, 109 of them for the windows, 799 triangles used for rendering).

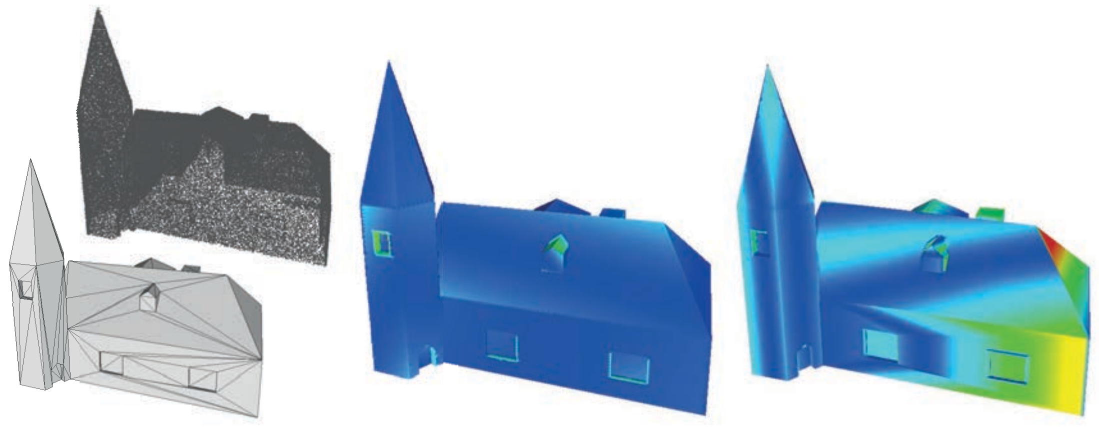

Figure 5.16: Point cloud with artificial noise (top left) sampled from a synthetic model (bottom left). Results from O-Snap (middle) and Pointools (right), colored according to approximation error, with blue meaning zero Hausdorff distance, and red high distance, respectively. Note that with O-Snap, the overall building structure is recreated very accurately. Errors mainly appear at sparsely sampled child elements (e.g., windows), especially at their side faces, which currently do not consider the point cloud at all.

implementation of geometry segmentation [CSAD04] (middle left, small image), and the same model with quadric-based mesh decimation [GH97] applied to each segment with fixed boundary, for which we used MeshLab [CCR08] again (middle right). The third row shows the results of our reconstruction and modeling pipeline.

Accuracy comparison

We used our system and a commercial point-based modeling tool, the Pointools plugin for SketchUp [Poi11], on a point cloud sampled from a synthetic house model. Noise was added to sample positions in the amount of 0.5% of the bounding box diagonal. Modeling was performed by a skilled artist, who was instructed to create the most accurate model possible in the two tools, both of which he had used before. There were no time constraints. After completion the artist reported to be confident having created perfectly accurate models in both tools, but also that he needed considerably more time and patience for modeling in Pointools. In order to quantify this feedback, we compared Hausdorff distances for each result to the original synthetic model (measured using the Metro tool [CRS96]), and modeling times (see Table 5.1). The results indicate that our method outperforms the commercial tool in terms of accuracy and modeling time. In addition to the Table 5.1, see also Figure 5.16 for a visualization of the approximation error.

Tool	Duration	Min	Mean	Max
O-Snap	6.2 minutes	0	0.000588	0.007794
Pointools	35.7 minutes	0	0.001433	0.009382

Table 5.1: Comparison of modeling times and approximation errors (color coded in Figure 5.16) relative to the model's bounding box diagonal.

Session	Data Set	Initial Reconstruction	Duration *Ø, (min-max)*
1	Town Hall	available	3.7 minutes (1-6)
2	Town Hall	no	6.8 minutes (5-9)
3	Old Church	available	13.1 minutes (6-20)

Table 5.2: Average modeling times of the reconstruction tasks carried out in the user study.

User study

In order to verify our tool's general usability for non-expert users, we performed an informal user study. All participants had never used the tool before and received the same ten minutes hands-on introduction. Each user had to complete three separate O-Snap sessions. Since not all candidates had a computer graphics or modeling background, we only gave the general directive to create a *good-looking model.* We stopped each session as soon as the user reached a closed model without major deficiencies. A time-stamped log file of all user interactions was generated for each session (see Figure 5.17).

Session 1 is based on the simple town hall model shown in Figure 5.18 (top row). Here the initial reconstruction (RANSAC, Sections 5.4 and 5.5) is almost perfect, and users only have to add a missing back wall and fine-tune some faces. An average user solves this task in three minutes. *Session 2* is equivalent to 1, but *without* the polygonalization step (Section 5.4). Users have to manually sketch all the polygons on the underlying segment planes (Section 5.4.1), and are aided by interactive optimization (Section 5.5), which takes seven minutes on average. We conclude that fine-tuning our automatic pipeline results is about twice as efficient than sketching all the polygons from scratch. On the other hand, the interactive optimization allows a novice user to create a model from scratch in still acceptable time. Finally, *Session 3* challenges users with a complex church model, shown in Figure 5.18 (second row). The initial model contains misaligned faces and some parts are missing, but all users were able to deal with the high geometric complexity and successfully create a closed model in only 13 minutes on average.

General observations Users who spend more than average time consistently try to model ever smaller details or extrapolate building parts not contained in the original data. All users are able to quickly recover from erroneous modeling actions using *undo*

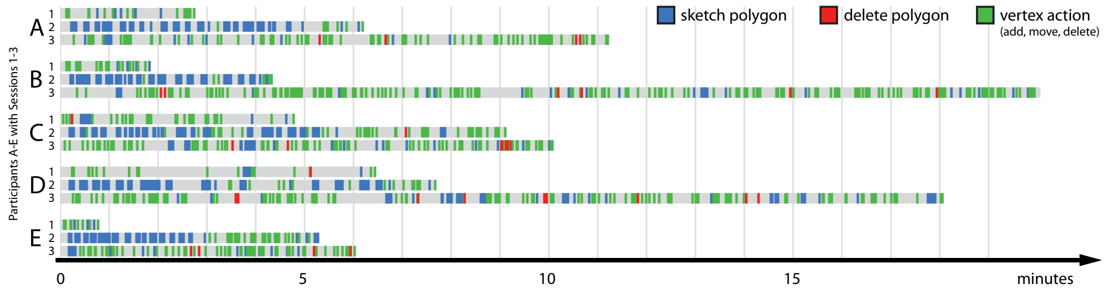

Figure 5.17: All modeling sessions of our informal user study. Each of the 15 horizontal bars represents the timeline of a single session. Each user interaction has been logged and timestamped. Different colors represent different kinds of interaction. Session 1 shows mostly vertex actions (move, add, delete) as results from the automatic pipeline are fine-tuned. Session 2 consists mostly of polygon sketching because no initial model is available. Session 3 is based on a complicated church data set. This takes more time, but all users are able to create a clean model. **Participants**: **A** has basic computer graphics skills. **B** and **E** are researchers in real-time rendering. **C** is a skilled user of commercial modeling software. **D** has absolutely no computer science and graphics background.

or by deleting incorrect shapes. No user was ever genuinely lost or stuck. An unexpected but useful observation is that users who manage to accidentally sketch a polygon inside a wrong plane, consistently try to flip this polygon over to the correct plane. Currently such an operation is not supported, but it seems to be expected intuitively, which is an inspiration for future work.

Our main conclusion is that non-expert users are perfectly capable of understanding and applying O-Snap's modeling tools after only ten minutes of basic introduction. Even participants without any prior CG and modeling experience are able to create shapes of buildings, aligned with the underlying point cloud data.

Comparison with Pointools We compared the effectiveness of our system to an existing commercial tool by instructing the user study participant with the most experience in 3D modeling to create models of *town hall* and *old church* using Pointools.A time limit of 30 minutes per scene was stipulated. The candidate was already familiar with the tool. While he succeeded in modeling the town hall in about 12 minutes (as compared to 5 minutes using O-Snap, see Figure 5.17, **C**), he had severe problems handling the more complex church model. Choosing appropriate points in the point cloud to construct the building's outline as well as to draw faces on top of the extruded outline consumed much of the available time. At the end of the 30 minutes time limit, he ended up with an incomplete model. Using O-Snap he was able to successfully complete the same task in about 10 minutes.

We conclude that a Pointools-like approach is very well suited for quickly creating simple axis-aligned models, but becomes tedious for more complex or incomplete (real-world) data sets. The recommended approach of ground plane-based extrusion of building outlines results in additional effort and inaccuracies, since many architectural elements cannot be captured by simple extrusion and have to be fixed manually.

In contrast, O-Snap's concept of sketching in 2D (on planes automatically fitted to the underlying point cloud) is more in line with an artist's workflow. It also strongly supports the comprehension of complex data sets by removing the effort required to extract geometric meaning from raw point data.

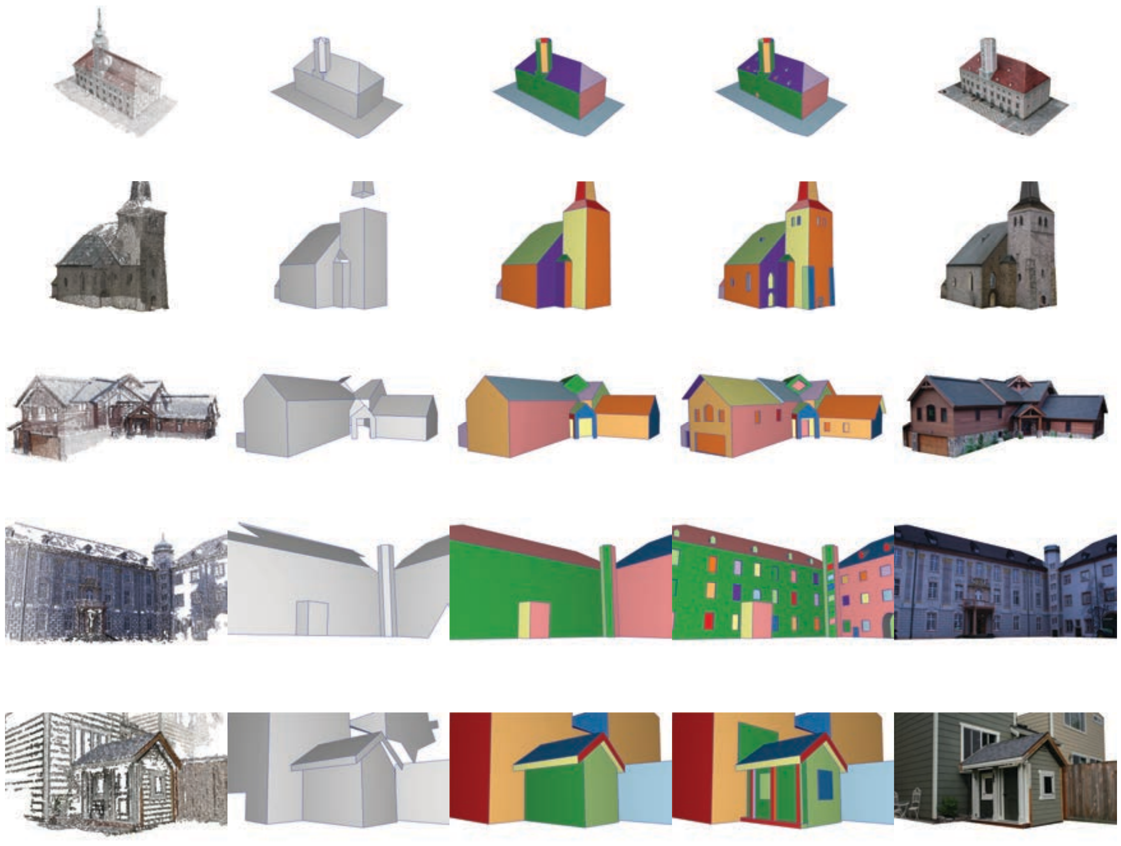

Figure 5.18: Results from five point cloud data sets (generated from photos) shown from top to bottom: town hall, old church (Photos courtesy of Rainer Brechtken), mountain house (Photos courtesy of Sinha et al. [SSS+08]), castle-P19 (Photos courtesy of Strecha et al. [SvHG+08]) and playhouse (Photos courtesy of Sinha et al. [SSS+08]). Left to right: input point cloud, initial automatic reconstruction, refined model, final model with advanced details added, final textured model (with the photos simply back-projected onto the model, more sophisticated methods for seamless texturing exist, e.g., Sinha et al.'s graph-cut optimization [SSS+08]).

