Scene Reconstruction
from Multi-Scale Input Data

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades
Doktor-Ingenieur

vorgelegt von

Simon Fuhrmann, M. Sc.
geboren in München

Referenten der Arbeit:	Prof. Dr.-Ing. Michael Goesele Technische Universität Darmstadt
	Prof. Brian L. Curless, Ph.D. University of Washington
	Prof. Dr.-Ing. Christian Theobalt MPI Informatik, Saarbrücken
Tag der Einreichung:	23. März 2015
Tag der Disputation:	18. Juni 2015

Darmstädter Dissertation, 2015
D 17

Abstract

Geometry acquisition of real-world objects by means of 3D scanning or stereo reconstruction constitutes a very important and challenging problem in computer vision. 3D scanners and stereo algorithms usually provide geometry from one viewpoint only, and several of the these scans need to be merged into one consistent representation. Scanner data generally has lower noise levels than stereo methods and the scanning scenario is more controlled. In image-based stereo approaches, the aim is to reconstruct the 3D surface of an object solely from multiple photos of the object. In many cases, the stereo geometry is contaminated with noise and outliers, and exhibits large variations in scale. Approaches that fuse such data into one consistent surface must be resilient to such imperfections.

In this thesis, we take a closer look at geometry reconstruction using both scanner data and the more challenging image-based scene reconstruction approaches. In particular, this work focuses on the uncontrolled setting where the input images are not constrained, may be taken with different camera models, under different lighting and weather conditions, and from vastly different points of view. A typical dataset contains many views that observe the scene from an overview perspective, and relatively few views capture small details of the geometry. What results from these datasets are surface samples of the scene with vastly different resolution. As we will show in this thesis, the multi-resolution, or, "multi-scale" nature of the input is a relevant aspect for surface reconstruction, which has rarely been considered in literature yet. Integrating scale as additional information in the reconstruction process can make a substantial difference in surface quality.

We develop and study two different approaches for surface reconstruction that are able to cope with the challenges resulting from uncontrolled images. The first approach implements surface reconstruction by fusion of depth maps using a multi-scale hierarchical signed distance function. The hierarchical representation allows fusion of multi-resolution depth maps without mixing geometric information at incompatible scales, which preserves detail in high-resolution regions. An incomplete octree is constructed by incrementally adding triangulated depth maps to the hierarchy, which leads to scattered samples of the multi-resolution signed distance function. A continuous representation of the scattered data is defined by constructing a tetrahedral complex, and a final, highly-adaptive surface is extracted by applying the Marching Tetrahedra algorithm.

A second, point-based approach is based on a more abstract, multi-scale implicit function defined as a sum of basis functions. Each input sample contributes a single basis function which is parameterized solely by the sample's attributes, effectively yielding a parameter-free method. Because the scale of each sample controls the size of the basis function, the method automatically adapts to data redundancy for noise reduction and is highly resilient to the quality-degrading effects of low-resolution samples, thus favoring high-resolution surfaces.

Furthermore, we present a robust, image-based reconstruction system for surface modeling: *MVE*, the *Multi-View Environment*. The implementation provides all steps involved in the pipeline: Calibration and registration of the input images, dense geometry reconstruction by means of stereo, a surface reconstruction step and post-processing, such as remeshing and texturing. In contrast to other software solutions for image-based reconstruction, *MVE* handles large, uncontrolled, multi-scale datasets as well as input from more controlled capture scenarios. The reason lies in the particular choice of the multi-view stereo and surface reconstruction algorithms.

The resulting surfaces are represented using a triangular mesh, which is a piecewise linear approximation to the real surface. The individual triangles are often so small that they barely contribute any geometric information and can be ill-shaped, which can cause numerical problems. A surface remeshing approach is introduced which changes the surface discretization such that more favorable triangles are created. It distributes the vertices of the mesh according to a density function, which is derived from the curvature of the geometry. Such a mesh is better suited for further processing and has reduced storage requirements.

We thoroughly compare the developed methods against the state-of-the art and also perform a qualitative evaluation of the two surface reconstruction methods on a wide range of datasets with different properties. The usefulness of the remeshing approach is demonstrated on both scanner and multi-view stereo data.

Zusammenfassung

Die Erfassung der Geometrie von Objekten mit Hilfe von 3D-Scannern oder Stereoverfahren ist ein sehr wichtiges und herausforderndes Problem im Bereich der Computer Vision. 3D-Scanner und Stereoalgorithmen rekonstruieren typischerweise Geometrie von nur einem Blickwinkel, und viele dieser Scans müssen zu einer konsistenten Oberfläche zusammengeführt werden. Die Daten von 3D-Scannern haben generell ein niedrigeres Rauschverhalten als Stereomethoden und entstehen oft unter kontrollierten Bedingungen. In bildbasierten Stereoverfahren ist das Ziel die 3D-Oberfäche eines Objektes nur mit Hilfe von Fotos des Objektes zu rekonstruieren. In vielen Fällen ist die aus den Stereoverfahren gewonnene Geometrie mit Rauschen und Fehlrekonstruktionen versehen und enthält große Auflösungsunterschiede. Methoden, welche diese Art von Daten in ein konsistentes Oberflächenmodell zusammenführen, müssen robust gegenüber diesen Datenmängeln sein.

In dieser Doktorarbeit beschäftigen wir uns näher mit Oberflächenrekonstruktion aus 3D-Scannerdaten und aus bildbasierten Rekonstruktionsverfahren. Diese Arbeit untersucht insbesondere die unkontrollierte Situation, bei welcher die Eingabebilder keiner Beschränkung unterliegen, und mit unterschiedlichen Kameramodellen unter beliebigen Licht- und Wetterverhältnissen aufgenommen sein können. Ein typischer Datensatz enthält viele Bilder, welche die ganze Szene überblicken, und nur relativ wenige Bilder, welche Details der Geometrie erfassen. Das Resultat aus diesen Datensätzen sind Messpunkte der Oberfläche mit sehr unterschiedlichen Auflösungen. Wie wir in dieser Arbeit aufzeigen werden ist der Multiskalenaspekt der Messpunkte äußerst relevant für die Oberflächenrekonstruktion, hat allerdings bisher in der Literatur kaum Beachtung gefunden. Die Integration des Multiskalenaspektes in die Oberflächenrekonstruktion kann einen substanziellen Unterschied in der Oberflächenqualität ausmachen.

Wir entwickeln und studieren zwei unterschiedliche Ansätze zur Oberflächenrekonstruktion, welche mit den Herausforderungen von unkontrollierten Daten umgehen können. Der erste Ansatz setzt auf die Vereinigung von Tiefenkarten mit Hilfe einer Multiskalendarstellung der vorzeichenbehafteten Distanzfunktion. Die hierarchische Darstellung erlaubt die Vereinigung von Tiefenkarten ohne geometrische Informationen auf inkompatiblen Skalen zu vermischen, was Details in hoch aufgelösten Regionen besser erhält. Zunächst wird ein Octree durch inkrementelles Hinzufügen von Tiefenkarten erstellt, was zu unstrukturierten Datenpunkten der

Distanzfunktion führt. Eine kontinuierliche Darstellung der Daten wird durch die Erzeugung eines tetraedrischen Netzes definiert. Letztendlich wird eine hochgradig adaptive Oberfläche mit Hilfe des Marching Tetrahedra Algorithmus extrahiert.

Ein zweiter, punktbasierter Ansatz basiert auf einer abstrakteren, impliziten Multiskalenfunktion, welche als Summe von einzelnen Basisfunktionen definiert ist. Jeder Eingabemesspunkt liefert eine einzelne Basisfunktion, welche lediglich durch die Auflösung des Messpunktes definiert ist und somit eine parameterfreie Rekonstruktionsmethode ermöglicht. Die Auflösung eines jeden Messpunktes kontrolliert die größe der Basisfunktion, wodurch die Methode sich automatisch automatisch an die Datenredundanz zum Zweck der Rauschunterdrückung anpassen kann. Ferner ist die Methode äußerst robust gegenüber den negativen Effekten von niedrig aufgelösten Messpunkten auf die Oberfläche und bevorzugt somit die hoch aufgelöste Messpunkte.

Des Weiteren stellen wir eine robuste, bildbasierte Oberflächenrekonstruktionssoftware vor: *MVE*, das *Multi-View Environment*. Diese Software implementiert alle notwendigen Schritte der Rekonstruktion: Kalibrierung und Registrierung der Eingabebilder, Geometrierekonstruktion mit Stereoverfahren, Oberflächenrekonstruktion und Nachbearbeitungsschritte, wie etwa Remeshing und Texturierung. Im Gegensatz zu anderen Softwarelösungen für bildbasierte Rekonstruktion kann *MVE* große, unkontrollierte Multiskalendatensätze sowie auch kontrollierte Datensätze handhaben. Der Grund liegt in der speziellen Wahl der Multi-View Stereo und Oberflächenrekonstruktionsalgorithmen.

Die resultierenden Oberflächen werden durch Dreiecksnetze dargestellt, welche eine stückweise lineare Annäherung an die unbekannten, originalen Oberflächen darstellen. Die einzelnen Dreiecke sind oft so klein, dass sie nur vernachlässigbare geometrische Information beisteuern. Die Dreiecke sind außerdem oft degeneriert was zu numerischen Problemen führen kann. Ein Remeshing-Ansatz wird vorgestellt, welche die Oberflächendiskretisierung verändert, so, dass besser geartete Dreiecke entstehen. Die Punkte des Dreiecksnetz werden dabei anhand einer Dichtefunktion verteilt, welche aus der Krümmung der Geometrie berechnet wird. Das resultierende Dreiecksnetz ist somit besser zur Weiterverarbeitung geeignet und nimmt weniger Speicherplatz in Anspruch.

Die vorgestellten Methoden werden sorgfältig mit dem aktuellen Stand der Technik verglichen. Außerdem wird eine qualitative Evaluierung der beiden Oberflächenrekonstruktionsmethoden auf einer Vielfalt von Datensätzen mit unterschiedlichen Eigenschaften durchgeführt. Die Zweckmäßigkeit des Remeshing-Ansatzes wird sowohl auf 3D-Scannerdaten als auch auf bildbasierten Rekonstruktionen demonstriert.

Acknowledgements

This dissertation would not have been possible without the help of many colleagues, friends and family. Here I would like to thank those individuals for their friendly support and the many technical advises I received.

I am very grateful for the opportunity to work with Michael Goesele in his rapidly evolving research lab. His feedback on research directions, my career, and his support in writing research papers are invaluable. Because I always had a passion for my research and the software engineering work, the four years in Michael's lab were quite an enjoyable experience. I also want to thank my external reviewers, Christian Theobalt and Brian L. Curless, for their interest in my work.

My special thanks goes to all my colleagues in the lab for their technical support, inspiring discussions, proof-reading papers and my thesis, and for making the research lab a friendly environment to work in. In particular, I enjoyed the years sharing one office with Jens filled with many inspiring discussions. I am grateful for Jens' and Ronny's support with many mathematical problems, and Fabian's experience with Structure from Motion which he shared with me in lengthy coding sessions. Many thanks go to Ursula, our secretary, who took care of various business matters.

I am thankful for the scanner datasets provided by Gianpaolo Palma from the CNR ISTI, and the high-quality scans provided by Peter Neugebauer from Polymetric GmbH. The photos for three datasets in Chapter 7 have kindly been provided by Stephan Richter and Stefan Roth at TU Darmstadt.

During my one-year long internship at Google in Seattle I met many great researchers, teachers and advisors. I want to thank my managers, Steve Seitz and Evan Rapoport, who always were an inspiration; Carlos Hernandez, who is a great advisor and always knew how to proceed when research was stuck; Sameer Agarwal and Changchang Wu for many discussions about optimization and Structure from Motion; and David Gallup and Yasutaka Furukawa for helpful conversations about Multi-View Stereo. Many of my colleagues eventually became friends, and I am grateful for my time at Google which I will certainly not forget.

Finally, I which to express my gratitude for my Mom and Dad's unlimited support during my education, and my wife for her understanding during paper deadlines and late night work in busy times. Without the support of my family this work would not have been possible.

Contents

Chapter 1

Introduction

Contents

1.1 Capturing Reality

Human vision (or sight), the capability to focus, perceive and interpret images of visible light may be the most immersive of the human senses. Synthesis of realistic images in order to simulate visual aspects of the real-world is the domain of computer graphics. The inception of computer graphics dates back several decades when it has been used for information visualization. Up to the present day it has developed into a vivid landscape of technologies that play a critical role in many industries. The computer games and movie industry are prime examples. In recent years the development of computer graphics has reached a level where digital objects and characters are almost indistinguishable from their real-life counterparts. Computer generated landscapes and visual effects in movies and video games go far beyond what exists in our real world.

Achieving a high level of realism in appearance is a challenging problem. Especially when designing realistic characters, small imperfections in the details, such as skin materials, eyes or character behavior can quickly result in unaesthetic appearance to the human perception. This is called the *Uncanny Vally* [Mori et al., 2012]. Achieving a fully realistic impression is often beyond what can be synthesised by hand by an artist. To improve on realism, several aspects of the real world can be digitally captured and used for modeling, rendering and animation. Examples include motion capturing, recovering reflectance properties of materials, or digitizing the geometry and appearance of objects and persons [Theobalt et al., 2007].

Capturing and digitizing the real world is a very relevant aspect of creating realistic and detailed maps of our world. Maps have reached a level of quality where precise information in terms of geometry and appearance of many parts of the world is readily available. Example applications include Google Maps and Street View, which offer the user an immersive presentation of cities and navigation information, often in 3D. Manually modelling a single building or a facade requires many expert hours. Modelling thousands of streets by hand is infeasible and automated ways of capturing the reality are required.

Capturing the reality and turning the data into a useful digital description falls into the framework of inverse problems, which are generally hard to solve. Many of these problems are computer vision tasks as we often have to deal with two-dimensional sensor data, such as images.