5.8 Conclusion

We presented an interactive 3D modeling system that leverages techniques from mathematical optimization to provide a novel way of user-guided reconstruction and modeling of architectural scenes. The system first proposes an initial automatic reconstruction from an unstructured point cloud, by extracting candidate planes and estimating coarse polygons on these planes. Local feasible adjacency relations between polygons are automatically computed and enforced to align (snap) different parts of the model, while maintaining a fit to the input data. Besides these local relations, the system also favors orthogonality. In an interactive modeling phase, the model can be refined using coarse strokes on 2D planes. After each step, snapping reestablishes a watertight model where feasible.

5.8.1 Limitations and Future Work

Since many man-made objects, and especially buildings, consist of planar surfaces, we solely used planes in our reconstruction and modeling pipeline. In practice, this already allows handling a wide range of architectural styles by approximating curved surfaces in the scene by planes (see the cylindrical and conical parts of the model approximated by a number of planes in Figure 5.15 bottom right). However, our method is not limited to planes and can support other shapes by extending our matching definition and defining appropriate modeling spaces, which is an inspiration for future work.

In order to keep our matching and pruning definition simple, we allowed a few restrictions: (1) skew edges aren't matched, and (2) vertex-face matches aren't pruned. The edge-edge matching definition and the extended matching graph can be revised to overcome these restrictions, but we didn't observe any cases in our data sets where this would be necessary.

The robustness of the boundary extraction algorithm can suffer from non-uniformly sampled segments. Even though we have not observed this known limitation of alpha shapes in our data sets, one could use *conformal alpha shapes* [CGPZ05], which employ a local scale parameter (instead of the global α) to reconstruct non-uniformly sampled surfaces.

Currently, our system does not investigate repetitive structures (e.g., balconies, windows on a facade), but these structures can efficiently be modeled through our optimization-aided hierarchy operation. However, we plan to extend our system to analyze regular structures as proposed by Pauly et al. [PMW$^+$08].

Our novel modeling system differs significantly from other 3D modeling techniques through its interactive coupling with the optimization routines. Currently, we are performing more extensive user studies to learn more about expectations of O-Snap's users.

CHAPTER 6

Sketching 3D Buildings using Oriented Photos:

This chapter is based on the publication:

Michael Schwärzler, Lisa-Maria Kellner, Stefan Maierhofer, and Michael Wimmer. Sketch-based Guided Modeling of 3D Buildings from Oriented Photos. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D 2017)*, pages 9:1–9:8. ACM, February 2017.

The original paper was adapted in terms of formatting and type-setting to fit this template and to increase readability. The introduction was adjusted to fit the topic of this thesis, and the abstract was removed. Minor corrections, such as fixing typos or unclear wording, were applied. The original version is available at

https://dl.acm.org/citation.cfm?id=3023374.

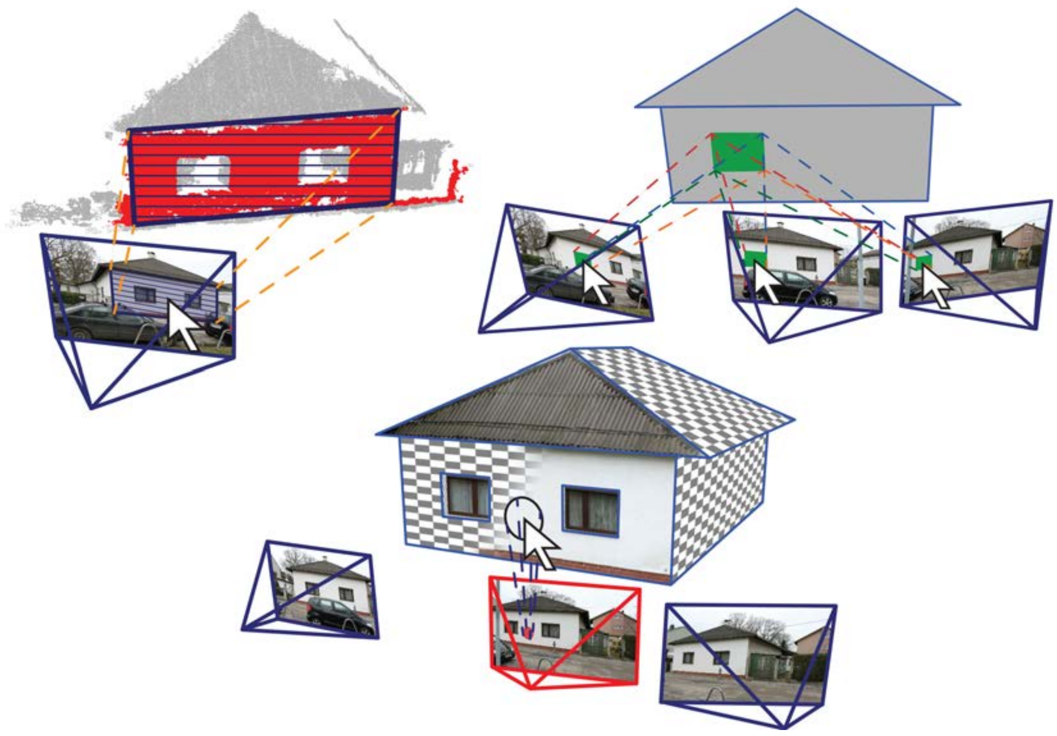

Figure 6.1: Modeling operations taking both oriented images and point cloud data into account. Top Left: Point cloud-supported single shot sketching, exploiting planar structures in the data. Top Right: Multi-view shot view sketching. Bottom: Texturing the generated polygons using an interactive brushing method.

6.1 Introduction

In the previous chapter, it has been shown that the combined use of geometric data together with extracted spatial relations can not only help to create CAD-ready 3D buildings with low polygonal detail while preserving planarity, but that such an approach can also be used to simplify and accelerate manual modeling steps. In this chapter, we go one step further and add image-based data to the reconstruction and modeling procedure, i.e., a real multimodal system is presented.

One problem that complicates urban reconstruction is that data sets are rarely complete: Laser scanners cannot be arbitrarily positioned, so that it is common that parts of the data are missing. Tachymetric point clouds are too sparse for reconstruction, and photogrammetric point clouds often have large holes caused by uniformly colored areas in them (see Section 6.3).

In this work, we propose an interactive modeling approach that remedies the problem of missing data by leveraging the available oriented 2D photos, which hold much more information than the point cloud or any reconstruction using only the point cloud: the user sketches the desired geometry directly on the image using simple 2D operations, both following the visual cues provided by a consistent image, which are much stronger than the features of a reconstructed point cloud, as well as supported by snapping to automatically extracted image edges. This allows recovering the missing structural information while integrating naturally into a simple sketch-based 2D workflow. From the user sketches, the 2D polygons are either directly projected to 3D space, or the user provides additional sketches in further views until a unique mapping to 3D space can be defined, depending on the available data quality of the underlying point cloud. This interactive process is supported and guided by providing suggestions for potential polygon candidates in neighboring images, as well as by giving visual feedback on the estimated accuracy for the projection to 3D space. This allows the user to make use of both point cloud and image data, while relying on an optimal, flexible workflow with minimal manual intervention. Our main contributions are:

- An intuitive 3D modeling approach for photogrammetric datasets, that relies on simple, sketch-based interactions on 2D photos based on snapping to edges in the image, and therefore even usable on mobile devices.
- A polygon-sketching method that obtains the required 3D plane automatically from the point cloud.
- A multi-view sketching method allowing polygons to be defined where no point cloud data is available, and that reduces user interaction by proposing suitable polygons in further views once an initial polygon has been sketched.
- A visual indicator guiding the user through the polygon sketching process by giving feedback on achieved accuracy.

6.2 Related Work

The field of urban reconstruction has gained a lot of scientific attention in the last years. A complete overview is out of scope for this work, and we refer the interested reader to the recent state-of-the-art report by Musialski et al. [MWA+13]. Instead, we put our focus on methods that incorporate *photos* in the geometric reconstruction pipeline, or that try to simplify 3D modeling by either *reducing the interaction dimensions* or *providing suggestions* to the user.

From a user-oriented perspective, our novel system is most closely related to the systems presented by Debevec et al. [DTM96] or Sinha et al. [SSS+08]: These interactive tools also rely on image-based modeling operations, and use photogrammetric data sets to calculate geometric correspondences in order to reconstruct 3D geometry. While operations like snapping to edges in images or multi-view texture generation have been integrated in these tools as well, neither of them exploits the availability of the underlying photogrammetric point cloud in order to simplify the sketching progress as in our method. Furthermore, we introduce an additional *guiding indicator* in the graphical user interface that operates as a feedback provider to give the user an easy-to-grasp preview on how much more modeling work is needed (see Section 6.5.3).

The derivation of polygonal meshes from point clouds has been intensively studied in recent years [KBSS01, BO05, KBH06, ACSTD07, SYM10]. Wang et al. [WFS$^+$15] use additional image data in their interactive tool in order to regularize the building by proposing a scaffold-like structure. Still, especially in the domain of urban reconstruction, the resulting 3D buildings differ significantly from typical models designed with CAD tools: While human users create building models consisting primarily of geometric primitives with exact intersections, meshed point clouds are inherently noisy and contain holes. Additionally, the absence of hierarchical relations makes geometric editing or semantic classification cumbersome.

To tackle these problems, Arikan et al. [ASF$^+$13] have proposed an interactive method that identifies basic planar shapes in a point cloud, on which initial coarse polygons are created. Holes between the polygons are automatically closed by an optimization step. In unclear cases, users can edit, fix or add polygons using simple 2D operations on the corresponding segmented plane. Unfortunately, this approach relies on point clouds that resemble nearly the whole surface. Especially in the case of photogrammetric data, often-occurring large holes cannot be accurately reconstructed. Reisner-Kollmann et al. [RKLS11] propose using image information for automatically filling holes in the surface. In our approach, we combine both approaches: planar surfaces in the point cloud are identified and used as a sketching plane for 2D modeling interaction – but by using photos as additional input in the 2D domain, sketched polygons can additionally snap to image edges, and polygons for which no point cloud data is available can be accurately reconstructed.

Our novel work is therefore a combination and extension of the before-mentioned interactive modeling tools, striving for simplicity in terms of modeling operations (2D image-based sketching and snapping) and exploiting structural information (planar point cloud segments, image edges) from all data sources available – while helping and guiding the user through the process and leaving all decisions to her artistic freedom.