1.2 Geometry from Images

In this thesis we develop new automated methods for image-based geometry modeling. Our work excels in uncontrolled scenarios with unconstrained camera parameters and lighting conditions. The uncontrolled nature of the input data makes reconstruction a challenging problem. Solving these problems is motivated by inexpensive and fast data acquisition. Instead of careful capture planning and execution by an expert user with dedicated 3D scanners, a set of photos taken with a consumer-grade camera in a simple but systematic fashion often results in very detailed and accurate reconstructions.

For popular tourist sites and architecture, it may not even be required to capture any photos. A plenitude of photos already exist on the Internet and can be harvested from community photo collections. The amount of photos available in online services such as Instagram, Flickr and Facebook is exponentially growing. To present some numbers, in early 2013, Flickr reported a total of over 8 billion uploaded photos[1]. The largest photo collection to date may be on Facebook with a reported total of over 250 billion photos and 350 million uploads every day in 2013[2]. In mid 2014 Yahoo released a data set with 100 million images[3] under the Creative Commons license, a valuable resource free to use for research purposes.

Community photo collections in particular possess some fairly inhomogeneous properties (Figure 1.1). Photos are taken with diverse camera models and lenses, resulting in variable focal lengths and image resolutions. The photos show the subject at different times of day and different days of the year, which leads to a wide spectrum of lighting conditions. The camera positions and orientations are unconstrained such that many photos usually show a broad overview of the scene while others show close-up details of small scene elements. The latter aspect results in a multi-scale sampling of the scene which is improperly handled in current state of the

[1] http://vrge.co/16JRRmJ
[2] http://mashable.com/2013/09/16/facebook-photo-uploads/
[3] http://labs.yahoo.com/news/yfcc100m/

Figure 1.1: Four images of a *Trevi Fountain* photo collection with clutter (tourists), different lighting conditions and scene details at varying scale. Photo credits: Flickr users *Andy Hay*, *April*, *Hank Word*, *David Ohmer*, Creative Commons License.

art reconstruction systems. This motivates the integration of the concept of sample scale in the scene reconstruction approach. Such scale-aware approaches will be developed in this thesis and shown to increase the quality of the results.

Turning the images of, e.g., architecture or a popular tourist site into an immersive 3D experience is certainly an ambitious goal. However, recent advances in the field of photogrammetry paved the way for automated geometry reconstruction solely from uncontrolled images. Although the quality of passive reconstruction is still outperformed by dedicated active scanning devices, image-based reconstruction has many important applications. It recently gained attention in the archaeological field as a tool to document excavation progress on-site by taking lots of photographs. An excavation site is a dynamic environment and continuous surveying helps in recording the progress. This task can be accomplished by, e.g., autonomous drones that continuously survey and map topographic information. The resulting virtual 3D model of the site as well as its change over time can be useful for further planning and cultural heritage preservation.

Another emerging field that makes heavy use of image-based reconstruction is the digitization of real-world characters. In contrast to active 3D scanning, which usually takes some time, multiple photos of an object or person can be captured at a single instance in time using a multi-camera rig. This is a great advantage for capturing moving or deforming objects such as humans and animals. A typical setup consists of about 100 synchronized and calibrated cameras in a cylindrical configuration around the subject. The resulting geometry is usually cleaned and remeshed before it can be used in practice. An impressive demonstration of this technology is maintained, e.g., by *Infinite Realities*[4] and *Ten24 3DScanStore*[5], see Figure 1.2.

In this thesis we study aspects of image-based geometry reconstruction. Given a set of input images that observe an object, or more generally a scene, we are faced with the problem of estimating the geometry of the original scene from the input images alone. A typical pipeline for image-based geometry reconstruction involves

[4] http://www.ir-ltd.net
[5] http://www.3dscanstore.com/

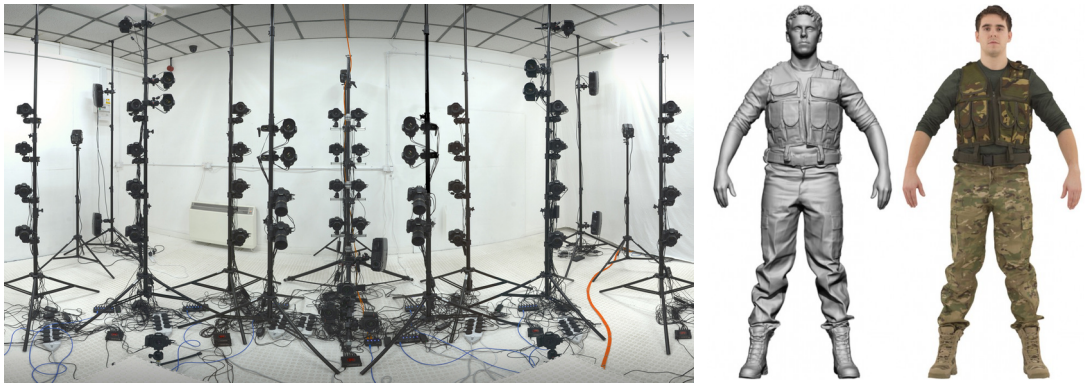

Figure 1.2: *Infinite Realities* uses a 115 camera rig (left) for full-body scans. The instantaneous capture of all cameras even allows to capture objects in motion. Photo courtesy of Lee Perry-Smith. The 80 camera rig of *Ten24 3DScanStore* was used to reconstruct the male model (right), followed by manual clean up. Images courtesy of James Busby.

the following algorithmic steps:

- Structure from Motion (SfM), which recovers the cameras extrinsic parameters (position and orientation) and intrinsic parameters (focal length and radial distortion) given sparse initial correspondences between images. These correspondences are usually established by matching features across images. A sparse point-based 3D representation of the subject is created as a byproduct of the SfM reconstruction.
- Multi-View Stereo (MVS), which, given the estimated camera parameters, reconstructs geometry by finding dense visual correspondences between the images. These correspondences are triangulated and yield a dense set of 3D samples that approximate the surface of the original scene.
- Surface Reconstruction, which takes as input all 3D samples and produces a globally consistent, manifold surface. This surface is usually discretized into a finite number of triangular simplices, yielding a triangle mesh.
- Surface Remeshing, which optimizes the mesh triangles and vertex distribution according to certain criteria, producing an output mesh with improved triangle shapes and sizes.

1.3 Problem Statement

Although a considerable amount of literature has been proposed in the area of image-based reconstruction, many of the involved problems are not adequately solved yet. For example, SfM suffers from long matching and reconstruction times, has trouble identifying and resolving repetitive scene structure, and the usual incremental

approach can lead to drift and inaccuracies that accumulate over long camera trajectories. MVS is mostly restricted to diffuse scenes and high-quality algorithms are computationally demanding, especially for large datasets. The task of surface reconstruction is to generate a surface approximation that resembles the unknown original surface as close as possible solely from imperfect measurements of the surface. Surface reconstruction has mostly been applied to controlled scenes, and the majority of the techniques cannot handle large datasets and samples with mixed scale. In general, only a few image-based reconstruction algorithms are applicable to real-world data sets captured outside a controlled lab environment. This thesis has a focus on less controlled datasets and improves upon the state-of-the-art.

Large datasets

The number of surface samples in a dataset can range anywhere from a few thousand to hundreds of millions of samples. Many methods are global in nature and need to solve an optimization problem in order to obtain the final surface. These approaches are not only demanding in memory consumption, but processing time can be a huge bottleneck. For this reason, many algorithms are unable to handle large datasets. Other algorithms employ uniformly sampled volumetric representations which limits the achievable resolution of the reconstructed surface.

Multi-scale data

The sample acquisition process is usually performed with 2D sensors which have a point spread function. Any acquired sample is the result of an integration process over a certain surface area depending on the resolution and imaging properties of the device. As a consequence, samples are not ideal point samples but have an inherent scale corresponding to the surface area on the object that gave rise to the measurement. This information is often neglected and leads to problems, e.g., if samples at drastically different scales are treated equally.

Varying redundancy

Data redundancy can generally be considered a good thing. It allows consolidation of measurements in order to distinguish between actual information and data noise. While redundancy can be utilized by most reconstruction techniques, it often requires setting a parameter which globally affects the reconstruction operator. Consequently, in the case of spatially varying redundancy, these operators will fail and either smooth away the details or overfit to the data.

1.4 Contributions

In this thesis we make several contributions in the area of image-based geometry reconstruction. In particular, the development of new multi-scale methods for surface

reconstruction will receive greater attention. Although a wealth of surface reconstruction techniques have been proposed in the literature, even recent state-of-the-art methods have several limitations that make them impractical for image-based modelling. Our focus is on uncontrolled data which raises several unsolved challenges.

The main contributions in the thesis have been published as papers at peer-reviewing international conferences. Each published paper is represented as one chapter in this thesis and has mostly been edited for layout. The remeshing approach in Chapter 6 is based on the master's thesis [Fuhrmann, 2009].

- An end-to-end software for image-based geometry modelling is presented. This software, *The Multi-View Environment (MVE)* is capable of performing all major steps of the reconstruction pipeline, i.e., SfM, MVS and surface reconstruction. In contrast to existing commercial software solutions, our software is free and open source. It is targeted towards more uncontrolled scenarios which is reflected in a general-purpose SfM algorithm, a depthmap based MVS implementation and scalable surface reconstruction algorithms.

  Chapter 3, [Fuhrmann et al., 2014]

- A depth map fusion algorithm for surface reconstruction is developed, which takes as input a set of registered depth maps with potentially drastically varying sampling rates of the surface. The method is based on the construction of a novel hierarchical signed distance field represented in an octree. The final surface is extracted as the isosurface of the signed distance field using a Marching Tetrahedra approach.

  Chapter 4, [Fuhrmann and Goesele, 2011]

- A surface reconstruction algorithm is described, which constructs a floating scale implicit function from oriented input samples with scale information. The implicit function is constructed as the sum of compactly supported basis functions defined by the input samples. A final multi-resolution surface mesh is extracted using a variant of the Marching Cubes algorithm over the domain of an octree hierarchy.

  Chapter 5, [Fuhrmann and Goesele, 2014]

- A new remeshing approach is introduced, which distributes a user-defined number of samples according to a prescribed density field over the surface mesh. This initial sample distribution is well suited for isotropic surface remeshing and results in fast convergence behavior of the relaxation procedure.

  Chapter 6, [Fuhrmann et al., 2010]

1.5 Thesis Outline

Chapter 2: Background

We introduce the reader to the area of image-based geometry reconstruction by reviewing some fundamentals in Chapter 2. In particular, we discuss Structure from Motion, Multi-View Stereo, surface reconstruction, remeshing and texturing while reviewing the related work in these areas.

Chapter 3: MVE – The Multi-View Reconstruction Environment

In Chapter 3 we showcase our open source software system for image-based geometry reconstruction. It has been developed at TU Darmstadt over the course of several years in order to ease the work with multi-view datasets and to provide a common software framework for the research group. The system covers the algorithms necessary for high-quality image based reconstructions, i.e., a Structure from Motion algorithm, Multi-View Stereo reconstruction, generation of very dense point clouds, and the reconstruction of surfaces. We show the relevance of such a system in cultural heritage scenarios and low-cost geometry acquisition projects.

Chapter 4: Fusion of Depth Maps with Multiple Scales

In Chapter 4 we introduce a surface reconstruction algorithm by fusing a set of input depth maps. Fusion of depth maps into a globally consistent surface mesh is a challenging problem especially if the depth maps are subject to vastly different sampling rates of the surface. The key ingredient is a hierarchical signed distance field which is able to represent geometry at different levels of detail. We use the concept of the *pixel footprint*, which associates a scale value to every sample of the depth map and drives the construction of the hierarchy. Extracting an isosurface from this hierarchy turns out to be a difficult problem because of missing data. We employ a scattered data interpolation technique by means of a tetrahedralization to extract an isosurface from scattered samples of the signed distance function.

Chapter 5: Floating Scale Surface Reconstruction

Despite the ability to generate high-quality multi-resolution reconstructions using the depth map fusion approach, its memory demands and the long running times led to further research. We develop a follow-up work in Chapter 5 which does not rely on depth maps as input but instead resorts to a more general type of input, namely oriented point samples. Our work distinguishes itself from other point-based reconstruction algorithms in that it uses an additional per-sample scale value as input. This allows identifying redundancy and distinguishing between low- and high-resolution information, and thus avoids intermingling geometry at incompatible scale. The novel formulation of the implicit function scales well to large datasets and produces meshes with fewer holes and a drastically reduced triangle count.

Chapter 6: Direct Resampling for Isotropic Surface Remeshing

While surface reconstruction approaches are usually concerned with generating high-quality geometry, the quality of the triangles as well as the triangle count is usually not considered. In Chapter 6 we develop a remeshing algorithm that optimizes the shapes of the triangles while achieving a pre-determined vertex budget. The main contribution is a fast resampling approach that distributes the required number of vertices over the surface. To avoid oversampling in flat regions, the resampling procedure is guided by a density field to adapt to the geometric complexity of the surface; this yields more sample points in regions with higher curvature.

Chapter 7: Surface Reconstruction Evaluation

We compare the reconstruction results of the presented surface reconstruction algorithms in Chapter 7. In a qualitative comparison of the two approaches we showcase the performance on several types of datasets. This includes controlled scanner data, controlled MVS data as well as multi-scale MVS data. We also demonstrate the effectiveness of the remeshing approach on some datasets.

Chapter 8: Conclusion

Finally, we conclude the thesis in Chapter 8 with a summary of the contributions and a discussion of the proposed techniques. We also identify important areas which would benefit from further research.

Chapter 2

Background

Abstract

In this section we give an introduction to the building blocks of image-based geometry acquisition and review relevant related work in the areas. Given a set of input images, we aim at recovering the camera parameters for the images with respect to a global coordinate system. This process is called *Structure from Motion* and explained in Section 2.2. Once camera parameters for the images are known, stereo vision algorithms are used to compute depth hypotheses from the image intensities. In the case of multiple images, *Multi-View Stereo* is used, which we cover in Section 2.3. The recovered depth hypotheses are then combined into a globally consistent surface representation by means of a *Surface Reconstruction*, introduced in Section 2.4. As the reconstructed surface is subject to discretization into small surface elements (usually triangles), the shape, size and distribution of these elements may be improved by *Surface Remeshing* algorithms. Finally, to make the purely geometric representation visually appealing, *Surface Texturing* is used to color the resulting meshes. We will briefly introduce the concepts of Remeshing and Texturing in Section 2.5 as post-processing operations.

Contents

Figure 2.1: This chapter introduces the basics of image based reconstruction. *Bottom:* 5 of 871 input images of the *Trevi Fountain* dataset downloaded from Flickr, and the corresponding depth maps computed with Multi-View Stereo. *Top:* Composition of the Structure from Motion point cloud and the surface reconstruction result of the *Trevi Fountain*. Image credits: Flickr users *Chris Williamson*, *Andy Hay*, *Giovanna Matarazzo*, *Federico Maccagni*, *Roaming Wab*, Creative Commons License.

2.1 Camera Model

When light interacts with objects, a portion of the light is reflected from the object and scattered in certain directions. The process of taking a photo is to capture the reflected light rays from a single viewpoint onto an image plane using a visual sensor. In order to interpret this visual information contained in images, we need to understand how an image is formed. This formation process is usually formalized using the standard *pinhole camera* model.

The pinhole camera is an idealized mathematical model that describes a camera with an infinitely small hole which gathers rays of light. In theory, the infinitely small hole implies that the whole scene will be in focus, but an infinite exposure is required to gather any light. If the size of the pinhole is increased, more light passes through the hole, which decreases exposure at the expense of an increasingly defocused image. In practice, however, a pinhole which is too small causes light diffraction effects which leads to a blurred image.

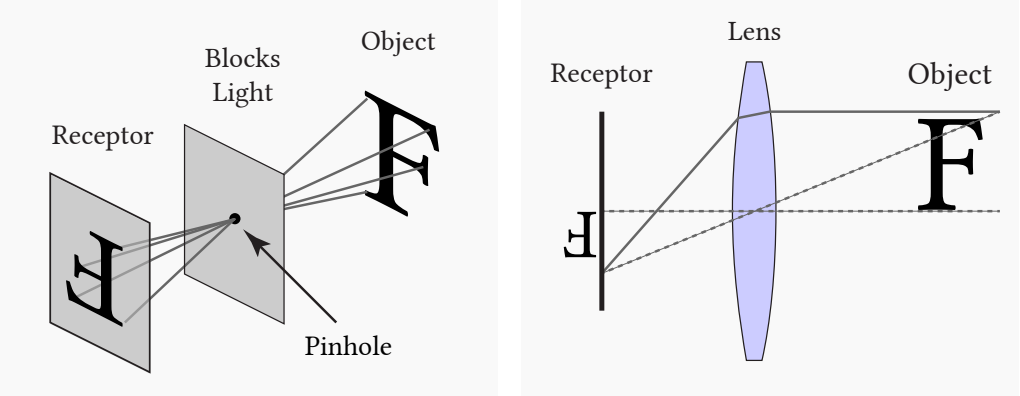

Figure 2.2: Illustration of the pinhole camera model and a lens camera model.

For practical reasons, real camera systems use optical lenses to gather incoming light more efficiently. Figure 2.2 illustrates both concepts. Although these lens systems are designed to mimic the behavior of the standard pinhole camera model, there are certain differences that need to be considered. First, because the aperture of a camera has a certain extent, the captured scene will be in focus only at exactly one depth controlled by the focus settings of the lens. Second, a lens system introduces various kinds of geometric and optical distortions. Such distortions include Barrel distortion, Pincushion distortion and Mustache distortion, see Figure 2.3.

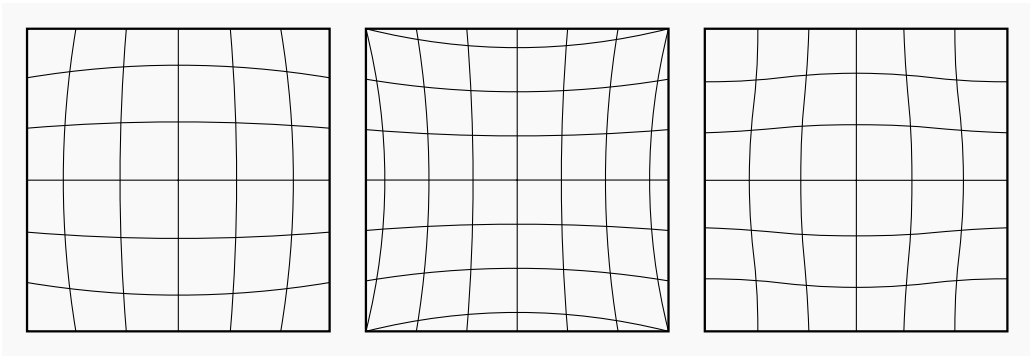

Figure 2.3: Illustrations of Barrel, Pincushion and Mustache distortion (adapted from Wikipedia). Barrel distortion is often associated with wide-angle and fisheye lenses, while Pincushion distortion occurs with telephoto lenses. The Mustache distortion is a more complex combination of a Barrel and Pincushion distortion.

For the sake of simplicity, we mostly work with the assumption of a pinhole camera model whenever we have to deal with geometric calibration of a camera. The limited ability to focus the whole scene does not affect the geometric properties. One source of errors, however, is the lens distortion, which does geometrically affect the image contents in a non-linear way. The usual practice is to explicitly model the distortion and estimate the distortion parameters during Structure from Motion. The distortion is removed from the images for the subsequent steps of the pipeline.

2.1.1 The Pinhole Camera

In the following we introduce the notation of the pinhole camera model and follow standard textbook notation [Hartley and Zisserman, 2004, p. 153] as closely as possible. The pinhole camera is a special case of the general projective camera, namely one that performs a central projection. A general projective camera P maps 3D world points $\mathbf{X}$ to points $\mathbf{x}$ on the image using the notation $\mathbf{x} = \mathsf{P}\mathbf{X}$, see Figure 2.4. The particular projection we are interested in is the *central projection* where every point $\mathbf{X}$ in space forms a ray with the *center of projection* $\mathbf{C}$. This ray intersects with the *image plane*, or *focal plane*, at point $\mathbf{x}$, and $\mathbf{x}$ is called the *image* of $\mathbf{X}$. The center of projection $\mathbf{C}$ is the camera center, and it follows that $\mathsf{P}\mathbf{C} = \mathbf{0}$. Further, the line from the camera center perpendicular to the image plane is called the *principal axis*, and the intersection between the principal axis and the image plane is the *principal point*. The plane parallel to the image plane that passes through the camera center is called the *principal plane* of the camera. Points on the principal plane, such as the camera center $\mathbf{C}$ itself, are a singularity under the central projection as their image is at infinity.

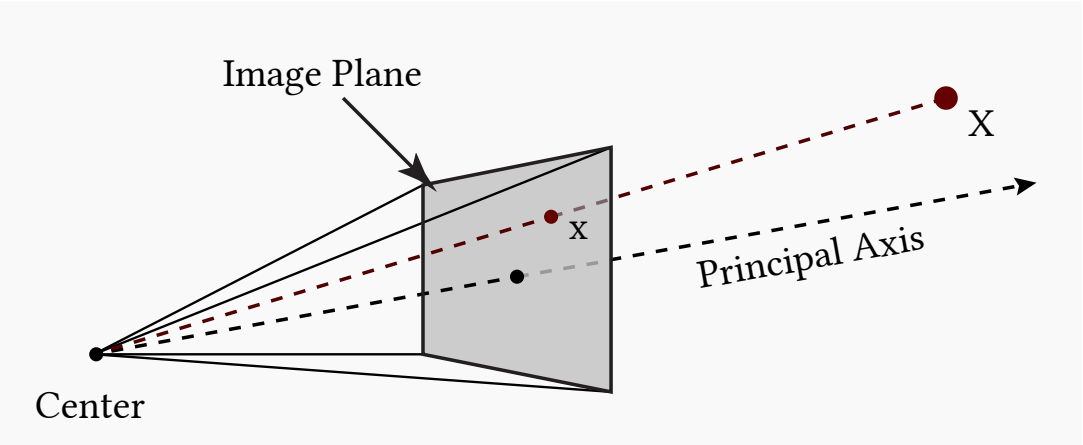

Figure 2.4: Illustration of the camera coordinate frame.

Let the camera center $\mathbf{C}$ be the origin of the Euclidean coordinate system and the focal plane orthogonal to the z-axis at $z = f$. Every point in space $\mathbf{X} = (x, y, z)$ is now mapped onto the image plane at $(fx/z, fy/z, f)$. Ignoring the last coordinate f, we obtain a mapping from the 3D Euclidean space $\mathbb{R}^3$ to the 2D Euclidean

space $\mathbb{R}^2$. This may be written in terms of matrix multiplication using homogeneous coordinates $\dot{\mathbf{X}}$ as

$$\dot{\mathbf{x}} = \mathtt{P}\dot{\mathbf{X}} \quad \Leftrightarrow \quad \begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{pmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{2.1}$$

Principal Point

The notation (2.1) assumes that the principal point (p_x, p_y) of the image plane lies in the origin, which may not be the case in general. The principal point is easily incorporated in the matrix representation as

$$\dot{\mathbf{x}} = \mathtt{P}\dot{\mathbf{X}} = \mathtt{K}\,[\,\mathtt{I} \mid \mathbf{0}\,]\,\dot{\mathbf{X}} \quad \text{with} \quad \mathtt{K} = \begin{pmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{pmatrix}. \tag{2.2}$$

where $\mathtt{I}$ is the 3×3 identity matrix. The matrix $\mathtt{K}$ is called the *calibration matrix* of the camera.

Coordinate Systems

Equation (2.2) assumes that the camera is located in the origin of the coordinate system pointing straight down the z-axis. This is the *camera coordinate frame*. Since points are generally defined in the *world coordinate frame*, a transformation from world to camera coordinates will be applied. This transformation is described by a rotation $\mathtt{R}$ and the camera center $\mathbf{C}$ such that

$$\mathbf{X}_{\text{cam}} = \mathtt{R} \cdot (\mathbf{X}_{\text{world}} - \mathbf{C}) = \mathtt{R}\mathbf{X}_{\text{world}} - \mathtt{R}\mathbf{C} = [\,\mathtt{R} \mid -\mathtt{R}\mathbf{C}\,]\dot{\mathbf{X}}_{\text{world}} \tag{2.3}$$

and introduce the *camera translation* as $\mathbf{t} = -\mathtt{R}\mathbf{C}$ for convenience. Assuming from now on that $\mathbf{X}$ is given in world coordinates, the overall transformation from world coordinates to image coordinates is given by

$$\dot{\mathbf{x}} = \mathtt{P}\dot{\mathbf{X}} = \mathtt{K}\,[\,\mathtt{R} \mid \mathbf{t}\,]\dot{\mathbf{X}} = \mathtt{K}(\mathtt{R}\mathbf{X} + \mathbf{t}). \tag{2.4}$$

This notation separates the camera matrix $\mathtt{P}$ into the camera's *intrinsic parameters* (i.e., the focal length f and the principal point (p_x, p_y)) encoded in the calibration matrix $\mathtt{K}$ and the camera's *extrinsic parameters* (i.e., the rotation $\mathtt{R}$ and translation $\mathbf{t}$) in the transformation matrix $[\,\mathtt{R} \mid \mathbf{t}\,]$. The camera center is then given by $\mathbf{C} = -\mathtt{R}^{-1}\mathbf{t}$. Because the rotation matrix $\mathtt{R}$ is orthogonal, $\mathtt{R}^{-1} = \mathtt{R}^{\top}$.

CCD Cameras

CCD cameras have the additional possibility of non-square pixels, which introduces different scale factors in x and y direction. Let m_x and m_y be the number of pixels per

unit distance in image coordinates in x and y direction, then the calibration matrix $\mathtt{K}$ and inverse $\mathtt{K}^{-1}$ is

$$\mathtt{K} = \begin{pmatrix} fm_x & & m_x p_x \\ & fm_y & m_y p_y \\ & & 1 \end{pmatrix} \qquad \mathtt{K}^{-1} = \begin{pmatrix} \frac{1}{fm_x} & & -\frac{m_x p_x}{fm_x} \\ & \frac{1}{fm_y} & -\frac{m_y p_y}{fm_y} \\ & & 1 \end{pmatrix} \tag{2.5}$$

This camera has 10 degrees of freedom (DoF): 4 DoF for the calibration fm_x, fm_y, $m_x p_x$ and $m_y p_y$, 3 DoF for the rotation $\mathtt{R}$ and 3 DoF for the translation $\mathbf{t}$.

2.1.2 Inverse Projection and Reprojection

The *inverse projection* of a homogeneous 2D image coordinate $\dot{\mathbf{x}}$ to a 3D point in world coordinates $\mathbf{X}$ with respect to a depth d is given by

$$\mathbf{X} = \mathtt{R}^\top(\mathtt{K}^{-1}\dot{\mathbf{x}} \cdot d - \mathbf{t}). \tag{2.6}$$

Here, the depth value d refers to the distance along the z-axis of point $\mathbf{X}_{\text{cam}}$ in camera coordinates (i.e., the z-component of point $\mathbf{X}_{\text{cam}}$). This stands in contrast to another common convention where the depth value $\tilde{d} = \|\mathbf{X} - \mathbf{C}\|_2$ is defined as the distance of a 3D point $\mathbf{X}$ to the camera center $\mathbf{C}$. If not noted otherwise we use the former convention of depth, because it is mathematically more convenient as no normalization of $\mathtt{K}^{-1}\dot{\mathbf{x}}$ is required before multiplying with the depth.