6.3 Photogrammetric Data

As this work focuses on interactive modeling using photogrammetric data, we describe the properties and distinctive characteristics of this input type: A *Photogrammetric Network* is obtained from a set of overlapping photos of an object by using *Structure from Motion (SfM)* techniques, which gives the locations and orientations from which the photos were taken, as well as a 3D point cloud consisting of matching image features that have been reprojected to 3D space (see Figure 6.2). The point cloud can be further densified using algorithms proposed by Furukawa et al. [FP10, FCSS10]. Since such points have not been measured but were calculated using image features, photogrammetrically generated points may not be as dense and – more importantly – not as uniformly distributed as

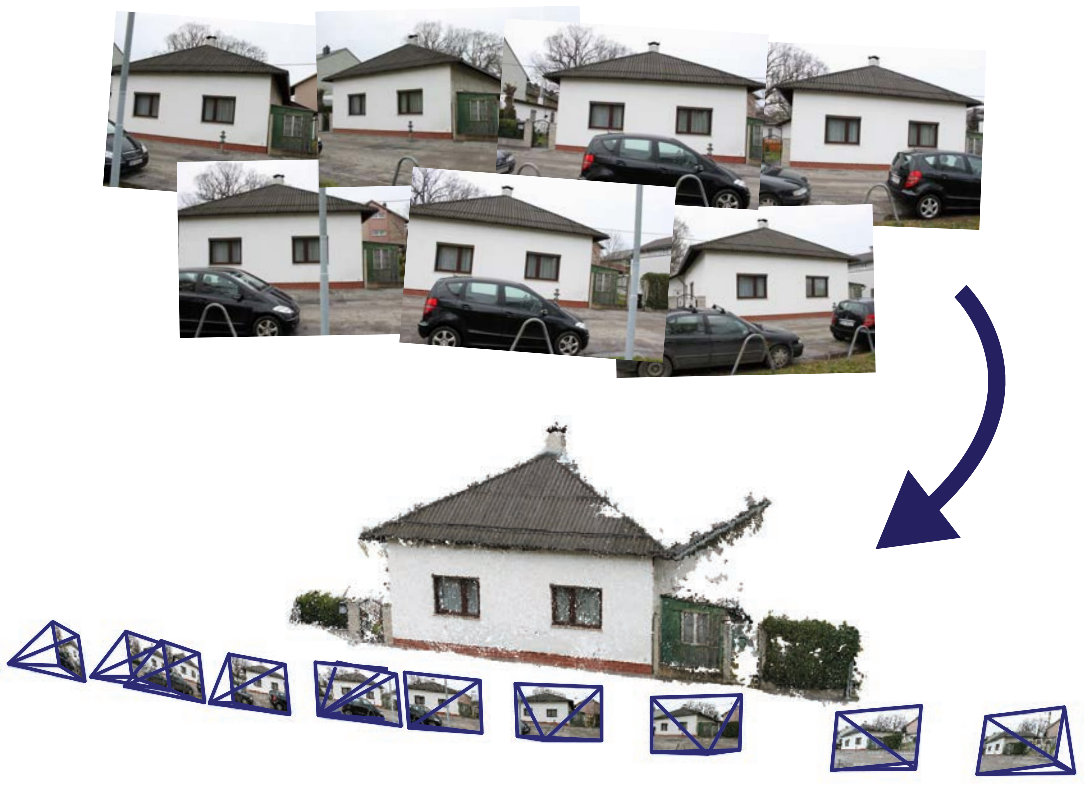

Figure 6.2: Photogrammetric Network (bottom), consisting of a 3D point cloud and photos (top), for which their relative positions and orientations have been computed. We refer to them as *shots*.

point clouds from laser scans, leading to more holes in the data. For example, it is difficult to extract features from completely flat, featureless walls. We therefore strive for compensating such missing information by defining polygons in multiple photos, see Section 6.4.1.

The oriented photos – we refer to them as *shots* – in the Photogrammetric Network are positioned around the point cloud. By having access to intrinsic and extrinsic camera parameters, transformations from the 2D image space to the 3D world space and vice versa can be achieved. In the case of our work, this is necessary for simple 2D editing and sketching steps and their according impact on the 3D world space, see Section 6.5.

Another advantage of the availability of shots is their use in further reconstruction steps, as for interactive line snapping or texture generation. Furthermore, the acquisition process can be done with a consumer-level photo camera and freely available SfM tools, making it a cheap and easy solution compared to other methods.

6.4 Definition of Polygons using Shots

The primary interaction and sketching target in our framework is a shot, selected from a photogrammetric network as described above. Sketching directly in a shot photo for the purpose of creating 3D geometry has two major advantages in terms of usability:

- The user immediately grasps the scene to reconstruct, as a photo is a very close approximation of what one perceives when looking at an object.
- The interaction is performed in a 2D environment. This not only makes the modeling tools less complex to handle, but also corresponds to human intuition, since humans are used to sketching or drawing on a flat sheet of paper since their early childhood.

While defining the approximate outline of a flat polygon in 2D space is relatively easy to achieve, the derivation of the corresponding representation in 3D space requires additional information: The *3D plane* on which the 2D outline has to be projected from the photo is completely unknown at first, but can be calculated by taking additional constraints into account. We therefore propose three methods to estimate this needed information in an intuitive way with the least possible user effort, and without having to leave the 2D sketching domain. We describe these three methods in the following subsections.

6.4.1 Multi-Shot Sketching

One method to obtain the 3D positions for the vertices of a sketched polygon in 2D image space is to define it not only in one, but in multiple photos. Since the orientation of the shots is known in 3D space, each pixel on the image plane can be used to define a ray from the focal point of the camera through the pixel position in world space. If this is done for a polygon vertex in multiple images, the intersection point of the corresponding rays defines its 3D position (see Figure 6.1, top right). This is repeated for all vertices, and the unknown plane can then be estimated using the least-squares method.

We implemented these ray intersections using the triangulation method based on homogeneous direct linear transformation (DLT) as described by Hartley and Zisserman [HZ04], resulting in a least-squares optimal solution. This approach is just one possible solution to this intersection problem. We opted for it is robust and easily generalizes to triangulation in more than 2 views. We exploit this during our guided sketching feedback, where we encourage the user to define the polygon in more than 2 shots (see Section 6.5.3).

6.4.2 Point Cloud Supported Single-Shot Sketching

Even though the multi-view approach described above works well, it is still an overhead for the user to have to sketch in two shots. To allow sketching in just a single shot, we exploit the available point cloud data: Similar to Arikan et al. [ASF$^+$13], we segment the point cloud into planar segments using the RANSAC algorithm by Schnabel et al. [SWK07]. After an initial polygon has been sketched, we try to find on which planar segment it was most likely intended to be drawn. For this, we transform the points of each planar segment from 3D world space into the 2D image space of the shot used for sketching, and test which points of each segment lie inside the polygon. Note that in our current implementation, we perform this test for all segments, which could be easily optimized by performing a culling step (e.g., by using the bounding boxes of the segments).

To determine how well a polygon fits a planar segment, we use the number of points lying *inside* the 2D polygon as well as the *uniformity* of the distribution of these points, i.e., whether the projected point cloud segment has "holes" in it, and express this in a heuristic $h \in [0, 1]$. The uniformity is estimated by rasterizing all points in the segment as splats over the polygon and then determining a fill ratio r, where 1 means fully filled and 0 not filled at all. The heuristic is then computed as

$$h = \frac{mr}{n},$$

where n is the total number of points of the segment and m is the number of points of the segment inside the polygon.

The splat size q used for splatting is based on the average point distance, where d_i is the distance between point i and its nearest neighbor:

$$q = \frac{1}{n}\sum_{i=1}^{n} d_i$$

If there is at least one segment that passes the (adjustable) acceptance threshold, we choose the one with the highest result as the potential candidate, and inform the user about the outcome (see Section 6.5.3). If the user decides to make use of it, the polygon is projected onto the plane that has been fit to the point cloud segment (see Figure 6.1, top left). Otherwise, the user continues sketching the polygon in further views, and the multi-view shot Sketching algorithm is applied. Nevertheless, the initially found candidate segment can still be helpful: if the normal of the polygon calculated using the multi-view method differs only 10 degrees from the segment plane normal, the polygon is adjusted to it accordingly.

6.4.3 Sketching Using the Plane of Existing Polygons

It is often obvious that some elements lie on the same plane in 3D space. This is especially the case for elements like windows, doors or balconies on a facade. We therefore allow the user to simply define the polygon of an existing element as the 3D sketching plane for the next polygon, and can therefore reproject the 2D outline to 3D space immediately.

6.5 Guided Polygon Creation

In this section, we describe how we integrated the described polygon-sketching methods in an interactive modeling workflow. All interactive concepts described in the following only guide and support the user. Despite all the suggestions of our system, we assume that the user "knows best" what her intentions are. Every suggestion and guidance step in our system can therefore also be safely ignored by the user.

6.5.1 Shot View Navigation

As described above, all sketching operations are performed within 2D photos for reasons of simplicity. However, it is of utter importance that the user also implicitly knows about the current view location in the 3D world, so that the spatial context can be used to sketch polygons in multiple shots without mixing them up.

During sketching, users are presented a 2D view of the current photo. Nevertheless, since the corresponding shot also includes 3D information, we allow the opacity value to be changed arbitrarily, such that 3D content (e.g., already modeled polygons) can be made visible. Furthermore, switching between shots is designed to help the user retain his spatial orientation: Instead of switching the displayed photo immediately, the camera performs a flying animation to show where the user is "virtually going".

6.5.2 Sketching and Snapping in Shot View

We facilitate sketching in a selected shot by a snapping feature which lets sketched lines snap to edges detected in the image. Thus, the user only needs to create a rough sketch. For this, we use an implementation of the Line Segment Detector described in the work of von Gioi et al. [vGJMR10] to find edges in the underlying image. The outline of the initial polygon is compared to the set of lines in the image. Two lines are considered matching if they are nearly parallel and spatially close. If no match is found, the sketched edge will be used as-is. Figure 6.3 shows this polygon-snapping workflow.

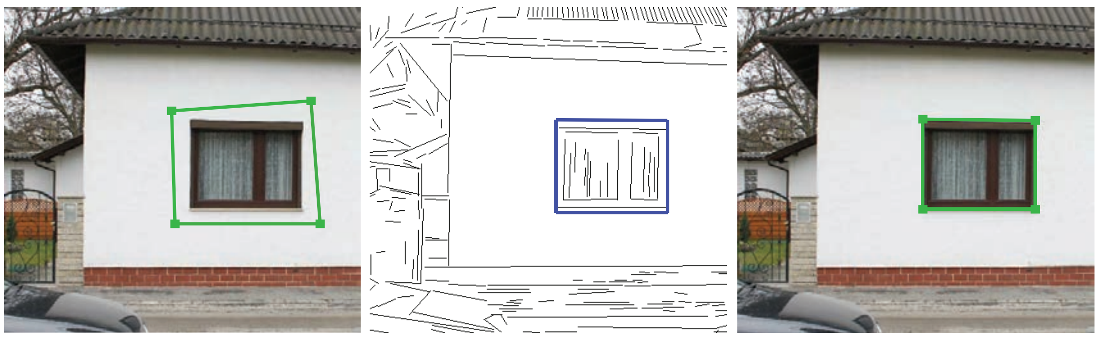

Figure 6.3: Polygon Snapping: sketched 2D polygon on an image (left), extracted image lines with matchings in blue (middle), snapped polygon (right).

If the user opts for using the multi-view shot modeling mode, the workflow requires the same polygon to be available in different images. Instead of having to sketch the corresponding polygon again, our proposed system tries to minimize this effort: As soon as the user switches to the next shot (defined by a nearest spatial neighborhood heuristic), the initial polygon is projected into the new image and repositioned to "fit the same sketched object". For example, if the polygon snapped to window edges in the initial image, the projected polygon in the neighbor image also tries to snap to the same window edges.

This is achieved by computing normalized color histograms at the edges of the initial polygon in the underlying photo. Then, the histograms are compared with histograms of edges found in the target image to find matching lines. The best match is used as starting point for the polygon in the other shot. Finally, adjacent edges are added step-by-step.

Our system makes no assumptions about specific structures or spatial arrangement of the underlying geometry to be reconstructed. We only assume a planar polygon viewed from different vantage points under arbitrary affine transformations. We found that histogram-based matching of polygon edges works reliably in this general case. Of course, this does not exclude the possibility to combine our approach with existing specialized methods for constrained scenarios, like highly repetitive structures [MWW12] or hierarchical block-like arrangements [XFT$^+$08].

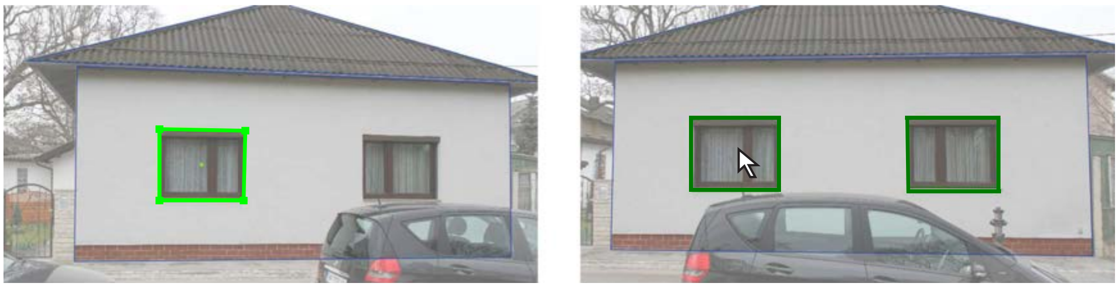
Figure 6.4: Multi-view sketching with suggestions. Left: The window is sketched in the first view and snapped to the image edges. In this case, no suitable planar point cloud segment is found, which is why the sketched 2D polygon cannot be projected to 3D space. Right: In a neighboring view, a search for similar polygons is performed in image space by using the histograms of the polygon edges of the source image. Two candidate polygons that resemble the same type of window are found. They are suggested to the user, and the left one is selected by a click.

Especially in the case of buildings, it becomes obvious why an interactive approach with minimized input, which mostly consists of deciding on proposed suggestions, is important: Architectural objects often consist of extremely similar and repetitive patterns, and an initially sketched window can be found multiple times on the next photo by our algorithm. We therefore display all found polygon candidates as suggestions in the target image. Since the user is able to retain a global spatial overview more easily, the correct window can then be picked with a single click. This process is repeated in each image the user navigates to. In images where the correct polygon cannot be found (e.g., due to occlusions), the wrong suggestions can be completely skipped.

6.5.3 Visual Guidance Feedback

As stated before, we want to minimize the needed user interaction while keeping the possibility to influence any design decision at the user level. It is therefore important for the user to get feedback on whether the polygon should be sketched in further views, or if enough information on the needed 3D plane is already available to compute the world space position of the object.

During each polygon creation process, the user gets continuous feedback via our novel visual guidance interface to realize this: In the user interface, a state bar appears as soon as the initial polygon is sketched. The state can vary between *red* (not enough information), *yellow* (the system can suggest a 3D polygon, but it may be inaccurate or ambiguous) and *green* (an accurate polygon can be provided, and no other plane candidates can interfere). See Figure 6.5 for a visualization of the guidance element in the user interface.

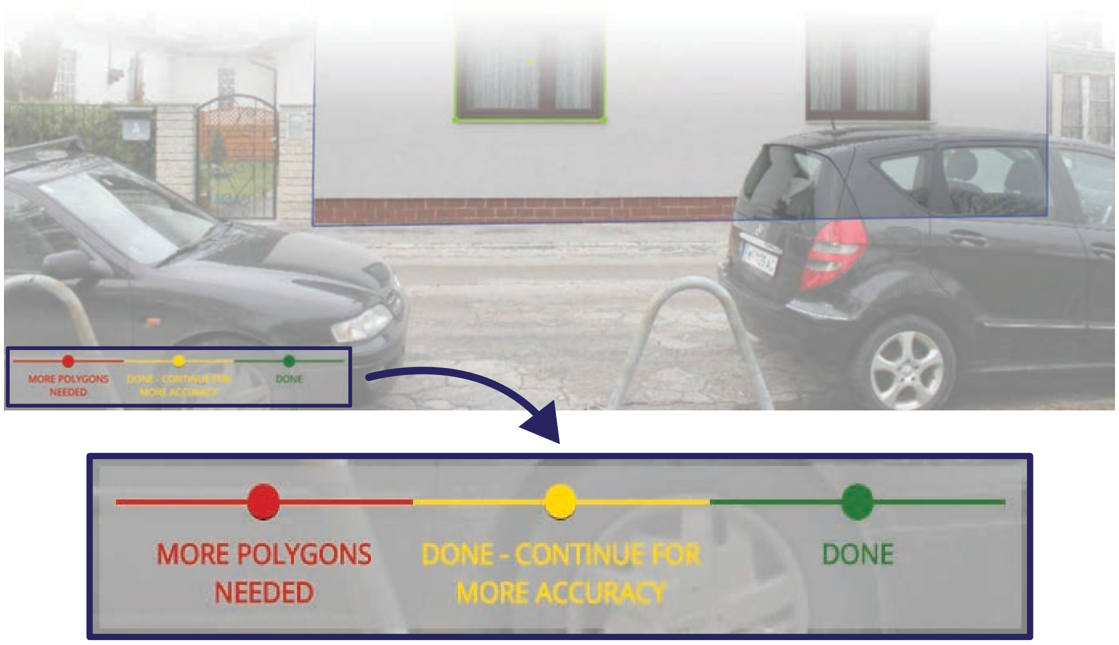

Figure 6.5: The visual guidance interface shows whether enough polygons have already been sketched to compute a 3D polygon, or if the user should continue sketching.

Concretely, we use the red state whenever an initial polygon has been sketched, and no plane to project the 2D outline onto is available. This is the case when no neighboring 3D polygon has been selected (see Section 6.4.3) and no fitting planar point cloud segment can be found (i.e., the metric returns no value above a certain user-definable threshold for all point cloud segments, see Section 6.4.2). The yellow state is used when, after sketching the initial polygon, either two or more potential planar point cloud segment candidates that are of equal quality are available, or, when using multi-view shot sketching mode, the polygon has only been defined in two shots yet (which may be inaccurate, see Section 6.4.1). The green state is shown as soon as the polygon is defined in at least three different views, or when a single significant plane to project the 2D outline to is available (single-shot sketching).

6.6 Additional Photo-Based Modeling

Apart from sketching the initial polygons, we have integrated further possibilities that demonstrate the combined use of photo data, point clouds and geometry in a single environment.

6.6.1 Model Refinement

All typical 3D polygon-modeling tasks in our system – like adding and removing vertices, translation or scaling – can be performed via the shot view. This is especially true for existing polygons that have not been modeled in this particular view, but can be reprojected and edited in the corresponding photo anyway. Following the same principle, polygons snap to edges, and can be aligned according to the image content. Model refinement through the shot view is more intuitive and accurate for a user than, for instance, fitting a polygon to the point cloud.

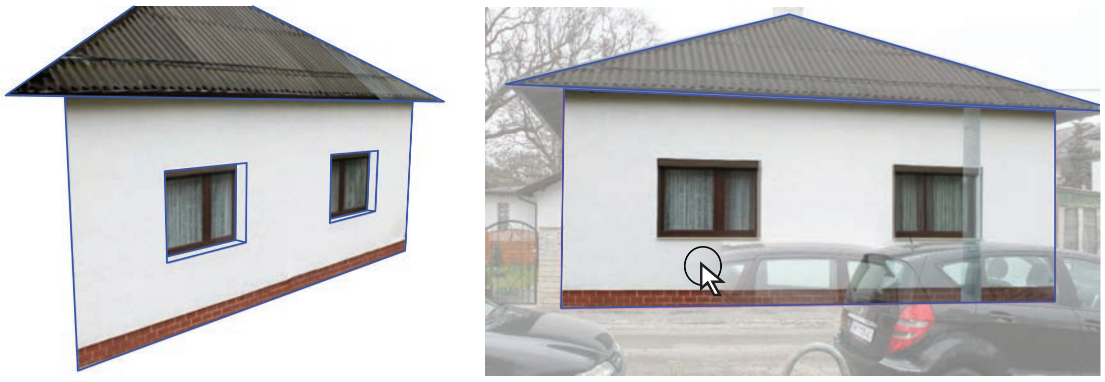
Figure 6.6: Left: By defining hierarchical relations, holes and side faces are automatically extracted. Right: Interactive removal of occluding objects from the texture by overlaying the photo semi-transparently using the shot view.

In addition to shot-based sketching operations described in previous sections, our framework supports standard polygon editing features known from other 3D modeling packages. For this, the camera can be moved around freely in 3D space. Switching to this mode is especially useful when a surface needs to be closed due to missing photos. We also included optimization-based snapping to close small gaps between polygons as proposed by Arikan et al. [ASF+13], which takes into account further constraints like parallelism or orthogonality of edges – an often-needed requirement for CAD-ready models. Moreover, we allow the definition of hierarchical relations: This makes it easily possible to define "holes" for windows and doors in the facades (the corresponding side faces are added automatically), like in Figure 6.6, left.

6.6.2 Texture Brushing

Shots can be used to generate textures by reprojecting associated photos onto the polygons. In order to obtain a suitable texture, we use a technique proposed by Musialski et al. [MLS+10], where a single polygon is textured from multiple photos, each pixel colored according to the best-fitting shot. While the initial source of each pixel is selected automatically based on angle and distance, arbitrary parts of the texture can be "repainted" with the photo of a user-selected shot in order to remove occluders or artifacts. Figure 6.7 shows textures with different coloring according to shots.

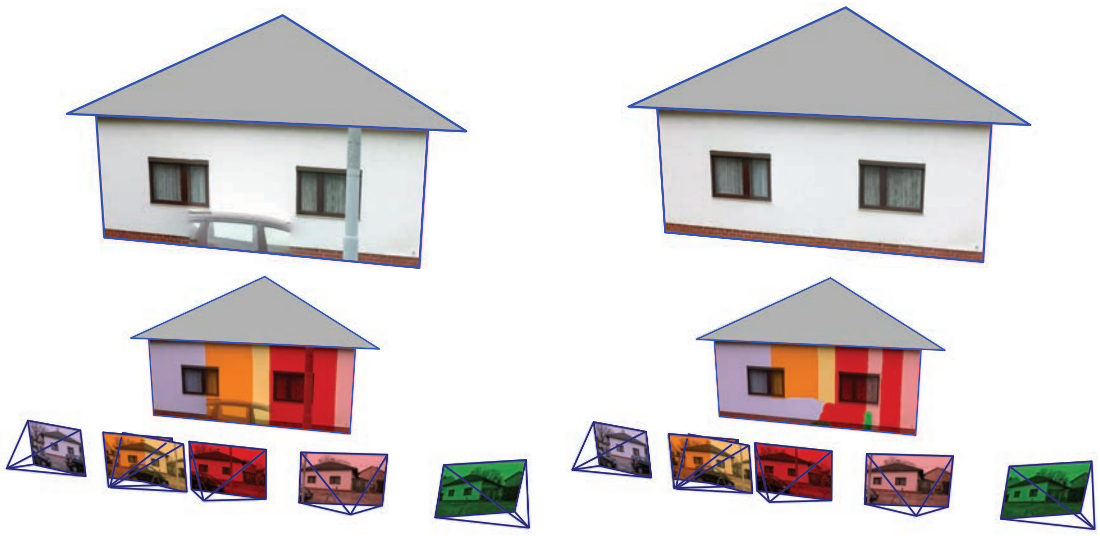

Figure 6.7: Top left: The initial texture containing occluders. Bottom Left: The associated shots, visualized using a false color mask. Top right: The cleaned texture after interactive brushing. Bottom right: The correspondingly modified mask.

We extended the original method, in which users could only brush a polygon in 3D view, to be also used via the shot view, in which transparency can be adjusted interactively. This way, users simultaneously have access to the current state of the textured polygon in the 3D world, and the 2D photo content. In shot view brushing mode, the brush paints over the texture with the content of the shot image the user is actually looking at, so that one can easily find proper image parts for specific texture positions (see Figure 6.6, right). Switching between shots and leaving the shot view is possible at any time as described in Section 6.5.1.

6.6.3 Touch-based Interaction

As navigating between the individual shots and sketching on photos – the main interaction tasks needed in our approach – do not require any pixel-accurate clicks or complicated keyboard commands, we also investigated the possibility to use our system on a touch-based interface. As demonstrated in Figure 6.8, the concept of roughly sketching on photos is well-suited for defining the planar shapes of a building. Right now, touch-based modeling is limited to sketching in the shots, as adding arbitrary polygons freely (see Section 6.6.1) has not been implemented in a touch-based manner in our prototype yet.