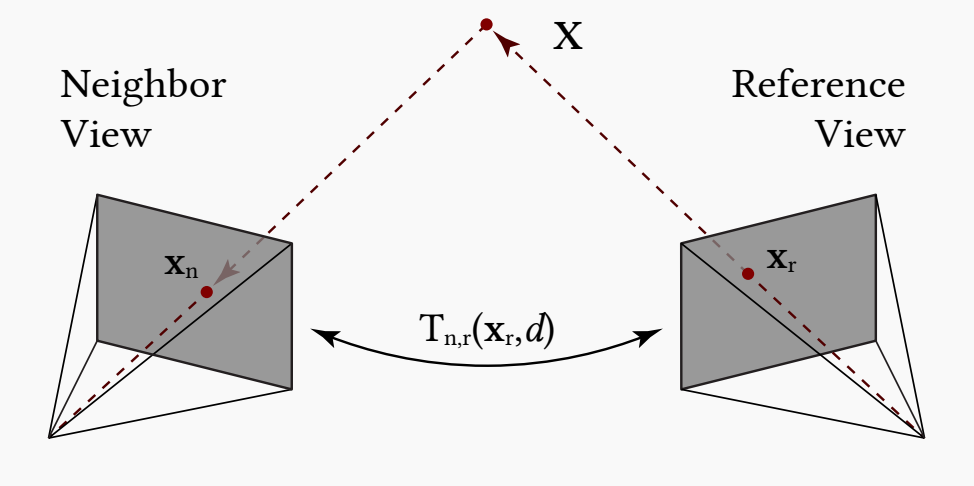

Figure 2.5: Reprojection of a point x_r in the reference camera to point x_n in the neighboring camera.

The *reprojection* of a point $\mathbf{x}_r$ in a reference image I_r with respect to depth d into a neighboring image I_n is a commonly required transformation in Multi-View Stereo, and will thus be introduced here. It consists of chaining the inverse projection of the point $\mathbf{x}_r$ in image I_r with a back projection to a point $\mathbf{x}_n$ in image I_n. Figure 2.5

Figure 2.6: Lenses can introduce a radial distortion of the incoming light rays, which breaks the assumptions of the pinhole camera model. This distortion can be removed from the original image (left) by creating an *undistorted* image (right). The distortion parameters for this image are $\kappa_2 = -1.6$ and $\kappa_4 = 2.1$. Note how, in this example, image information gets pushed out of the frame during undistortion. Other distortions can introduces boundaries with undefined color. Photo credits: Flickr user *Patrick Subotkiewiez*, Creative Commons CC BY 2.0 License.

illustrates the reprojection of a point in the reference view into the neighboring view. The reprojection of $\dot{\mathbf{x}}_r$ with respect to depth d

$$\dot{\mathbf{x}}_n = \mathtt{K}_n(\mathtt{R}_n\mathtt{R}_r^\top(\mathtt{K}_r^{-1}\dot{\mathbf{x}}_r \cdot d - \mathbf{t}_r) + \mathbf{t}_n) \tag{2.7}$$

describes a relative transformation $\mathtt{T}_{n,r}(\mathbf{x}_r, d)$ from image I_r to image I_n and can conveniently be written as

$$\begin{aligned} \mathtt{T}_{n,r}(\dot{\mathbf{x}}_r, d) &= \mathtt{T}_{n,r} \cdot \dot{\mathbf{x}}_r \cdot d + \mathbf{t}_{n,r} && (2.8)\\ \text{with } \mathtt{T}_{n,r} &= \mathtt{K}_n\mathtt{R}_n\mathtt{R}_r^\top\mathtt{K}_r^{-1} && (2.9)\\ \text{and } \mathbf{t}_{n,r} &= \mathtt{K}_n\mathbf{t}_n - \mathtt{K}_n\mathtt{R}_n\mathtt{R}_r^\top\mathbf{t}_r && (2.10) \end{aligned}$$

and the quantities $\mathtt{T}_{n,r}$ and $\mathbf{t}_{n,r}$ can be pre-computed for the image pair I_r, I_n.

2.1.3 Distortion Model

Radial lens distortion is inconvenient to handle because it introduces a non-linear image coordinate transformation and thus cannot be expressed using matrix notation. This transformation also constitutes computational overhead. For these reasons, lens distortion is usually estimated once per image during Structure from Motion, and is then removed from the image, producing an *undistorted* image, see Figure 2.6. Subsequent steps in the pipeline, such as Multi-View Stereo, operate on the undistorted images only.

There are several models for lens distortion. We will introduce the most common, yet simple and powerful model. Consider image coordinates $\mathbf{x}$ obtained by central

projection of 3D points $\mathbf{X}_{\text{cam}}$ in camera coordinates

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{X}_{\text{cam}} = [\ \mathtt{R} \mid \mathbf{t}\ ]\mathbf{X}_{\text{world}} \tag{2.11}$$

and denote $\mathbf{x} = (x/z, y/z)^\top$ the central projection of $\mathbf{X}_{\text{cam}}$. The radial lens distortion is then applied, yielding distorted image coordinates $(x_d, y_d)^\top$.

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(r) \cdot \mathbf{x} \tag{2.12}$$

The distortion $L(r)$ is a function only of the radius (distance from the image center) $r = \|\mathbf{x}\|_2$. An approximation to an arbitrary distortion function is given by the Taylor expansion [Hartley and Zisserman, 2004, p. 189]

$$L(r) = 1 + \kappa_1 r^1 + \kappa_2 r^2 + \kappa_3 r^3 + \kappa_4 r^4 + \dots \tag{2.13}$$

It is common in literature to only consider even exponents [Szeliski, 2010]. The reason is that Barrel and Pincushion distortions both increase quadratically from the image center and are well modeled using κ_2 only. For the Mustache distortion the quartic term κ_4 becomes significant as well; the quadratic term $\kappa_2 r^2$ dominates in the image center and the quartic term $\kappa_4 r^4$ dominates at the image boundaries.

$$L(r) = 1 + \kappa_2 r^2 + \kappa_4 r^4 + \kappa_6 r^6 + \dots \tag{2.14}$$

Because radial distortion is often estimated as part of a larger optimization problem (such as in Bundle Adjustment during Structure from Motion) it is useful to keep the number of parameters at a minimum. Thus, in practice, usually only the parameters κ_2 and κ_4 are considered.

2.2 Structure from Motion

Structure from Motion (SfM) refers to the process of recovering the 3D structure of the scene from a set of images together with the camera parameters of the images. The term *Structure from Motion* is misleading in the sense that it does not refer to moving objects in the scene, but rather to a moving camera, taking photos or video frames. Just as humans perceive many depth cues by moving through their environment, Structure from Motion performs three dimensional reasoning by observing points in space from different points of view.

Although SfM is often referred to as a technique to recover camera poses from 2D visual information, SfM actually does not require any visual information as input. Instead, SfM solely relies on 2D correspondences between images. A *pairwise correspondence* (or just *correspondence* for short) is essentially the information that a point in space $\mathbf{X}$ has been imaged by two cameras, resulting in two observations $\mathbf{x}_1 = \mathtt{P}_1\mathbf{X}$ and $\mathbf{x}_2 = \mathtt{P}_2\mathbf{X}$ in two images. In order to recover the relative pose between images,

several correspondences are required. Obtaining these image correspondences is one of the fundamental problems in computer vision. We first detail the process of finding pairwise image correspondences in Section 2.2.1. Given correspondences, we then describe how the scene is reconstructed using SfM techniques. SfM methods can be divided into two classes: Incremental SfM is introduced in Section 2.2.2, and Global SfM is briefly covered in Section 2.2.3.

2.2.1 Finding Correspondences

Many techniques rely on image correspondences, such as two-view and multi-view stereo, optical flow, and also SfM. Establishing image correspondences is one of the fundamental problems in computer vision. While stereo and flow methods try to estimate dense correspondences, i.e., to find correspondences for all or the majority of the image pixels, SfM, on the other hand, only requires a sparse set of correspondences. Establishing sparse correspondences can be done by hand, e.g., by clicking on locations in images that correspond to the same 3D point. However, this procedure is not feasible for datasets with many images and an automatic procedure is required. Such a procedure first finds "interesting" points in a single image, and then searches for corresponding points in a second image.

Image Features

Just as humans pick visually unique points in the images to identify correspondences across images, feature detection algorithms find visually distinctive regions in the images such as blobs or edges. Once features have been detected, a local image region around each feature is represented by forming a high-dimensional numeric vector, called a *feature descriptor*. Descriptors can then reliably be compared across images in order to find correspondences. Popular image features include SIFT [Lowe, 1999, 2004], SURF [Bay et al., 2008], ORB [Rublee et al., 2011], BRISK [Leutenegger et al., 2011] and DAISY [Tola et al., 2008, 2010].

SIFT features [Lowe, 2004] are probably the most well-known and reliable images features reported in the literature to date. SIFT uses a Difference of Gaussian approach to find distinctive blobs in the image. Because corresponding regions in two images can appear quite different, most feature descriptors are robust against certain changes in appearance. This includes invariance to image rotation, which is achieved by computing the descriptor relative to a dominant orientation of the feature. Further, the size (or scale) of the feature may change, and scale-invariance is achieved by searching for features not only spatially but in scale-space. Invariance to image brightness is achieved by encoding gradients instead of intensity values in the descriptor. Last but not least, invariance to contrast changes is the result of normalizing the descriptors to unit-length. Further, blurring the input images to some degree results in a certain resilience to image noise. See Figure 2.7 for visual representation of SIFT and SURF features on an example image.

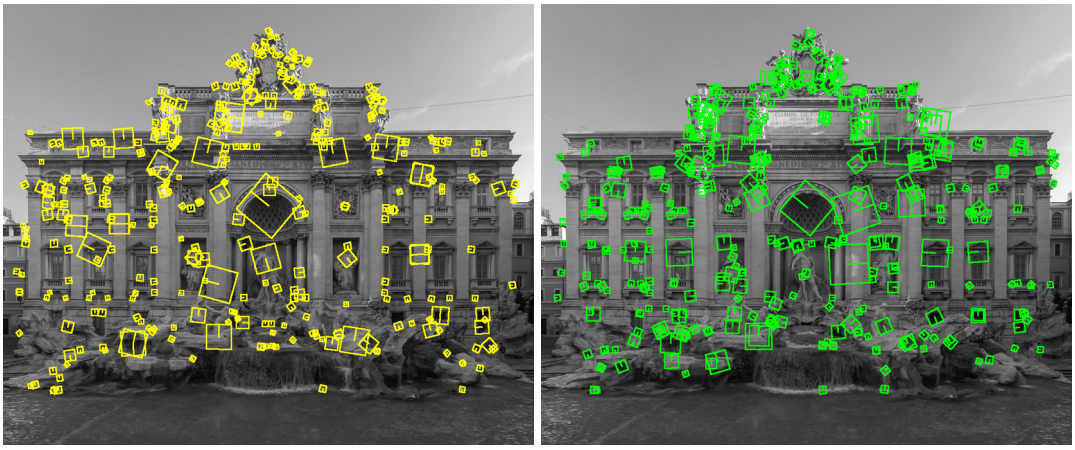

Figure 2.7: The detection of SIFT features [Lowe, 2004] (left, in yellow) and SURF features [Bay et al., 2008] (right, in green) in a photo of the *Trevi Fountain*. Differently sized detections correspond to features at different scale. Photo credits: Flickr user *Nicolas Grevet*, Creative Commons CC BY-NC-SA 2.0 License.

Pairwise Matching

The resulting feature descriptors are essentially high dimensional numeric vectors and can be matched to each other by comparing their Euclidean or angular distances in high dimensional space. The simplest and slowest but most exact procedure to establish correspondences between two images is to match every descriptor of image I_1 to all descriptors of the other image I_2: For every descriptor d of I_1 the nearest neighbor $d_{1\text{st}}$ and second nearest neighbor $d_{2\text{nd}}$ in the second image I_2 are found. In order to eliminate ambiguous nearest neighbors, two thresholds are usually applied. First, a *distance threshold* t_{dist} between the query descriptor d and the nearest neighbor $d_{1\text{st}}$ is checked by

$$\|d - d_{1\text{st}}\|_2 < t_{\text{dist}}. \tag{2.15}$$

The choice of t_{dist} depends on the maximum Euclidean distance between two descriptors, which is 2 for a signed descriptor (e.g., SURF), and $\sqrt{2}$ in the unsigned case (e.g., SIFT). The second threshold is called the *Lowe threshold*, t_{lowe}, which uses the ratio of the nearest and second nearest neighbor distance and has initially been introduced by Lowe [2004] in the context of object recognition:

$$\frac{\|d - d_{1\text{st}}\|_2}{\|d - d_{2\text{nd}}\|_2} < t_{\text{lowe}} \tag{2.16}$$

It turns out that setting a global threshold on the nearest neighbor distance t_{dist} does not perform well because some descriptors are more discriminative than others. In practice it is often sufficient to only use the Lowe ratio threshold t_{lowe}. Depending on whether descriptors are signed or unsigned, t_{lowe} is set to a value of 0.7 and 0.8, respectively. Choosing a smaller ratio leads to fewer but more reliable image correspondences.

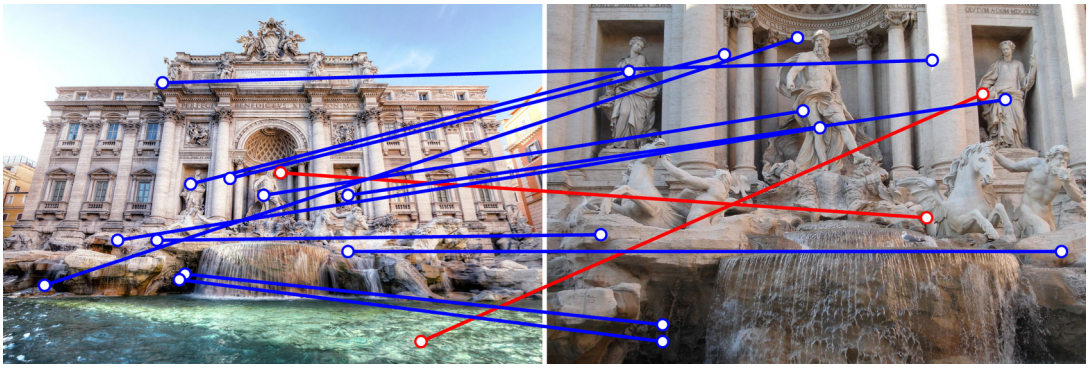

Figure 2.8: Pairwise image correspondences. By enforcing mutual nearest neighbor matches only, some false matches will be filtered away (red lines). Photo credits: Flickr users *NMK Photography*, and *Tom Magliery*, Creative Commons CC BY-NC-SA 2.0 License.