Although the touch input is less precise than mouse and keyboard, this is often alleviated by the snapping mechanism, and we see the possibility of using a touch-based interface as an interesting first step towards the development of specialized mobile apps: A complete on-site 3D building reconstruction (taking photos, computing orientations in the cloud, and modeling the final textured model) could be made possible using just a small hand-held device.

6.7 Implementation

Our novel modeling and reconstruction framework has mainly been implemented using an existing rendering framework based on the .NET framework and OpenGL. The visual guidance feedback element has been realized using a web-based overlay that was created using HTML5 and the D3.js toolkit [BOH11]. The interactive texture-brushing method makes use of a Poisson solver implemented in OpenCL. For polygon snapping, we were given access to the original implementation of Arikan et al. [ASF+13].

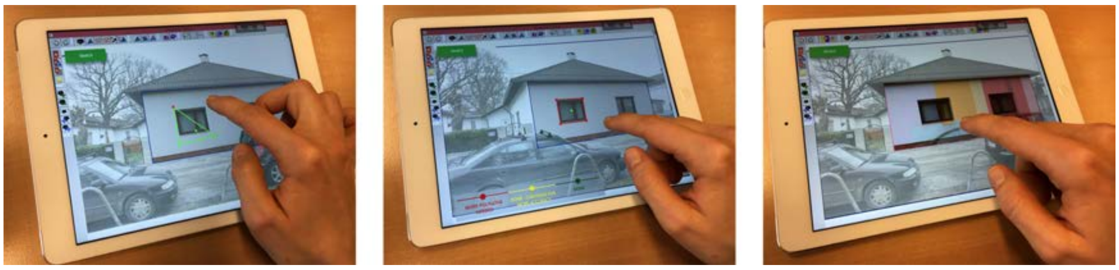

Figure 6.8: Evaluation of our proposed system on a touch-based device. Left: Coarse sketching of the polygon in the initial view. Middle: The polygon has been snapped to image edges and been reprojected to a neighboring image in the multi view sketching mode. The visual guidance feedback indicates that enough views have been used. Right: Interactive brushing.

The tool currently supports photogrammetric data sets generated with either the PMVS/CMVS toolkit [CMV10] or with the commercially available software Agisoft Photoscan [Agi18]. During the import process, the shot neighborhood relations as described in Section 6.5.1 are computed, and the image edges for snapping are extracted in preprocessing steps.

We believe that our proposed workflows and interaction methods can be integrated into existing 3D modeling packages, but it has to be carefully evaluated whether the interactions described in this work conflict with the standards established there.

6.8 Evaluation and Results

All described interaction methods are fully interactive. Real-time frame rates allow fluent work on consumer-level hardware. We have evaluated our novel framework by gathering feedback from five users we let try to reconstruct several buildings from photogrammetric datasets. All had some background knowledge in the field of 3D modeling or interactive editing in 3D scenarios. We intentionally chose datasets where the photogrammetric reconstruction produced point clouds that did not fully represent the building structure (i.e., some planar surfaces like white walls or reflective windows that are clearly visible in the photos are not depicted in the point cloud). Thus, both single shot and multi shot sketching could be applied.

While all users grasped the modeling interactions after a short introduction and managed to fulfill the requested task – to reconstruct and texture the given building model as fast and accurately as possible – within 10 to 20 minutes, their feedback differed greatly based on their background and experiences:

- Two experts from the field of surveying were immediately happy with the idea of defining polygonal shapes using single points from a shot, as it resembles their typical workflow when taking geodetic measurements. They were eager to see such techniques applied to their currently used GIS software. One expert was rather skeptical regarding the achievable accuracy of the photogrammetric approach, and requested a prior registration of the point cloud to geodetic data.
- One user working as content creator for the gaming industry first intended to coarsely sketch the ground plan of the building, extrude the walls from it, and wanted to align the walls to fit the photos. While this seems to be a feasible approach at first, it would require the rotation of the defined planes – an operation we have not allowed in our prototype (but would definitely be worth to evaluate). After adapting to our proposed workflow, it was pointed out that the shot-based creation of occluder-free building textures from multiple photos was a lot faster than manual stitching that is usually done.
- We asked a user who was familiar with the O-Snap system [ASF$^+$13] for creating buildings solely from point clouds to use our system. Since the optimization-based

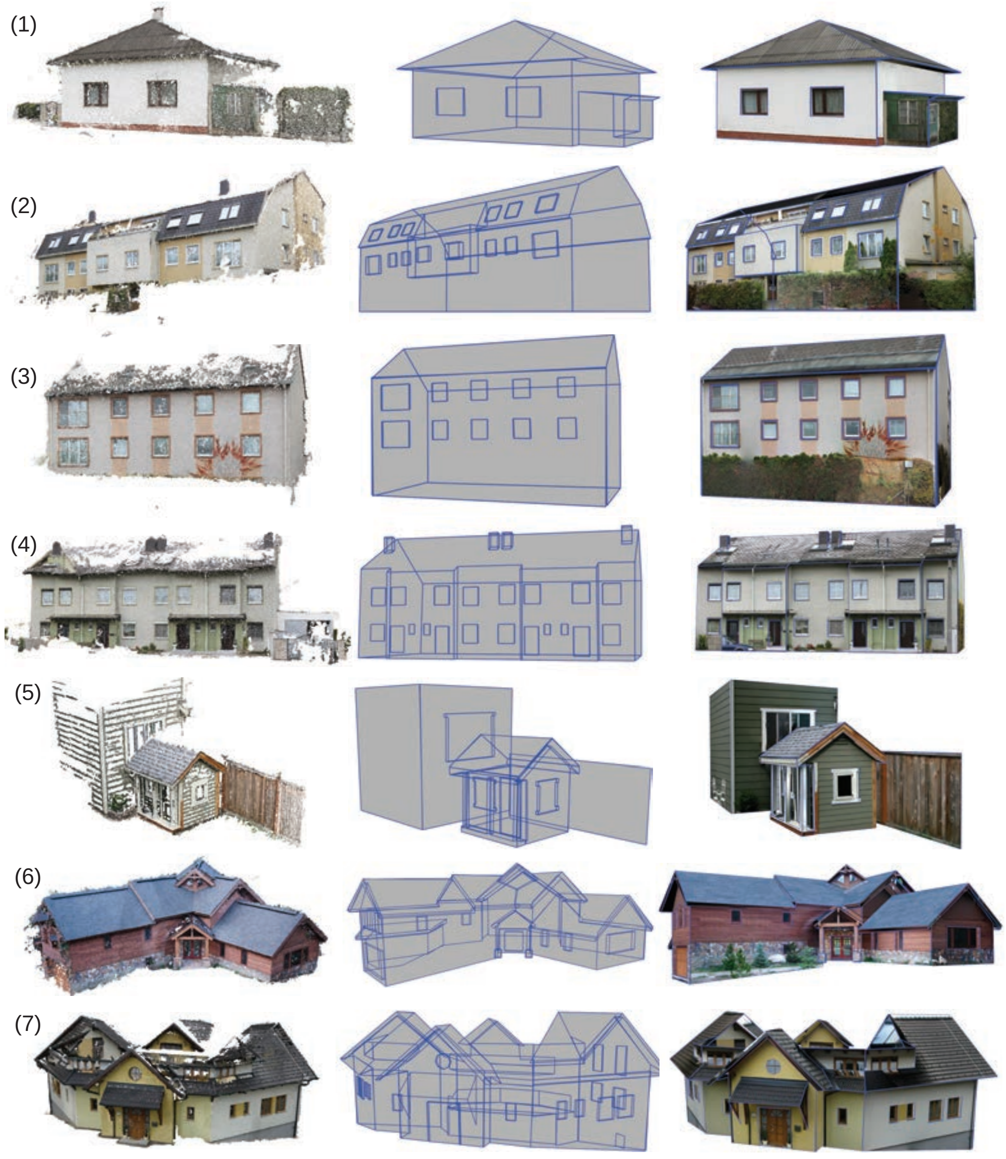

Figure 6.9: Textured 3D building models generated with our approach. Left: Photogrammetric point cloud. Middle: Geometric reconstruction including hierarchical definitions. Right: Final model with textures generated from multiple photos using the interactive occluder removal. Parts that are not depicted in the point cloud could be accurately reconstructed using the photos. The backsides of the houses 1-4 were modeled freely, as they were not accessible for the photographer. Buildings 6 and 7 were modeled using data sets with complete photo coverage from all sides, but required significantly more modeling time due to their complexity. For building 7, not enough photos were taken to model the roof completely. Modeling time including texture generation in minutes, from top to bottom: 5, 15, 10, 20, 15, 40, 45.

snapping is also integrated in our prototype, the user tried to make use of it as often as possible. This sometimes failed when there were only very few planes sketched in the beginning of the modeling process, as the O-Snap algorithm tried to close holes when the basic structure was not defined yet, overriding the results from the image-based edge snapping. As soon as the main planes of a building had been modeled, the two algorithms complemented each other very well, though. We therefore removed the accessibility of the geometric optimization tool during shot view sketching, so that it is now only available as a post-processing option.

- In order to verify the feasibility of the touch-based version, we asked a developer of mobile VR/AR apps to evaluate our prototype implementation. Even though he had several remarks on how to improve the usability (especially concerning the used gestures), he was convinced that the proposed modeling concept could be applied in a dedicated mobile app.

As can be seen in Figure 6.9, the targeted goal of creating low-polygonal, textured, CAD-ready 3D buildings in just a few minutes could be reached: The average modeling times for the buildings lie between five and fifteen minutes – including the generation of textures for the polygons. All users used the feedback from the guidance indicator during the polygon sketching progress, and even pointed us towards interaction workflows in tools they frequently use, where similar guidance approaches could be useful.

It is important to notice that especially the side parts of the buildings, where no complete point cloud was available due to the limited access for the photographer to the area, could be accurately reconstructed using our image-based approach. The front facades, where the point cloud is usually quite dense, could be successfully modeled with the single shot method described in Section 6.4.2. Once a single window of a certain type was modeled, all the others on the same facade could be created using the same plane and the edge-based snapping feature within seconds.

6.8.1 Limitations

Even though we have shown that using both photos and point clouds from photogrammetric data sets in an interactive workflow makes it possible to reconstruct more areas accurately, our approach still suffers from the fact that objects that are hidden, occluded or only visible in a single photo require manual, inaccurate modeling steps. Furthermore, we are (similar to the methods proposed by Arikan et al. [ASF^{+}13] and Sinha et al. [SSS^{+}08]) limited to the reconstruction of planar surfaces. Even though curved surfaces can be approximated using multiple polygons, the handling of such primitives is more challenging than it is for planar shapes.

6.9 Conclusion & Future Work

We have demonstrated how to combine interactive techniques from both image-based and point cloud-based methods to reconstruct CAD-ready 3D models of buildings within a few minutes. 3D planes, on which sketched 2D polygons are reprojected, can not only be computed from multiple views, but also from planar segments detected in the corresponding point cloud. Image-based snapping features and suggestions further improve the sketching workflow. Our novel method is a natural extension of these related techniques, and does not interfere with their concepts, but improves them. By introducing an intuitive *visual guidance indicator*, users can take shortcuts during the image-based modeling steps, while being aware of the quality impact this has.

In the future, we plan to further fine-tune the user-oriented interaction methods – we are especially interested in bringing the touch-based interaction to a level that a complete 3D building modeling process is possible on mobile devices. Together with the camera capabilities of modern smartphones and server-based photogrammetric computation, a complete reconstruction workflow on a single device seems to be within reach.

We will also investigate if we can use learning algorithms to replace currently user-defined parameters and thresholds, as they may vary depending on input data. Furthermore, we were inspired from the user feedback we got from the expert users, and plan to enhance the modeling workflow by also integrating additional data sources like geodetic measurements or ground plans.

CHAPTER 7

Conclusion & Outlook

The scientific contribution of this thesis emerges primarily from the work presented in the chapters 3 to 6. Even though in each of these chapters specific, individual scientific problems are solved, they all contribute to advance the state-of-the-art in the field of Multimodal Urban Reconstruction and Modeling.

The proposed algorithms and methods are currently being used and improved in several scientific research projects and applications, where they will not only act as the base for consecutive research questions, but hopefully also contribute to the creation of novel multimodal workflows.

7.1 Contribution of the Presented Methods

Based on the methodology that has been applied in order to reach the goals set by the stated problems, the scientific contributions of the individual methods are summarized as follows:

Scientific contribution of the methods presented in this thesis:

Fast Accurate Sampling of Area Lights (Chapter 3): With the proposed technique, it is possible to render physically accurate soft shadows that in most cases outperform the regular light sampling method with a fixed sampling rate, since only the samples which contribute to the visual quality are computed and evaluated. By reprojecting the corresponding shadow maps to the camera's point of view and comparing them using an occlusion query, the decision whether another sampling point is needed in-between two neighboring ones is made. The time needed for these checks is often more than compensated by the reduced number of shadow maps which have to be calculated, leading to interactive or even real-time frame rates for generating physically accurate soft shadows.

Reusing Soft Shadows in Consecutive Frames (Chapter 4): The scientific contribution of this work is a novel method to accelerate the computationally expensive PCSS algorithm by exploiting temporal coherence techniques: Soft shadow intensities with varying penumbra sizes are stored in a history buffer and are potentially reused in consecutive frames. To account for soft shadows cast by moving objects, the shadow-mapping algorithm was extended by a so-called movement map – a light-weight 8-bit buffer storing the location of moving objects in light space. The algorithm is easy to integrate into an existing rendering framework, and can be robustly used for all kinds of different scenes. The achievable performance gain (between 250% in static scenes and 130% in fully dynamic scenes) comes at the cost of memory consumption, though: Apart from the 8-bit buffer for the movement map, two more 32 bit 2-channel screen-size buffers have to be allocated for the history buffer.

Interactive Polygon Snapping for 3D Building Reconstruction (Chapter 5): The scientific contribution of this work is an interactive system that leverages techniques from mathematical optimization to provide a novel way of user-guided reconstruction and modeling of architectural scenes. After an initial automatic reconstruction from an unstructured point cloud, candidate planes are extracted, and coarse polygons are estimated on these planes. Adjacency relations between polygons are automatically computed and enforced to align (snap) different parts of the model, while maintaining a fit to the input data. In an interactive modeling phase, the model can be refined using coarse strokes on the 2D planes that have been identified in the point cloud. Missing holes (that occur due to erroneous or missing data) can be filled, and hierarchical relations can be defined, leading to automatically extracted holes and side polygons in the building. Further advanced modeling steps, such as affine transformation of polygonal groups building entities, or the integration of photos for texturing, are possible. After each manual modeling step, the automatic snapping reestablishes a watertight model where feasible, allowing an interactive reconstruction of complex 3D building in a CAD-ready way within a few minutes.

Sketching 3D Buildings using Oriented Photos (Chapter 6): This chapter contributes to the scientific field of multimodal urban reconstruction by demonstrating how to combine interactive techniques from both image-based and point cloud-based methods to reconstruct CAD-ready 3D models of buildings within a few minutes. For this, we propose an intuitive 3D modeling approach for photogrammetric data sets, which relies on simple, sketch-based interactions on 2D photos based on snapping to edges in the image, and is therefore even usable on mobile devices. As the simplest interaction, a polygon-sketching method obtains the required 3D plane automatically from the point cloud in order to reproject the

2D polygon to 3D space. A multi-view sketching method allows polygons to be defined where no such point cloud data is available: It reduces user interaction by proposing suitable polygons in further views once an initial polygon has been sketched. The whole polygon sketching process is guided by a visual indicator by giving feedback on the achieved accuracy.

7.2 Impact on the Field

Through these individual scientific contributions presented above, the general problems for the field of *Multimodal Urban Reconstruction and Modeling* and the corresponding goals defined in Section 1.2 were tackled and partly solved. Of course, this area remains an active research field, as the ultimate goal of providing a common environment for the reconstruction of urban environments using arbitrary (city-related) data has by far not been reached yet. Nevertheless, the presented publications contribute the following aspects to move closer to this target:

- By treating **different kinds of input data in a common reconstruction and modeling environment**, it has been demonstrated how such interactive, multimodal approaches can be successfully applied for heterogeneous data. Apart from using point cloud data and photos, user-based knowledge information and spatial relations are applied to create **textured 3D models within a short time range**.
- The results have further demonstrated how multimodality helps to **handle** and – most importantly – **reduce the vast amounts of input data** by generating low-polygonal, structured, **CAD-ready 3D models**. This compactness, combined with the extracted or generated meta data, is a key factor for an instant **reusability** in the destined target domain – may it be another 3D modeling or BIM tool, 3D games, or the application in simulations.
- It has further been shown how a maximum level of **accuracy** – one of the key aspects for follow-up applications in which reconstructions are needed – can be maintained **while improving other aspects** such as computational speed (in the case of physically accurate soft shadows) or reducing the amount of data (in the case of point-based reconstruction).
- Being able to track the achievable accuracy, and raising awareness for potentially erroneous data, are key aspects to allow the user to use her or his knowledge to **interactively account for problems** (such as holes in the data caused by occlusions, reflective materials, or too sparse sampling). The exploitation of the **multimodal data helps to create guidance and suggestions-based tools**, making it possible to “repair” errors with a few clicks.

- Apart from accounting for erroneous data, also the general modeling aspects benefit from multimodality. By **reducing the dimension** of the interactions from 3D to 2D, by using easy-to-grasp **photographs as sketching canvas**, by treating **different data sources in the same spatial context**, or by providing **feedback** on the currently achieved level of accuracy – the simplicity of the specialized user interfaces makes the **handling of the novel tools a lot easier** than in previous approaches.

- In the presented approaches, a high level in terms of **visual quality** and **realism** has been reached. This has been achieved by automatic and manual removal of artifacts in both geometry and textures, and by properly simulating direct light distribution.

- All this has been targeted to achieve the **best possible user experience**: Real-time performance of the algorithms, interactive optimization after each modelling step, immediate feedback, and a tight coupling between manual and automatic modeling steps allow keeping the user in the loop without disturbing waiting periods.

7.3 Future Work

Based on the successful steps towards the creation of an integrated multimodal reconstruction environment presented in this thesis, it has recently become possible tackle the next challenge: the interdisciplinary handling of aspects concerning both, reconstruction and light transport, in a common project (see Section 7.3.1). The following individual research problems can be seen as direct follow-up challenges based on the findings presented in the Chapters 3 to 6:

Lighting-aware reconstruction: Lighting conditions — from both sunlight as well as from artificial light sources — play an important aspect during the acquisition process, as they have a major impact on the quality of the acquired data (e.g., artifacts such as glare, reflex, and shadows) and the perception of the scene. Current reconstruction algorithms (reaching from better photogrammetric pipelines up to the generation of illumination-free textures for 3D buildings) can be improved by generating test data using a light simulation framework, where any artifact can be artificially simulated and tagged for learning purposes.

Semantic-driven smart modelling using information retrieved from photos and machine learning: As a further extension to the systems proposed in the Chapters 5 and 6, computer vision methods can be used to gain information on the visible semantic entities. This information can be used to assign tags to polygons or regions, derive hierarchical structures, create textures and geometries for occluded regions, or even to generate 3D modeling suggestions (e.g., connect a wall with a roof by specialized helper geometry) and semantic-aware modeling tools (based on the edited type of entity).

Data on the acceptance and rejection of the suggestions is collected and used to improve the system automatically.

Suggestion-based, smart modelling techniques for lighting design: In large scenes, it is oftentimes very challenging and tiresome for lighting designers to place light sources according to given standards or customer wishes. Furthermore, the 3D geometry of the objects in a scene is very expensive to model. The use and enhancement of existing, suggestion-based smart modeling techniques for the creation and modification of 3D geometry enables working with multimodal, distributed data: In GIS or BIM systems, plenty of semantic information is available that can be potentially be used to derive the placement of various objects in outdoor scenes (trees, lanterns, block geometry of buildings, etc.), including an initial setup of light sources.

Real-time shadow algorithms for interaction in point clouds: In extremely large point cloud data sets, typical interactions as for example lasso selection and partial deletion can hardly be performed immediately – but against the obvious assumption, the bottleneck is not the handling of out-of-core data, but updating the colors of the selected or deleted points on the GPU (i.e., the time it takes to visualize the changed data is significantly longer than the actual computation of this point subset). This could be overcome by exploiting real-time GPU shadow algorithms, which are able to compute and visualize all scene parts lying in shadow within a fraction of a second: By seeing the drawn lasso as an occluder lying on the near plane of the viewer camera, and the focal point as a point light source, the shadow umbra (i.e., the selection volume) can be estimated quickly, and used to visualize the selected points without a costly GPU-CPU transfer.

Apart from these examples, we strive for an extension of our research focus towards areas that might trigger further interdisciplinary aspects in *Multimodal Urban Reconstruction*, and hope that they will contribute to solving (or at least simplifying) today's challenges.

7.3.1 SHARC - Smart, Hyper-Accurate Reconstruction for Geodesy

In January 2017, the first project that combines the two research fields of light transport and urban reconstruction has been started at the *VRVis Research Center*: *rmDATA GmbH*[1], an IT service provider for surveying and geoinformation, *Zumtobel Lighting*[2] and the infrastructure department of the Austrian railways (*ÖBB -Infrastruktur AG*[3]) joined forces in the collaborative applied research project *SHARC*[4]. The overall goal in the project is to develop tools and methods to handle, manage, manipulate and assess varying survey and light planning data sources in a common environment (see Figure 7.1).

[1]http://www.rmdata.at
[2]http://www.zumtobel.com
[3]http://infrastruktur.oebb.at/
[4]https://www.vrvis.at/research/projects/sharc/

The prototypes and results developed in *HILITE*, *VAMOS*, and *REPLICATE* build the foundation for it.

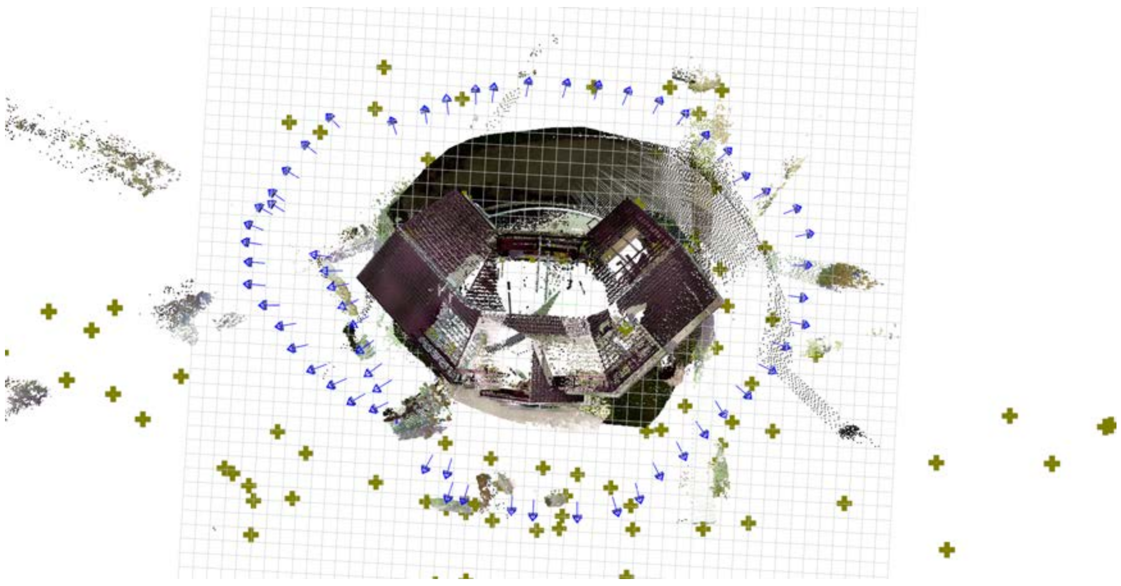

Figure 7.1: Screenshot of early first results of the *SHARC* project: Three kinds of point data – laser scans from multiple positions, photogrammetry and tachymetry – have been perfectly registered and aligned in a common coordinate system. This allows to exploit the individual strengths in a common context, making new interaction methods and improvements possible (e.g., using photos for coloring laser scans, or using tachymetry to detect and repair inaccuracies in photogrammetric data sets, etc.)