To eliminate inconsistent nearest neighbors, a two-way matching is performed: The matching is repeated but this time descriptors of image I_2 are matched to descriptors of image I_1. Successful matches are kept if and only if the nearest neighbor is mutual, i.e. if $d_{I_1} \rightarrow d_{I_2}$ and $d_{I_2} \rightarrow d_{I_1}$. Figure 2.8 illustrates the effect of enforcing mutual nearest neighbors.

Let m_1 and m_2 be the number of features in image I_1 and I_2. Two-way matching performs a total of $2m_1m_2$ comparisons and is thus computationally expensive. The search for nearest neighbors can be accelerated to sub-linear time using search trees. However, these space partitioning approaches suffer from diminishing returns with increasing dimension, and in practice only approximate nearest neighbors are considered [Muja and Lowe, 2009].

Geometric Filtering

The putative matches so far contain many false correspondences, i.e. correspondences between image positions that belong to different 3D points. Filtering of the matches using the Lowe ratio test and enforcing mutual nearest neighbors is purely based on the values of the descriptors. False putative matches at this stage are likely because many image regions may appear similar although they are geometrically unrelated. Because we assume that the visual information in the images is generated under a perspective camera model, corresponding points in the images are geometrically related. This relationship is expressed in the *fundamental matrix* $\mathtt{F}$, a 3×3 matrix defined as

$$\mathbf{x}_{I_2}\mathtt{F}\mathbf{x}_{I_1} = 0. \tag{2.17}$$

for any corresponding image points $\mathbf{x}_{I_1}$, $\mathbf{x}_{I_2}$ in homogeneous coordinates between images I_1 and I_2. If eight point correspondences are available, the fundamental matrix can be determined (up to a scale ambiguity) with the 8-point algorithm [Hartley, 1997] as the solution of a linear system of equations. The fundamental matrix $\mathtt{F}$ is a singular matrix of rank 2 and the left and right null spaces of $\mathtt{F}$ represent the epipoles

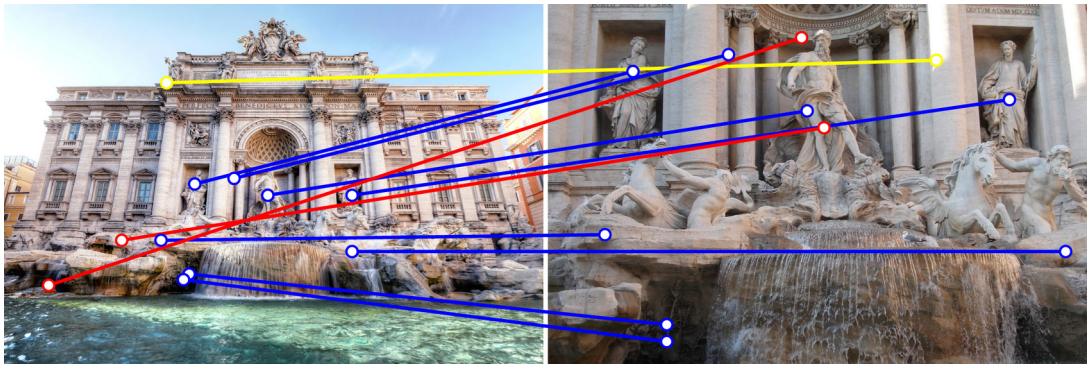

Figure 2.9: Pairwise image correspondences. Geometric filtering of correspondences removes even more outliers (red). Some false matches can survive all filtering steps (yellow) and will either be removed during SfM reconstruction or remain as outlier points in the final reconstruction. Photo credits as in Figure 2.8.

in the two images. In case of noisy image correspondences, the estimated $\mathtt{F}$ will have a rank of 3, and the singularity constraint can be enforced by decomposing $\mathtt{F}$ using the SVD and by setting the smallest eigenvalue to zero.

In practice the correspondences between two images are not only contaminated with noise but also contain many outliers. In this case, $\mathtt{F}$ can be estimated using an iterative RANSAC procedure [Fischler and Bolles, 1981]. In every iteration, eight correspondences are randomly picked, F is computed using the normalized 8-point algorithm, and the number of inliers among all correspondences is recorded. Finally, the matrix $\mathtt{F}$ that leads to the most inliers is chosen as the best estimate for the set of point correspondences and all correspondences that do not support the consensus are rejected as outliers. Figure 2.9 illustrates the effect of geometric filtering.

Note that a non-linear solution for $\mathtt{F}$ is available from 7 point correspondences [Hartley and Zisserman, 2004, p. 281]. If the camera calibration $\mathtt{K}$ and $\mathtt{K}'$ for both images is known, the Essential matrix $\mathtt{E} = \mathtt{K}'^{\top}\mathtt{F}\mathtt{K}$ can be computed from as few as 5 point correspondences [Nistér, 2004]. It is worth considering algorithms that require fewer correspondences because fewer RANSAC iterations are required to find a good solution.

Exhaustive Matching

In order to find all correspondences across all images, every possible image pair has to be matched. Given N input images, the total number of image pairs is $^{N(N-1)}/_{2}$. Pairwise matching is thus clearly a quadratic algorithm in the number of input images and it constitutes the largest bottleneck of Structure from Motion in terms of processing time. Unfortunately, no clear solution to this problem exists. There are several attempts to mitigate this situation, which can broadly be classified into two strategies: One can either reduce the number of potential image pairs to be matched at the cost of losing some matchable image pairs, or one can reduce the number of feature descriptors per image at the cost of losing correct matches. These strategies

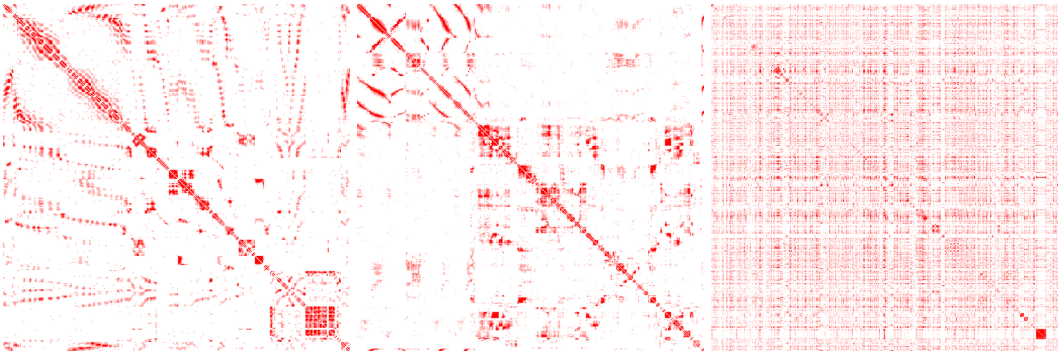

Figure 2.10: The pairwise matching matrices for three datasets: *Die Badende* with 343 images (density 11.8%), *Engel auf der Rosenhöhe* with 453 images (density 9.8%) and the *Trevi Fountain* with 871 images (density 19%). Each matrix entry corresponds to an image pair and color indicates the number of correspondences between a pair.

are complementary and can be combined.

Reducing the number of image pairs is inspired by the fact that, especially in large datasets, only few pairs match with each other. See Figure 2.10 for an illustration of a matching matrix. Quickly deciding if a given image pair matches without performing the full matching procedure can considerably speed up the matching. A common technique is to match a small number N of low-resolution features (e.g. $N = 100$), and to perform a full matching only if the number of low-resolution matches is above a threshold (e.g. 5% of N) [Wu, 2013]. Although this strategy causes many image pairs being rejected that would otherwise match, the loss of quality is negligible as most images pairs will still be transitively linked. Another line of work uses vocabulary trees [Nister and Stewenius, 2006] to construct a weighted bag of words and assigns a single high dimensional vector to every image. Pairwise comparison of these per-image vectors is still quadratic but orders of magnitudes faster than a full matching and can quickly identify potentially matching image pairs [Agarwal et al., 2009; Havlena and Schindler, 2014].

Similarly, reducing the number of descriptors per image before any pairwise matching is performed can drastically reduce the overall matching time. Feature detectors like SIFT or SURF produce many descriptors per image, most of which will be eliminated by the Lowe ratio test and not lead to a successful match. The time spent to find the nearest neighbors for these descriptors in the first place is thus wasted. Learnt feature classifiers [Hartmann et al., 2014] have been used with some success in order to eliminate image descriptors that are unlikely to match later on at the cost of losing many good matches.

Generating Tracks

The previous steps established correspondences of 2D image points between pairs of images. These pairwise correspondences are now chained by transitively expanding them across multiple images. A single connected component of linked 2D points in

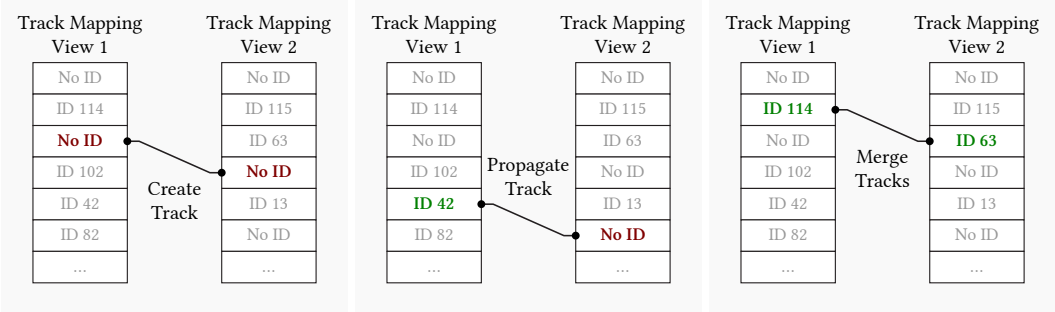

Figure 2.11: Illustration of the three relevant cases when generating tracks.

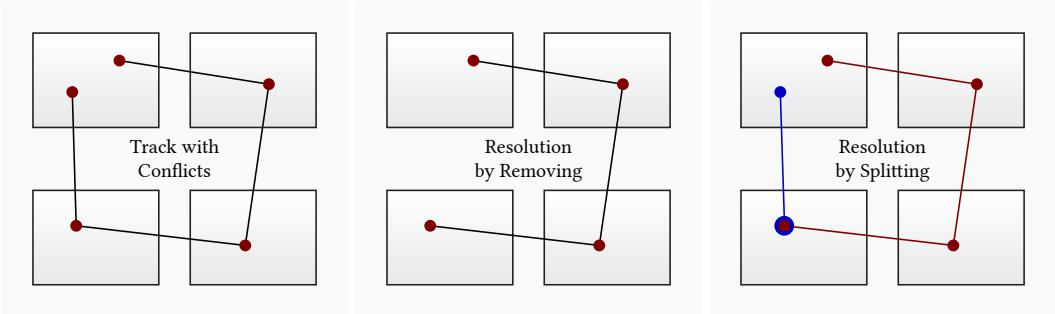

Figure 2.12: A track with two features in the same view is in conflict. Resolution strategies include removing a conflicting feature and splitting the track.

multiple images is called a *feature track*. Each track will eventually be triangulated into a single 3D point once the parameters of at least two cameras in the track are known. Algorithmically, each correspondence of every image pair is traversed, and it is checked whether the two endpoints of a correspondence already have a track ID assigned. Three different situations can occur, see Figure 2.11. First, if both endpoints of a correspondence do not have a track ID assigned, a new track is created and assigned. Second, if only one of the endpoints have a valid track ID assigned, the ID is propagated to the other endpoint. Third, if both endpoints have a different track ID assigned, the two tracks are merged.

After track IDs have been propagated, conflicts are detected and resolved. A conflict is detected if a track contains two or more features in a single image. This is an inconsistent situation because it suggests that a single 3D point projects to two different 2D points in the image, see Figure 2.12. There are several resolution strategies such as splitting the track into two tracks or removing conflicting features from the track. A more conservative strategy, however, is to delete the whole track. Once all tracks are generated, the pairwise matching result is not required anymore.

2.2.2 Incremental Reconstruction

After correspondences have been established and linked between pairs of images the parameters of the camera model (see Section 2.1) can be recovered. The incre-

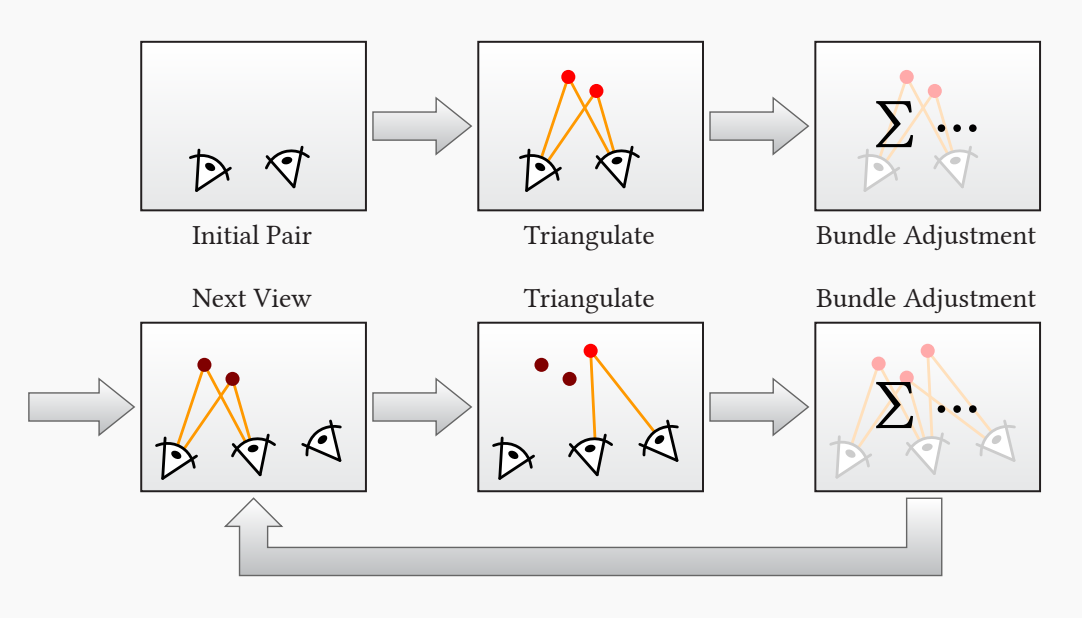

Figure 2.13: The incremental Structure from Motion reconstruction pipeline.

mental approach starts from a suitable initial pair and incrementally adds new cameras to the reconstruction until all cameras are part of the scene. The incremental strategy is still predominant in literature and has been implemented in, e.g., *PhotoTourism* [Snavely et al., 2006], *VisualSfM* [Wu, 2013] and *MVE* [Fuhrmann et al., 2014] for images and also for video [Tompkin et al., 2012]. However, global SfM approaches are becoming more popular. We describe the basics of the global technique in Section 2.2.3. The individual steps of the incremental reconstruction approach are illustrated in Figure 2.13. These steps are now described in more detail.

Initial Pair Selection

The reconstruction of the scene is initiated by computing the relative pose between a suitable image pair. A good initial pair has many correspondences $(\mathbf{x}_i, \mathbf{x}'_i)$ such that many tracks can be triangulated. However, for a robust relative pose and for a well conditioned bundle adjustment optimization the unknown 3D points $\mathbf{X}_i$ leading to the correspondences $(\mathbf{x}_i, \mathbf{x}'_i)$ must not be in a degenerate configuration. In particular, two degenerate cases where all points $\mathbf{X}_i$ lie on a single plane, and where the two cameras have no or little parallax, must be avoided.

In both cases, i.e., if points $\mathbf{X}_i$ are located on a plane, or the cameras do not have enough parallax, then there exists a homography $\mathtt{H}$ that relates the images $\mathbf{x}_i$ and $\mathbf{x}'_i$ of $\mathbf{X}_i$ by a homography transformation $\mathbf{x}'_i = \mathtt{H}\mathbf{x}_i$. $\mathtt{H}$ can be determined using the direct linear transformation (DLT) algorithm [Hartley and Zisserman, 2004, p. 88]. In order to avoid a badly conditioned initial pair, we seek a pair where the correspondences cannot be explained well with a homography. To this end, a homography $\mathtt{H}$ is determined for the correspondences of a candidate initial pair using RANSAC,

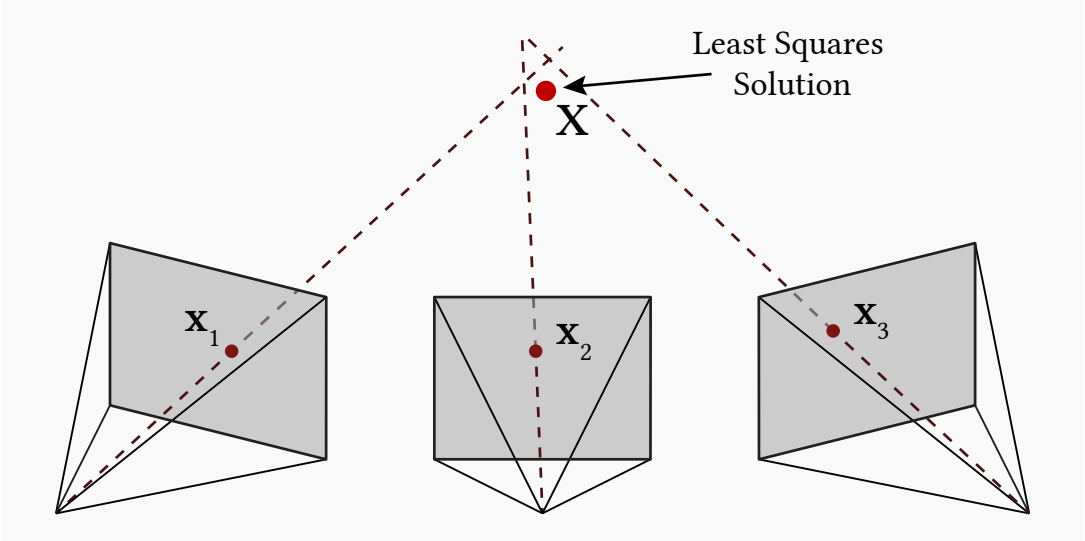

Figure 2.14: Triangulation of tracks. Because the measurements $\mathbf{x}_i$ are not precise, the corresponding rays will not intersect in 3D space and a least squares solution $\mathbf{X}$ is found that minimizes the squared distance to all rays.

and if the number of inliers is above a threshold, e.g., 50%, the initial pair is rejected. A popular initial pair selection approach is to iteratively try candidate initial pairs ordered by the number of correspondences, and pick the first pair where the number of homography inliers is below the threshold.

Once an initial pair has been found, the relative orientation of the initial pair is extracted. Given the fundamental matrix $\mathtt{F}$ computed from the correspondences, $\mathbf{x}_i'\mathtt{F}\mathbf{x}_i = 0$, and given the intrinsic calibration of the two cameras $\mathtt{K}$ and $\mathtt{K}'$ (e.g., from camera EXIF tags), the essential matrix $\mathtt{E}$ is given by $\mathtt{E} = \mathtt{K}'^{\top}\mathtt{F}\mathtt{K}$ [Hartley and Zisserman, 2004, p. 257]. If we now assume that the first camera is in canonical form, i.e. $[\mathtt{R} \mid \mathbf{t}] = [\mathtt{I} \mid \mathbf{0}]$, then four possible solutions for the second camera $[\mathtt{R}' \mid \mathbf{t}']$ can be obtained from $\mathtt{E}$ using the SVD [Hartley and Zisserman, 2004, p. 259]. There exists only one correct pose, and it is sufficient to triangulate a single point and test if it is in front of both cameras to identify the correct solution.

Triangulating Tracks

Each track with at least two reconstructed cameras can now be triangulated in order to obtain a 3D point $\mathbf{X}$. In each camera we have the images of $\mathbf{X}$ given as $\mathbf{x} = \mathtt{P}\mathbf{X}$, $\mathbf{x}' = \mathtt{P}'\mathbf{X}$. These measurements can be combined into a linear system of equations and solved for the unknown $\mathbf{X}$. If more than two cameras are given one obtains an overdetermined system of equations and the least-squares solution can be found via SVD (see Figure 2.14). This is the DLT triangulation method [Hartley and Zisserman, 2004, p. 312], which is simple but not optimal in terms of the reprojection error in the images. However, the obtained 3D point is usually a good initialization and can further be optimized by bundle adjustment, which is explained next.

Bundle Adjustment

Given a set of 3D points $\mathbf{X}_i$ and reprojections of point i in camera j, $\mathbf{x}_{i,j} = \mathtt{P}_j\mathbf{X}_i$, the *reprojection error* is the squared Euclidean distance from the projection $\mathbf{x}_{i,j}$ to the original feature detection $\hat{\mathbf{x}}_{i,j}$ in image space

$$d_{i,j} = \|\mathtt{P}_j\mathbf{X}_i - \hat{\mathbf{x}}_{i,j}\|_2^2 \tag{2.18}$$

The goal of bundle adjustment is to optimize 3D point positions and certain camera parameters such that the reprojection error of the 3D features into the images is minimized. Let $\mathbf{x}$ be a vector of parameters, i.e. a vector of 3D point positions and camera parameters, and let $f(\mathbf{x})$ be the vector of residuals, i.e. the reprojection errors $d_{i,j}$, then bundle adjustment tries to obtain optimal parameters $\mathbf{x}^*$ that minimize the non-linear least-squares problem [Lourakis and Argyros, 2009; Wu et al., 2011]:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_i \|f_i(\mathbf{x})\|^2. \tag{2.19}$$

The results of the optimization procedure are new 3D point positions and camera parameters. What makes bundle adjustment so important is that most initial parameters are usually quite imprecise. The triangulated 3D points are optimal in the sense that they minimize the intersection distance in 3D, not the reprojection error. Furthermore, the camera's focal length is influenced by focus settings, which is not reflected in the image EXIF information. Additionally, the input photos are often distorted and violate the assumptions of the pinhole camera model. To this end, bundle adjustment also estimates the distortion parameters which drastically reduces the overall error. Without bundle adjustment, the incremental addition of new cameras quickly accumulates large errors and is likely to fail.

Next View Selection

Once 3D points have been estimated, new cameras can be added. In contrast to the relative pose estimation used for the initial pair, all other cameras are reconstructed using perspective absolute pose estimation by considering 2D-3D image-to-point correspondences. The most suitable next camera is the one that observes the most 3D points, i.e. the one with the most 2D-3D correspondences.

Given sufficiently many 2D-3D correspondences $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$ for a new camera, the camera matrix $\mathtt{P}$ can be determined such that $\mathbf{x}_i = \mathtt{P}\mathbf{X}_i$. The minimum number of 2D-3D correspondences in order to obtain a unique solution is 6, and this algorithm is called the *absolute 6-point algorithm* [Hartley and Zisserman, 2004, p. 182]. Because the calibration matrix $\mathtt{K}$ is assumed to be known, it is not necessary to compute $\mathtt{P} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}]$ as we are only interested in $[\mathtt{R} \mid \mathbf{t}]$. Knowing $\mathtt{K}$, it is possible to obtain a solution for $[\mathtt{R} \mid \mathbf{t}]$ with a minimum of three 3D-2D correspondences [Haralick et al., 1994], leading to the *perspective 3-point problem* [Kneip et al., 2011]. These algorithms return up to four solutions which can be disambiguated using a fourth point.

Once the parameters of the new camera are known, new tracks become available for triangulation. It is good practice to run bundle adjustment after every newly added camera. However, bundle adjustment is an expensive operation. For large datasets it is common practice to add multiple cameras at a time before a full bundle adjustment is performed. The last steps, i.e. reconstructing a new camera, triangulating new tracks, and regularly performing bundle adjustment, are repeated until all images are part of the scene.

2.2.3 Global Reconstruction

While the incremental reconstruction strategy (see Figure 2.13) is still most widely used, there are certain issues with pose drift due to camera calibration error accumulation, especially in datasets with long camera trajectories. The drift in camera poses leads to a "bending" of the geometry, and returning to a past location might not be reflected in the reconstructed geometry. This issue is known as the *loop closure problem*, and is well studied in the literature [Estrada and Tard, 2005; Williams et al., 2009]. Even though the (exhaustive) pairwise matching clearly indicates that a previously visited area re-appears in the images (*loop detection*), loops may not be closed because the cameras are incrementally placed at either end of the loop, rejecting good matches as geometric outliers.

In contrast, global SfM methods distribute the residual errors evenly across the dataset, effectively avoiding drift [Moulon et al., 2013; Wilson and Snavely, 2014]. Global methods are a promising direction for future SfM, and will briefly be treated here. The basic idea of global SfM methods is to globally optimize the *pose graph*, which consists of pairwise relative transformation between the views [Kummerle et al., 2011; Engel et al., 2014]. Contrary to incremental methods, which perform multiple bundle adjustments after adding new cameras to the scene, global methods fuse all relative motions in a single optimization step.

The main challenges in global SfM are two-fold: First, some pairwise relations, even though they share many correspondences, are mismatches due to repetitive patterns. These outliers cause problems during least-squares optimization. Second, although the relative rotation between two views can precisely be estimated [Hartley and Zisserman, 2004, p. 258], the relative translation is often unstable and can only be recovered up to scale. For this reason some global approaches work with relative rotations and *directions* (rather than with the full relative transformation) and concentrate on outlier removal [Wilson and Snavely, 2014]. Other approaches try to remove the translation scale ambiguity [Moulon et al., 2013; Jiang et al., 2013]. The individual steps of the global SfM procedure are summarized as follows:

- Pairwise relative rotations $\mathtt{R}_{ij}$ and translation directions $\mathbf{t}_{ij}$ between views i, j are computed from the Essential matrix. The relative translations turn into translation directions due to the scale ambiguity.
- The *epipolar graph* is built whose nodes represent the cameras and the edges represent the pairwise relationship $(\mathtt{R}_{ij}, \mathbf{t}_{ij})$ between the cameras.

- Global rotations $\mathtt{R}_i$ are estimated using *rotation averaging* [Hartley et al., 2013] while identifying and rejecting outlier pairs.
- Relative translation scales λ_{ij} are estimated from the directions $\mathbf{t}_{ij}$ and the global rotations $\mathtt{R}_i$ and lead to absolute translations $\mathbf{T}_i$.
- Finally, a global model is created by linking feature tracks across images, triangulating track positions and running a final bundle adjustment.

2.2.4 Reconstruction Ambiguity

It is worth noting that any Structure from Motion reconstruction is ambiguous. It is generally not possible, with any number of views, to recover the absolute position, orientation or scale of the scene. The reconstruction is thus unique up to a 7-dimensional ambiguity described by a similarity transformation that allows for scaling, translation and rotation.

In fact, during reconstruction of the initial pair, these ambiguities are resolved by setting the first camera to the canonical form $[\mathtt{I} \mid \mathbf{0}]$ (determining the absolute position and orientation of the scene) and by constraining the translation between the initial pair to $\|\mathbf{t}'\| = 1$ (determining the scale of the scene).

2.3 Multi-View Stereo

From the human visual system we know that perception of depth and three dimensional structure of the surroundings is mostly a result of binocular vision (i.e., seeing with two eyes). The process of inferring three dimensional structure from images is called binocular stereopsis, or simply stereo. The remarkable performance of the human vision apparatus to identify and correctly interpret three dimensional structure can be attributed to several clues such as the interpretation of occlusions, the size of objects, scene haze and desaturation, linear perspective, etc. In computer vision algorithms, these clues are very difficult to exploit as high-level knowledge of the scene is required. Thus most techniques rely purely on visual information from different viewpoints, i.e., basically exploiting motion clues for stereo reconstruction. This is somewhat similar to Structure from Motion: Visual information from different points of view is used to reason about the structure of the scene. But while SfM requires *sparse* correspondences as input to reconstruct camera parameters, MVS requires camera parameters as input to establish *dense* correspondences.