The complexity involved in processing and manipulating multimodal data in a common context has already reached a level that is impossible to tackle with traditional approaches. We therefore strive for the development of a novel system that allows for dealing with extremely large amounts of heterogeneous, distributed, and temporarily evolving geodetic data in combination with simulated light data in an integrated, dynamic environment. Concrete research steps are the use of BIM data for indoor lighting based on the building layout or room function (e.g., placement of emergency exit lights in floors, energy-efficient light setups that account daylight, etc.), reverse lighting design (i.e. influence of the desired light distribution in room on the internal parameters of the luminaire), and perceptual considerations (physiological aspects of the lighting conditions in the reconstructed buildings and scenarios).

The project is again financed by the COMET funding scheme. It will last for 4 years until the end of 2020, and is then planned to be continued for another four years until 2024. Currently, between 6 and 8 researchers are actively participating in this project.

List of Figures

List of Tables

Bibliography

[AAM03] Ulf Assarsson and Tomas Akenine-Möller. A Geometry-based Soft Shadow Volume Algorithm using Graphics Hardware. *ACM Trans. Graph.*, 22(3):511–520, 2003.

[ACSTD07] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based Variational Reconstruction of Unoriented Point Sets. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 39–48, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[ADM$^+$08] Thomas Annen, Zhao Dong, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. Real-time, All-Frequency Shadows in Dynamic Scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)*, 27(3):34:1–34:8, 2008.

[Agi18] Agisoft. PhotoScan. http://www.agisoft.com/, 2018. Accessed: 2018-06-11.

[AHL$^+$06] Lionel Atty, Nicolas Holzschuch, Marc Lapierre, Jean-Marc Hasenfratz, Chuck Hansen, and François Sillion. Soft Shadow Maps: Efficient Sampling of Light Source Visibility. *Computer Graphics Forum*, 25(4), 2006.

[AMB$^+$07] Thomas Annen, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. Convolution Shadow Maps. In *Rendering Techniques 2007: Eurographics Symposium on Rendering*, pages 51–60, Grenoble, France, 2007. Eurographics Association.

[AMS$^+$08] Thomas Annen, Tom Mertens, Hans-Peter Seidel, Eddy Flerackers, and Jan Kautz. Exponential Shadow Maps. In *GI '08: Proceedings of Graphics Interface 2008*, pages 155–161, Toronto, Ont., Canada, 2008. Canadian Information Processing Society.

[AR00] A. C. Atkinson and Marco Riani. *Robust Diagnostic Regression Analysis*. Springer-Verlag, 2000.

[ARHM00] Maneesh Agrawala, Ravi Ramamoorthi, Alan Heirich, and Laurent Moll. Efficient Image-based Methods for Rendering Soft Shadows. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 375–384, 2000.

[ASF$^+$13] Murat Arikan, Michael Schwärzler, Simon Flöry, Michael Wimmer, and Stefan Maierhofer. O-Snap: Optimization-Based Snapping for Modeling Architecture. *ACM Transactions on Graphics*, 32:6:1–6:15, January 2013.

[ASGCO10] Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. ℓ_1-Sparse Reconstruction of Sharp Point Set Surfaces. *ACM Trans. Graph.*, 29:135:1–135:12, November 2010.

[ASK06] B. Aszódi and L. Szirmay-Kalos. Real-Time Soft Shadows with Shadow Accumulation. In *Eurographics 2006 Short Presentations*, pages 53–56. Eurographics Association, 2006.

[BFGL09] Yang Baoguang, Jieqing Feng, Gael Guennebaud, and Xinguo Liu. Packet-Based Hierarchal Soft Shadow Mapping. *Computer Graphics Forum*, 28(4):1121–1130, 2009.

[BMH98] Dirk Bartz, Michael Meißner, and Tobias Hüttner. Extending Graphics Hardware for Occlusion Queries in OpenGL. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, HWWS '98, pages 97–ff., New York, NY, USA, 1998. ACM.

[BO05] Jean-Daniel Boissonnat and Steve Oudot. Provably Good Sampling and Meshing of Surfaces. *Graph. Models*, 67:405–451, September 2005.

[BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 Data-driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[BPGK06] Mario Botsch, Mark Pauly, Markus Gross, and Leif Kobbelt. PriMo: Coupled Prisms for Intuitive Surface Modeling. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 11–20, 2006.

[CC08] J. Chen and B.Q. Chen. Architectural Modeling from Sparsely Scanned Range Data. *IJCV*, 78(2-3):223–236, 2008.

[CCR08] Paolo Cignoni, Massimiliano Corsini, and Guido Ranzuglia. MeshLab: an Open-Source 3D Mesh Processing System. *ERCIM News*, (73):45–46, April 2008.

[CGPZ05] F. Cazals, J. Giesen, M. Pauly, and A. Zomorodian. Conformal Alpha Shapes. *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics*, 0:55–61, 2005.

[CMV10] Yasutaka Furukawa CMVS. Clustering Views for Multi-view Stereo (CMVS). http://www.di.ens.fr/cmvs/, 2010. Accessed: 2018-06-11.

[Cor14] Daniel Cornel. Analysis of Forced Random Sampling. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, October 2014.

[Cro77] Franklin C. Crow. Shadow Algorithms for Computer Graphics. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, volume 11, pages 242–248. ACM Press, July 1977.

[CRS96] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: Measuring Error on Simplified Surfaces. Technical report, Paris, France, 1996.

[CSAD04] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational Shape Approximation. *ACM Trans. Graph.*, 23:905–914, August 2004.

[CSLW17] Daniel Cornel, Hiroyuki Sakai, Christian Luksch, and Michael Wimmer. Forced Random Sampling: Fast Generation of Importance-guided Blue-noise Samples. *The Visual Computer*, 33(6):833–843, 2017.

[Dav11] Timothy A. Davis. Algorithm 915, SuiteSparseQR: Multifrontal Multi-threaded Rank-Revealing Sparse QR Factorization. *ACM Transactions on Mathematical Software*, 38(1), 2011.

[DL06] William Donnelly and Andrew Lauritzen. Variance Shadow Maps. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, I3D '06, pages 161–165, New York, NY, USA, 2006. ACM Press.

[DO07] E.D. Demaine and J. O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra.* Cambridge University Press, 2007.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-based Approach. In *Proceedings of SIGGRAPH 96*, pages 11–20, 1996.

[DWS$^+$88] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The Triangle Processor and Normal Vector Shader: a VLSI System for High Performance Graphics. *SIGGRAPH Comput. Graph.*, 22(4):21–30, 1988.

[EM94] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional Alpha Shapes. *ACM Trans. Graph.*, 13:43–72, January 1994.

[ESAW11] Elmar Eisemann, Michael Schwarz, Ulf Assarsson, and Michael Wimmer. *Real-Time Shadows.* A.K. Peters, 2011.

[FBP06] Vincent Forest, Loïc Barthe, and Mathias Paulin. Realistic Soft Shadows by Penumbra-wedges Blending. In *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH '06, pages 39–46, New York, NY, USA, 2006. ACM.

[FCOS05] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. Robust Moving Least-Squares Fitting with Sharp Features. *ACM Trans. Graph.*, 24:544–552, 2005.

[FCSS09] Y Furukawa, B Curless, S M Seitz, and R Szeliski. Manhattan-world Stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (2009)*, pages 1422–1429, 2009.

[FCSS10] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Towards Internet-scale Multi-view Stereo. In *In: Proceedings of IEEE CVPR*, 2010.

[Fer05] Randima Fernando. Percentage-Closer Soft Shadows. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 35, New York, NY, USA, 2005. ACM Press.

[FP10] Yasutaka Furukawa and Jean Ponce. Accurate, Dense, and Robust Multi-view Stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1362–1376, August 2010.

[GBP06] Gael Guennebaud, Loïc Barthe, and Mathias Paulin. Real-Time Soft Shadow Mapping by Backprojection. In *Eurographics Symposium on Rendering (EGSR 2006), Nicosia, Cyprus*, pages 227–234. Eurographics Association, 2006.

[GBP07] Gael Guennebaud, Loïc Barthe, and Mathias Paulin. High-Quality Adaptive Soft Shadow Mapping. *Computer Graphics Forum*, 26(3):525–534, 2007.

[GH97] Michael Garland and Paul S. Heckbert. Surface Simplification using Quadric Error Metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[Gra10] Graphite. http://alice.loria.fr/index.php/software.html, June 2010. Accessed: 2018-06-11.

[GSMCO09] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. iWIRES: An Analyze-and-Edit Approach to Shape Manipulation. *ACM Transactions on Graphics (Siggraph)*, 28(3):#33, 1–10, 2009.

[HH97] Paul S. Heckbert and Michael Herf. Simulating Soft Shadows with Graphics Hardware. Technical Report CMU-CS-97-104, CS Dept., Carnegie Mellon U., Jan. 1997.

[HLHS03] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François Sillion. A Survey of Real-Time Soft Shadows Algorithms. In *Eurographics 2003 State of the Art Reports*. Eurographics Association, 2003.

[HSM+14] Georg Haaser, Harald Steinlechner, Michael May, Michael Schwärzler, Stefan Maierhofer, and Robert F. Tobler. CoSMo: Intent-based Composition of Shader Modules. In *Proceedings of International Conference on Computer Graphics Theory and Applications (Grapp 2014)*, 2014.

[HSM+15] Georg Haaser, Harald Steinlechner, Michael May, Michael Schwärzler, Stefan Maierhofer, and Robert Tobler. *Semantic Composition of Language-Integrated Shaders*, pages 45–61. Springer International Publishing, Cham, 2015.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*, chapter 12.2, page 312f. Cambridge University Press, second edition, 2004.

[Jol02] I. T. Jolliffe. *Principal Component Analysis.* Springer, New York, 2nd edition, 2002.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[KBS+18] Katharina Krösl, Dominik Bauer, Michael Schwärzler, Henry Fuchs, Michael Wimmer, and Georg Suter. A VR-based User Study on the Effects of Vision Impairments on Recognition Distances of Escape-route Signs in Buildings. *The Visual Computer*, 34(6):911–923, Jun 2018.

[KBSS01] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature Sensitive Surface Extraction from Volume Data. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 57–66, New York, NY, USA, 2001. ACM.

[Kel99] C. T. Kelley. *Iterative Methods for Optimization.* Society for Industrial and Applied Mathematics, Philadephia, PA, 1999.

[KFB08] Myers Kevin, Randima Fernando, and Louis Bavoil. Integrating Realistic Soft Shadows into Your Game Engine. Technical report, NVIDIA Corporation, 02 2008.

[KFC+08] M. Kilian, S. Flöry, Z. Chen, N. J. Mitra, A. Sheffer, and H. Pottmann. Curved Folding. *ACM Transactions on Graphics*, 27(3):#75, 1–9, 2008.

[KLSW17] Katharina Krösl, Christian Luksch, Michael Schwärzler, and Michael Wimmer. LiteMaker: Interactive Luminaire Development using Progressive Photon Tracing and Multi-Resolution Upsampling. In *Vision, Modeling and Visualization 2017*. The Eurographics Association, 2017.

[Krö16] Katharina Krösl. Interactive, Progressive Photon Tracing using a Multi-Resolution Image-Filtering Approach. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, March 2016.