The basic problem statement for Multi-View Stereo is as follows: *Given several images of an object or a scene, compute a 3D shape representation.* This is a very generic statement, and the usual assumption is that camera parameters for the images are given. An image with associated camera parameters is called a *view*. The number of views can range from just two up to thousands, and both of these extremes are usually treated as special cases: Many specialized binocular stereo algorithms exist for the case of two views [Scharstein and Szeliski, 2002]. Video-based MVS approaches utilize the massive amount of small baseline views, can rely on

a simple view selection and small matching windows [Newcombe et al., 2011; Vogiatzis and Hernandez, 2011]. Most MVS algorithms, however, focus on a sparse set of input images with tens, sometimes hundreds of views, depending on the size and complexity of the scene [Seitz et al., 2006]. The object or scene itself is usually assumed to be *diffuse*, i.e., the appearance of the scene is assumed to be independent of the viewpoints (which is not true for specular surfaces). Finally, there are many possible shape representations, such as individual depth maps, point clouds, meshes, and volumetric representations. Because so many fundamentally different approaches exist, it is difficult to provide a generic description for MVS algorithms. In the following we describe some common steps that most MVS algorithms perform.

2.3.1 View Selection

Given a reference view for which geometry is to be reconstructed, a set of neighboring views is required for stereo matching. Clearly, this selection depends on the stereo algorithm at hand, e.g., in a video-based system, the structured nature of the views can be exploited by selecting a couple of previous and next frames in the video sequence [Newcombe et al., 2011]. In general, the selection of suitable neighboring views involves a trade-off. As the triangulation angle of a 3D point in space between the two views increases, the corresponding image regions become increasingly dissimilar and occlusions become more likely, thus reliable matching becomes harder. A small triangulation angle restricts the search for a correspondence to a smaller search window, but leads to an unstable 3D position with a large depth uncertainty. This is illustrated in Figure 2.15. The view selection makes this compromise between using wide and small baseline views, so that matching is possible while depth estimation is still well conditioned.

Although the view selection can be based purely on the angle between the principal axis of the views, the triangulation angle ultimately depends on the 3D position of the triangulated point. Thus, utilizing the 3D feature points from SfM, if available, will yield more accurate results. A successful view selection has been introduced by Goesele et al. [2007]. It greedily selects neighboring views that share many SfM features with the reference view, encouraging large triangulation angles and a similarity in *scale*, discouraging neighboring views with insufficient resolution.

2.3.2 Measuring Similarity

Given a carefully selected set of neighboring views, the next step in MVS reconstruction is to establish dense correspondences between the pixels in the master view and the neighboring views. The search for a correspondence can be restricted to the *epipolar line*, which is essentially the viewing ray associated with a pixel in the master view reprojected in the neighboring view, see Figure 2.16. The search is performed over a depth range for a given pixel in the master view, and the respective 3D position is reprojected in the neighboring views in order to evaluate

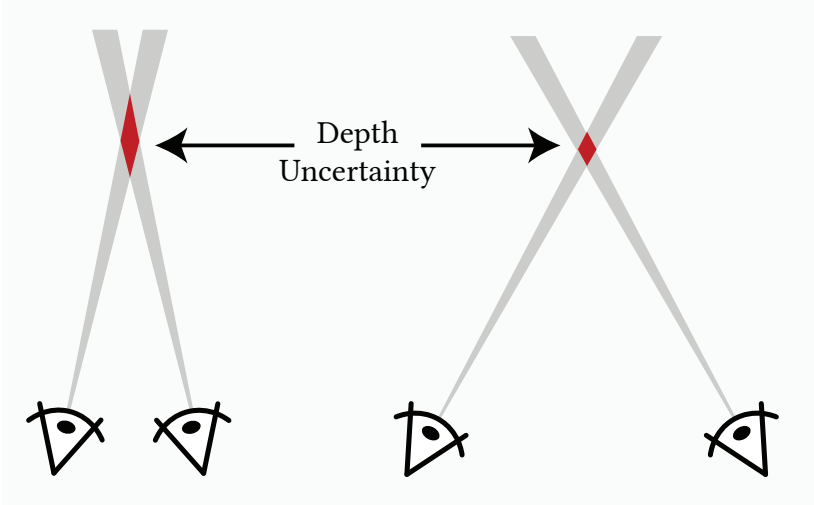

Figure 2.15: Illustration of depth uncertainty depending on the triangulation angle. The left image shows a large depth uncertainty due to a small triangulation angle, however, a small amount of parallax leads to more stable correspondences. A larger triangulation angle leads to less depth uncertainty but increased difficulty in finding visual correspondences.

similarity. Because this similarity is purely based on photometric information, it is often referred to as *photometric consistency*, or *photo-consistency*.

Photometric consistency can be measured in several ways. A common approach is to consider a planar, fronto-parallel patch of a certain size (e.g., 5×5 pixels) in the master view $\mathbf{p}_m$ and to reproject the sampling points into the neighboring view, yielding a transformed patch $\mathbf{p}_n$. The rationale behind this is that small regions on the surface can be approximated by a plane. The color values in patches $\mathbf{p}_m$ and $\mathbf{p}_n$ can then be compared using a similarity measure $S(\mathbf{p}_m, \mathbf{p}_n)$.

The simplest possible similarity metric is to compare the color values i for every sampling point of the patches $\mathbf{p}_m$ and $\mathbf{p}_n$, and measure the error as the sum of absolute differences (SAD):

$$S_{\text{SAD}}(\mathbf{p}_m, \mathbf{p}_n) = \sum_i |\mathbf{p}_m(i) - \mathbf{p}_n(i)| \tag{2.20}$$

This metric is an error measure and small values correspond to more similar patches. A related similarity measure is to compute the sum of *squared* differences (SSD), which penalizes larger errors even more, see Figure 2.17:

$$S_{\text{SSD}}(\mathbf{p}_m, \mathbf{p}_n) = \sum_i (\mathbf{p}_m(i) - \mathbf{p}_n(i))^2 \tag{2.21}$$

Both SAD and SSD are sensitive to exposure differences in the images, which leads to an multiplicative intensity change in the patch. False correspondences become more likely because structural similarity becomes less important. To this end, the

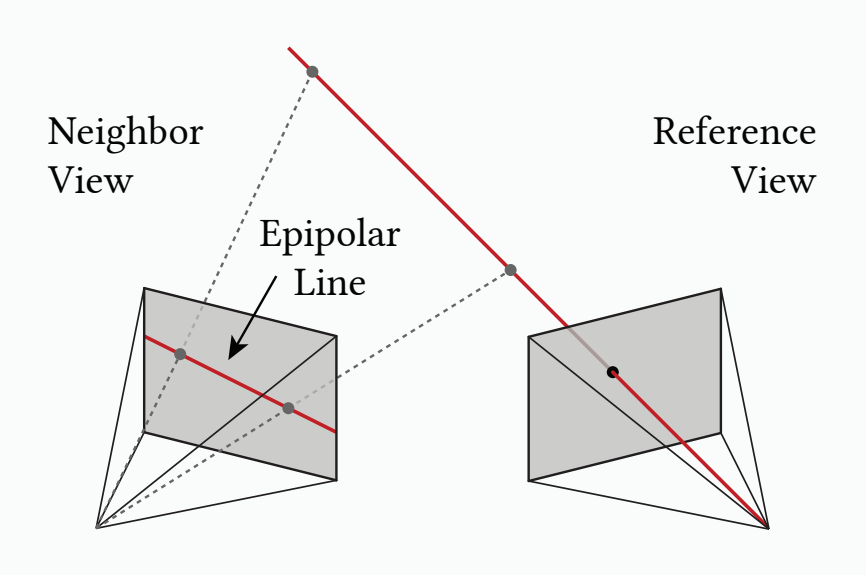

Figure 2.16: The ray corresponding to a pixel in the master view reprojects as the *epipolar line* into the neighbor view.

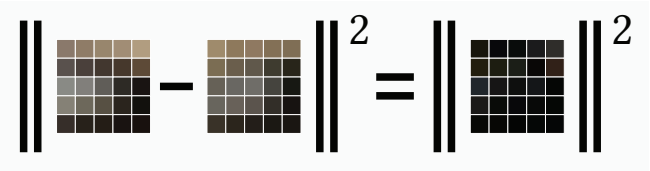

Figure 2.17: Similarity is measured along the epipolar line by comparing patches, e.g. using the sum of squared differences (SSD).

mean intensity of the patch can be removed, which leads to the mean-removed sum of squared differences (MR-SSR):

$$S_{\text{MR-SSD}}(\mathbf{p}_m, \mathbf{p}_n) = \sum_i \left((\mathbf{p}_m(i) - \bar{\mathbf{p}}_m) - (\mathbf{p}_n(i) - \bar{\mathbf{p}}_n)\right)^2 \tag{2.22}$$

Here, $\bar{\mathbf{p}}$ is the mean of patch $\mathbf{p}$. Contrast differences are another common artifact in photographs, therefore the above similarity metrics can report large errors at true correspondences. This is especially true with uncontrolled photos taken under non-uniform lighting conditions. The normalized cross-correlation (NCC) is a popular measure in signal processing and function analysis, which is invariant to both the level (additive difference) and strength (multiplicative difference) of the correlated signals. NCC is defined as

$$S_{\text{NCC}}(\mathbf{p}_m, \mathbf{p}_n) = \frac{1}{N} \sum_i \frac{(\mathbf{p}_m(i) - \bar{\mathbf{p}}_m) \cdot (\mathbf{p}_n(i) - \bar{\mathbf{p}}_n)}{\sigma_m \sigma_n} \tag{2.23}$$

where σ_x is the standard deviation of the patch $\mathbf{p}_x$. In contrast to the other similarity measures, NCC is a correlation, thus larger values correspond to more similar

patches, with the minimum and maximum score being -1 and 1, respectively. If one patch has uniform color, its standard deviation is zero and the S_{NCC} is undefined. In case of a uniform but noisy image patch, NCC is basically correlating the image noise, which is usually dissimilar and results in a small NCC score.

A conceptually different measure has recently gained renewed interest. The non-parametric local *Census transform* [Zabih and Wood, 1994] summarizes local image structure in a bit-vector. Instead of relying on absolute intensity values, Census encodes the relative intensities of patch pixels $\mathbf{p}(i)$ with respect to the central pixel $\mathbf{p}(c)$ in the patch. The Census bit-vector $\mathbf{b}(\mathbf{p})$ of a patch $\mathbf{p}$ is given by

$$\mathbf{b}(\mathbf{p}) = \bigotimes_{i \text{ s.t. } i \neq c} \xi(\mathbf{p}(c), \mathbf{p}(i)) \quad \text{with} \quad \xi(x, y) = \begin{cases} 0 & \text{if } x < y \\ 1 & \text{if } x \geq y \end{cases} \tag{2.24}$$

where $\otimes$ denotes bit-wise concatenation. The dissimilarity between two patches can then be computed using the Hamming distance between the bit-vectors.

$$S_{\text{Census}}(\mathbf{p}_m, \mathbf{p}_n) = \| \mathbf{b}(\mathbf{p}_m) - \mathbf{b}(\mathbf{p}_n) \|_{\text{Hamming}} \tag{2.25}$$

This transform is invariant to exposure and contrast changes and only bit-wise operations are needed for comparison. The binary representation, however, is less discriminative and the dissimilarity can only take a small number of discrete states.

2.3.3 Geometry Estimation

Now that the basic ingredients of Multi-View Stereo have been covered, different strategies for geometry estimation are discussed.

Probably the most straightforward approach is the computation of depth maps in a brute-force fashion [Goesele et al., 2006]. Given a central view and a selection of neighboring views, the samples of a patch $\mathbf{p}$ in the central view are reprojected into the neighboring views with respect to a depth d, and the similarity is evaluated using one of the photo-consistency measures presented in Section 2.3.2. In order to find a good depth value d where photometric error is low, the depth range $[d_{\text{min}}, d_{\text{max}}]$ with depth increments Δd is exhaustively evaluated. These brute force approaches usually employ a *winner take all* strategy by choosing the depth value that resulted in the best photo-consistency score for each pixel, which leads to noisy depth maps.

Such a brute-force approach is computationally very expensive. In particular, reprojecting every pixel of a patch with respect to a depth d will do a lot of redundant computations as pixels of neighboring patches overlap. A conceptually equivalent but more efficient strategy is to warp the whole neighboring view with respect to a fronto-parallel plane induced by depth d. Photo-consistency can then be evaluated on the central image and the warped neighboring image for all pixels at once without additional reprojection overhead. Furthermore, if the photo-consistency measure performs per-pixel operations only (e.g., SAD and SSD), the patch-based operations can be cast as a per-pixel operation followed by a box-shaped low-pass filtering step on the resulting image. A convolution with a Gaussian corresponds to circular

Figure 2.18: Depth map reconstruction using [Goesele et al., 2007]. The input photograph (left), the reconstructed depth map (middle) and a rendering of the triangulated depth map (right). Photo credits: Flickr user *Nathan Laurell*, Creative Commons CC BY 2.0 License.

patches with spatially varying weights, which can improve quality. Because this approach processes one plane at a time that is swept along depth, this algorithm is called *plane sweeping stereo* [Yang and Pollefeys, 2003]. Traditionally these planes are fronto-parallel to the central view and tend to perform badly on slanted surfaces. In order to improve the reconstruction, slanted planes can be used [Gallup et al., 2007].

Because geometry is *smooth* in the sense that neighboring pixels often have a similar depth value, regularization is employed by more sophisticated MVS approaches. One mild form of regularization is implemented by Goesele et al. [2007], which uses a region-growing approach. Starting from initial sparse depth estimates obtained from SfM feature points, these depth values are propagated to neighboring pixels and used as initialization for depth and normal optimization using NCC photo-consistency scores. A master image and the reconstructed depth map is shown in Figure 2.18. In a similar fashion, Furukawa and Ponce [2010] employ a *match, expand and filter* procedure. An initial set of patches is determined by finding Harris and Difference-of-Gaussian features in each image, and generating candidate patches by searching corresponding features along the epipolar line in the neighboring images, and triangulating the features. The candidate patches are optimized using NCC photo-consistency measures and new patches are created by spatial expansion. Erroneous patches are eliminated in a filtering step by identifying occluding patches (outside the true surface) and occluded patches (inside the true surface).