[LTH+13] Christian Luksch, Robert F. Tobler, Ralf Habel, Michael Schwärzler, and Michael Wimmer. Fast Light-Map Computation with Virtual Polygon Lights. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2013*, pages 87–94. ACM, March 2013.

[LTM+14] Christian Luksch, Robert F. Tobler, Thomas Mühlbacher, Michael Schwärzler, and Michael Wimmer. Real-Time Rendering of Glossy Materials with Regular Sampling. *The Visual Computer*, 30(6-8):717–727, June 2014.

[LWC+11] Yangyan Li, Xiaokun Wu, Yiorgos Chrysanthou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. GlobFit: Consistently Fitting Primitives by Discovering Global Relations. *ACM Transactions on Graphics*, 30(4):to appear, 2011.

[MÏ2] Thomas Mühlbacher. Real-Time Rendering of Measured Materials. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, November 2012.

[May15] Michael May. Design and Implementation of a Shader Infrastructure and Abstraction Layer. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, September 2015.

[MLS+10] Przemyslaw Musialski, Christian Luksch, Michael Schwärzler, Matthias Buchetics, Stefan Maierhofer, and Werner Purgathofer. Interactive Multi-View Façade Image Editing. In *Vision, Modeling and Visualization 2010*, pages 131–138, November 2010.

[MWA+13] Przemyslaw Musialski, Peter Wonka, Daniel G. Aliaga, Michael Wimmer, Luc van Gool, and Werner Purgathofer. A Survey of Urban Reconstruction. *Computer Graphics Forum*, 32(6):146–177, September 2013.

[MWW12] Przemyslaw Musialski, Michael Wimmer, and Peter Wonka. Interactive Coherence-Based Façade Modeling. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2012)*, 31(2):661–670, May 2012.

[NSL+07] Diego Nehab, Pedro V. Sander, Jason Lawrence, Natalya Tatarchuk, and John R. Isidoro. Accelerating Real-Time Shading with Reverse Reprojection Caching. In *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, pages 25–35. Eurographics Association, 2007.

[NSZ$^+$10] Liangliang Nan, Andrei Sharf, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. SmartBoxes for Interactive Urban Reconstruction. *ACM Trans. Graph.*, 29:93:1–93:10, 2010.

[Ope07] OpenGL Working Group. OpenGL Occlusion Query Extension. http://www.opengl.org/registry/specs/ARB/occlusion_query.txt, April 2007. Accessed: 2018-06-11.

[PHYH06] Helmut Pottmann, Qi-Xing Huang, Yong-Liang Yang, and Shi-Min Hu. Geometry and Convergence Analysis of Algorithms for Registration of 3D Shapes. *Int. J. Comput. Vision*, 67:277–296, 2006.

[PMW$^+$08] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. Guibas. Discovering Structural Regularity in 3D Geometry. *ACM Transactions on Graphics*, 27(3):#43, 1–11, 2008.

[Poi11] Pointools Ltd. Pointools plugin for sketchup. http://www.pointools.com/pointools-plug-in-for-sketchup.php, January 2011.

[RKLS11] Irene Reisner-Kollmann, Christian Luksch, and Michael Schwärzler. Reconstructing Buildings as Textured Low Poly Meshes from Point Clouds and Images. In *Eurographics 2011 - Short Papers*, pages 17–20, April 2011.

[RL87] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection.* John Wiley & Sons, Inc., New York, NY, USA, 1987.

[RLD$^+$12] Tim Reiner, Sylvain Lefebvre, Lorenz Diener, Ismael García, Bruno Jobard, and Carsten Dachsbacher. A Runtime Cache for Interactive Procedural Modeling. *Computers & Graphics*, 36(5):366 – 375, 2012.

[RSC87] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering Antialiased Shadows with Depth Maps. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 283–291. ACM Press, 1987.

[SaLY$^+$08a] Pitchaya Sitthi-amorn, Jason Lawrence, Lei Yang, Pedro V. Sander, and Diego Nehab. An Improved Shading Cache for Modern GPUs. In *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, pages 95–101, Aire-la-Ville, Switzerland, 2008. Eurographics Association.

[SaLY$^+$08b] Pitchaya Sitthi-amorn, Jason Lawrence, Lei Yang, Pedro V. Sander, Diego Nehab, and Jiahe Xi. Automated Reprojection-Based Pixel Shader Optimization. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2008)*, 27(5):127, 2008.

[SAPP05] Jean-François St-Amour, Eric Paquette, and Pierre Poulin. Soft Shadows from Extended Light Sources with Penumbra Deep Shadow Maps. In *Graphics Interface 2005*, pages 105–112, May 2005.

[SB03] Konrad Schindler and Joachim Bauer. A Model-Based Method For Building Reconstruction. In *Proceedings of the First IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, pages 74–82, 2003.

[Sch05] Daniel Scherzer. Shadow Mapping of Large Environments. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 8 2005.

[SDK09] Ruwen Schnabel, Patrick Degener, and Reinhard Klein. Completion and Reconstruction with Primitive Shapes. *Computer Graphics Forum (Proc. of Eurographics)*, 28(2):503–512, March 2009.

[SEA08] Erik Sintorn, Elmar Eisemann, and Ulf Assarsson. Sample-based Visibility for Soft Shadows Using Alias-free Shadow Maps. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering 2008)*, 27(4):1285–1292, June 2008.

[SH05] Christian Sigg and Markus Hadwiger. Fast Third-Order Texture Filtering. In *GPU Gems 2*, pages 313–329. Addison-Wesley, 2005.

[SJW07] Daniel Scherzer, Stefan Jeschke, and Michael Wimmer. Pixel-Correct Shadow Maps with Temporal Reprojection and Shadow Test Confidence. In *Rendering Techniques 2007 (Proceedings of Eurographics Symposium on Rendering)*, pages 45–50. Eurographics Association, 2007.

[SKMW17] Michael Schwärzler, Lisa-Maria Kellner, Stefan Maierhofer, and Michael Wimmer. Sketch-based Guided Modeling of 3D Buildings from Oriented Photos. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D 2017)*, pages 9:1–9:8. ACM, February 2017.

[SLSW13] Michael Schwärzler, Christian Luksch, Daniel Scherzer, and Michael Wimmer. Fast Percentage Closer Soft Shadows using Temporal Coherence. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2013 (i3D 2013)*, pages 79–86, New York, NY, USA, March 2013. ACM.

[SMSW12] Michael Schwärzler, Oliver Mattausch, Daniel Scherzer, and Michael Wimmer. Fast Accurate Soft Shadows with Adaptive Light Source Sampling. In *Vision, Modeling and Visualization 2012*, pages 39–46. Eurographics Association, November 2012.

[SOL$^+$16] Johannes Sorger, Thomas Ortner, Christian Luksch, Michael Schwärzler, Meister Eduard Gröller, and Harald Piringer. LiteVis: Integrated Visualization for Simulation-Based Decision Support in Lighting Design. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):290–299, January 2016.

[SS07] Michael Schwarz and Marc Stamminger. Bitmask Soft Shadows. *Computer Graphics Forum*, 26(3):515–524, 2007.

[SSMW09] Daniel Scherzer, Michael Schwärzler, Oliver Mattausch, and Michael Wimmer. Real-Time Soft Shadows Using Temporal Coherence. In *Advances in Visual Computing: 5th International Symposium on Visual Computing (ISVC 2009)*, Lecture Notes in Computer Science, pages 13–24. Springer, 2009.

[SSS$^+$08] Sudipta N. Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala, and Marc Pollefeys. Interactive 3D Architectural Modeling from Unordered Photo Collections. *ACM Trans. Graph.*, 27:159:1–159:10, 2008.

[SvHG$^+$08] Christoph Strecha, Wolfgang von Hansen, Luc J. Van Gool, Pascal Fua, and Ulrich Thoennessen. On Benchmarking Camera Calibration and Multi-view Stereo for High Resolution Imagery. In *CVPR*, 2008.

[SW08] Daniel Scherzer and Michael Wimmer. Frame Sequential Interpolation for Discrete Level-of-Detail Rendering. *Computer Graphics Forum (Proceedings EGSR 2008)*, 27(4):1175–1181, 2008.

[SWK07] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226, 2007.

[SYM10] Nader Salman, Mariette Yvinec, and Quentin Merigot. Feature Preserving Mesh Generation from 3D Point Clouds. *Computer Graphics Forum*, 29(5):1623–1632, 2010.

[SYM$^+$11] Daniel Scherzer, Lei Yang, Oliver Mattausch, Diego Nehab, Pedro V. Sander, Michael Wimmer, and Elmar Eisemann. A Survey on Temporal Coherence Methods in Real-Time Rendering. In *Eurographics 2011 State of the Art Reports*, pages 101–126. Eurographics Association, 2011.

[Sza18] Attila Szabo. A Composable and Reusable Photogrammetric Reconstruction Library. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, March 2018. 1.

[VAB10] Carlos A. Vanegas, Daniel G. Aliaga, and Bedrich Benes. Building Reconstruction using Manhattan-World Grammars. In *CVPR*, pages 358–365, 2010.

[vdHDT$^+$07] Anton van den Hengel, Anthony Dick, Thorsten Thormählen, Ben Ward, and Philip H. S. Torr. VideoTrace: Rapid Interactive Scene Modelling from Video. *ACM Trans. Graph.*, 26, 2007.

[vGJMR10] Rafael Grompone von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: A Fast Line Segment Detector with a False Detection Control. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 32(4):722–732, 2010.

[Vog13] Günther Voglsam. Real-time Ray Tracing on the GPU - Ray Tracing using CUDA and kD-Trees. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, April 2013.

[Wal17] Andreas Walch. Lens Flare Prediction based on Measurements with Real-Time Visualization. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 2017.

[WFS$^+$15] Jinglu Wang, Tian Fang, Qingkun Su, Siyu Zhu, Jingbo Liu, Shengnan Cai, Chiew-Lan Tai, and Long Quan. Image-based Building Regularization Using Structural Linear Features. *Transactions on Visualization and Computer Graphics*, 1(99):1, 2015.

[WH03] Chris Wyman and Charles Hansen. Penumbra Maps: Approximate Soft Shadows in Real-Time. In *Proceedings of the 14th Eurographics workshop on Rendering*, pages 202–207. Eurographics Association, 2003.

[Wil78] Lance Williams. Casting Curved Shadows on Curved Surfaces. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12(3):270–274, Aug. 1978.

[WKL$^+$18] Andreas Walch, Katharina Krösl, Christian Luksch, David Pichler, Thomas Pipp, and Michael Schwärzler. An Automated Verification Workflow for Planned Lighting Setups using BIM. In *REAL CORP 2018, Proceedings*, REAL CORP, pages 55–65, 2018.

[WLS$^+$18] Andreas Walch, Christian Luksch, Attila Szabo, Harald Steinlechner, Georg Haaser, Michael Schwärzler, and Stefan Maierhofer. Lens Flare Prediction based on Measurements with Real-time Visualization. *The Visual Computer*, May 2018.

[WZ02] Tomás Werner and Andrew Zisserman. New Techniques for Automated Architectural Reconstruction from Photographs. In *Proceedings of ECCV 2002*, pages 541–555, 2002.

[XFT$^+$08] Jianxiong Xiao, Tian Fang, Ping Tan, Peng Zhao, Eyal Ofek, and Long Quan. Image-based Façade Modeling. *ACM Trans. Graph.*, 27(5):161:1–161:10, December 2008.

[YDF$^+$10] Baoguang Yang, Zhao Dong, Jieqing Feng, Hans-Peter Seidel, and Jan Kautz. Variance Soft Shadow Mapping. *Computer Graphics Forum*, 29(7):2127–2134, 2010.

[YNS$^+$09] Lei Yang, Diego Nehab, Pedro V. Sander, Pitchaya Sitthi-amorn, Jason Lawrence, and Hugues Hoppe. Amortized Supersampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2009)*, 28(5):135, 2009.