Algorithms for video data can use many small-baseline neighboring views for more robust Multi-View Stereo matching. Gallup et al. [2007] use SAD photo-consistency on small patches with per-frame gain compensation and simple occlusion handling in a plane-sweep framework. Newcombe et al. [2011] use a large number of neighboring frames and pixel-wise absolute differences (SAD with 1×1 windows) with a GPU friendly regularization, see Figure 2.19.

Another line of work is based on variational mesh refinement driven by photo-consistency optimization [Vu et al., 2009, 2012]. First, a dense set of points is extracted using guided matching, e.g., by detecting Harris corners or Laplacian-of-Gaussians blobs, and filtered using multi-scale NCC with several window sizes. Second, an initial surface is extracted from the point cloud by building a global Delaunay

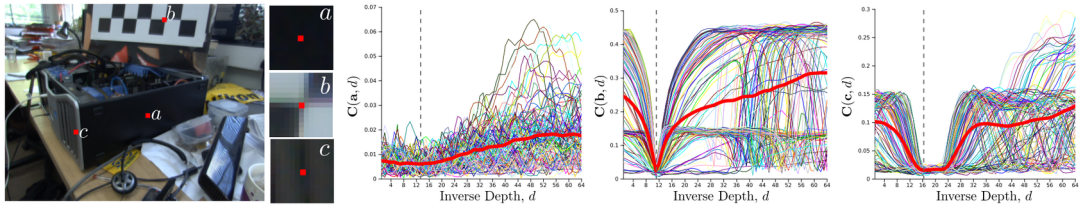

Figure 2.19: Multi-View Stereo matching on many small-baseline frames. The illustration shows the cost curves for three example pixels in the image. While textureless and repetitive regions are difficult to reconstruct, the total cost shows a clear minimum in textured regions. Figure adapted from [Newcombe et al., 2011].

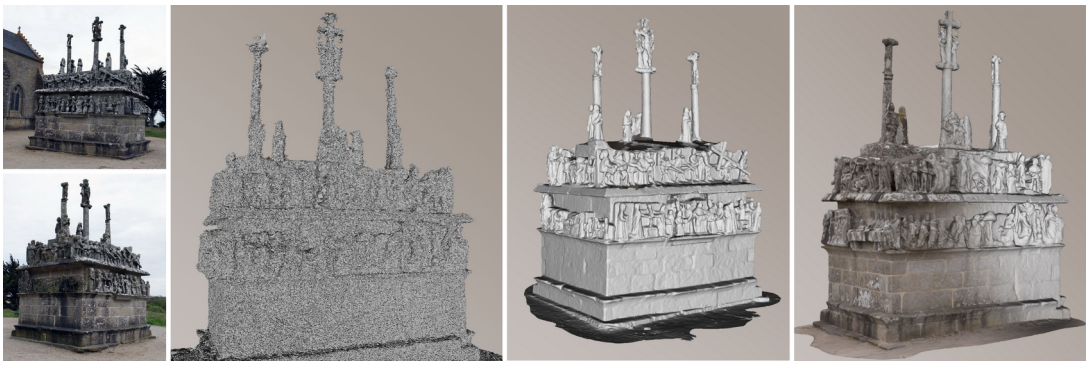

Figure 2.20: From left to right: Two of the input images, the intial visibility-consistent mesh, the refined final reconstruction, and the texture mapped reconstruction. Figure adapted from [Vu et al., 2009].

Tetrahedralization and extracing the surface between inside/outside labelled tetrahedra [Labatut et al., 2007]. The resulting surface interpolates the initial noisy point cloud and is refined [Faugeras and Keriven, 2006] in a variational multi-view stereo framework by means of a gradient descent energy minimization. See Figure 2.20.

2.3.4 Scene Representation

MVS approaches use different scene representations, and it is often desirable to compute a globally consistent surface representation from the Multi-View Stereo output. Some algorithms directly produce a surface mesh [Hernandez and Schmitt, 2004; Vu et al., 2009, 2012] and use mesh evolution approaches, which are difficult to implement. Many approaches use discrete volumes that encode either some kind of occupancy function or level-sets, such as the signed distance function. A surface can then easily be obtained using the Marching Cubes algorithm [Lorensen and Cline, 1987], or variants thereof [Kazhdan et al., 2007]. Uniform representations do not scale well to large or high-resolution datasets and hierarchical space partitioning is often employed. Furukawa and Ponce [2010] use a patch-based representation, which is often treated as a simple point cloud for further processing. This representation is convenient to handle and efficient to store, however, neighborhood queries are difficult to

implement. Producing depth maps for each input image [Goesele et al., 2006, 2007] is a local and memory-efficient approach, but causes vast computational overhead as overlapping surface parts are computed multiple times. For both, points and depth maps, the construction of a final surface is a challenging problem.

2.4 Surface Reconstruction

The problem addressed by surface reconstruction approaches is to recover the unknown original surface of a shape from imperfect measurements of a surface. These measurements almost always take the form of a point cloud, sometimes with connectivity information. The sampling of the original surface is at discrete positions and intermediate positions are a priori unknown. Furthermore, the samples are subject to various imperfections such as noise and outliers, and the formation of the samples is usually unknown or not well understood.

Reconstructing surfaces from unorganized points, overlapping range scans and depth maps are important problems in computer graphics and occur in various application domains. Geometry acquisition appears in cultural heritage for the purpose of digitally preserving artifacts of cultural value. Mesh-based representations of urban street-level geometry and famous landmarks recently gained significant interest for creating 3D maps of our world. Digitized real-world objects are also commonly used as blueprints for modeling assets for movies and games.

2.4.1 Surface Representation

The goal of surface reconstruction is to produce an explicit surface representation. Because surfaces must be discretized in one form or the other, we usually consider polygonal surface meshes. The most common special case of a polygonal mesh is the triangle mesh $M = (V, T)$, which consists of vertices $\mathbf{v}_i \in V$ and triangular polygons $t_i \in T$ spanned by three vertices each. The edges $e_i \in E$ between two vertices are induced by the triangulation. Such a surface mesh M defines a two-dimensional subspace $\{\mathbf{x} \mid \mathbf{x} \in M\} \subset \mathbb{R}^3$ embedded in 3D space. The triangle mesh itself is a piecewise linear polynomial approximation to the real surface, but others are possible [Vlachos et al., 2001]. The triangle is a 2-simplex and thus the simplest choice for a planar approximation, because all three vertices of a triangle are guaranteed to lie on a plane. More general polygonizations (such as quadrangulations) are less often used and will not be discussed here.

Since we usually seek to generate digital representations of real-world objects, the resulting triangulations are subject to constraints. In particular, real-word objects are closed, two-dimensional manifolds, or just 2-manifold. As an exception to the rule, we also allow *open* 2-manifolds with boundaries, which is convenient in open scenes that cannot be reconstructed in a closed manner. A 2-manifold is a 2D topological space which resembles a 2D Euclidean space in a finite region around every point. A triangle mesh M is clearly a 2-manifold inside each triangle t_i, but may not be at vertices v_i and edges e_i. A triangle mesh M is a 2-manifold if and only

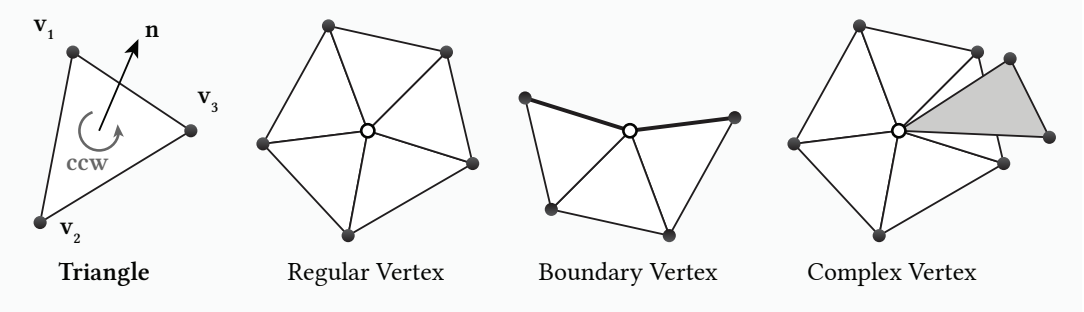

Figure 2.21: Triangles are defined in counter-clockwise order (ccw) and the normal $\mathbf{n}$ is defined as $\mathbf{n} = (\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)$. Each vertex can be classified into a *regular*, *border*, *complex*, and *unreferenced*. Unreferenced vertices are not part of any triangle.

if all triangles adjacent to a vertex v_i form a (possibly open) *triangle fan*. This also implies that every edge e_i has exactly one or two adjacent triangles. The set of all edges with only one adjacent triangle represents the *boundary* of the manifold. See Figure 2.21 for an illustration.

2.4.2 Reconstruction Input

Surface reconstruction approaches can broadly be classified into *point-based* and *depth map fusion* approaches, see Figure 2.22. Where point based approaches take as input a set of points with additional attributes, depth map fusion approaches process triangulated depth maps and usually require the scanner position, and are thus less generic. Typical surface reconstruction algorithms take some of the following attributes as input:

- Surface sample positions $P = \{p_1, \ldots, p_m\} \subset \mathbb{R}^3$
- Surface normals $N = \{n_1, \ldots, n_m\} \subset \mathbb{R}^3$ s.t. $\|n_i\|_2 = 1$
- Confidence values $C = \{c_1, \ldots, c_m\} \subset [0, 1]$
- Scale values $S = \{s_1, \ldots, s_m\} \subset \mathbb{R}$

Few algorithms solely take unorganized points P as input [Hoppe et al., 1992; Alliez et al., 2007]. However, a normal estimation is part of these methods, and the core reconstruction approach makes use of normals. Oriented points are used by RBF-based methods [Carr et al., 2001], the variational method by Calakli and Taubin [2011] and Kazhdan's work based on the characteristic function [Kazhdan, 2005]. Various approaches allow specifying a confidence value per surface point [Kazhdan et al., 2006; Klowsky et al., 2012; Kazhdan and Hoppe, 2013; Fuhrmann and Goesele, 2014], which is useful if uncertainty information is provided by the acquisition system. Recently, the use of additional per-vertex scale values S proved to be a useful information for less controlled data and in the presence of surface samples at different scales [Klowsky et al., 2012; Fuhrmann and Goesele, 2011, 2014].

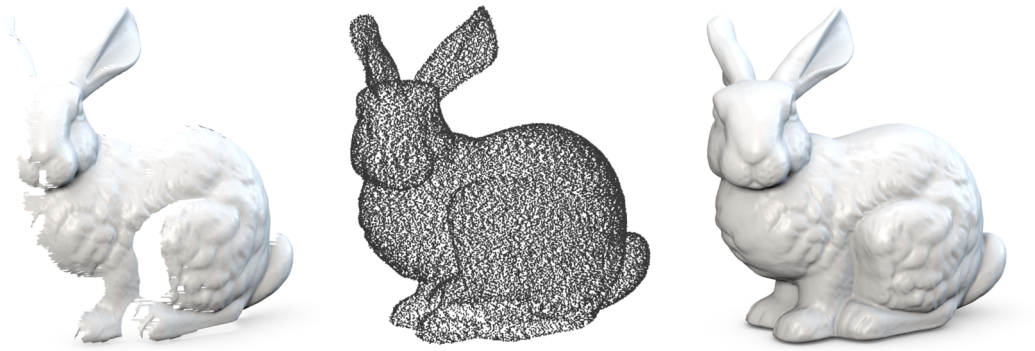

Figure 2.22: *Left:* A single triangulated range image only covers parts of the surface and multiple range images need to be fused to model an object. *Middle:* Surface samples are used as input for point-based reconstruction algorithms. *Right:* A reconstructed surface mesh.

Another more structured type of input is *range scanner data*, or *depth maps*. Here, the samples are essentially stored in a rectangular grid where each element measures the distance from the sensor position to the surface. Having direct access to this input has several advantages: Because the samples are structured in a grid, connectivity information between the samples can be inferred and the depth maps can be triangulated. Normals N for the samples P can be computed from the triangulation. Curless and Levoy [1996] describe a simple method to generate confidence values C from triangulated scans, which improves reconstruction quality. Additionally, since the size of a sample can be inferred from neighboring samples, scale values S can easily be computed [Fuhrmann and Goesele, 2014]. This type of input can therefore be converted to point samples with a rich set of attributes.

Some algorithms directly take range scanner data as input for *depth map fusion*, or *range image integration*. They usually rely on the position of the sensor as additional information. Examples include the classical and popular *VRIP* method by Curless and Levoy [1996], range image integration by energy minimization [Zach et al., 2007], an improved version of VRIP [Vrubel et al., 2009], and a multi-scale approach for depth map fusion [Fuhrmann and Goesele, 2011]. All of these methods fuse depth maps by averaging individual signed distance fields in a volumetric representation. A notable exception are the surface-based approaches such as *Mesh Zippering* [Turk and Levoy, 1994], which are, however, unreliable in the presence of noise and outliers.

It is debatable whether it is useful to strip away the connectivity information from the triangulated depth maps and process the resulting point cloud with point-based approaches. Clearly, algorithms specifically designed for depth maps are less versatile because they rely on additional input, such as the sensor position, which might not be available for the dataset at hand. In this thesis we will demonstrate that state-of-the-art techniques for depth map fusion do not, and likely will not in the near future, outperform the more general point based approaches. The reason lies in the demand to process increasingly more uncontrolled input that is contaminated

with various artifacts. This demand is driven by the growing number of commodity sensors, such as consumer depth cameras (e.g., the *Kinect*) or passive image-based reconstruction systems (e.g., smart phone cameras). These sensors produce data with various properties including non-uniform sampling rate, noise, outliers, misalignments, missing data, and multi-scale samples.

2.4.3 Implicit Function

Few methods directly merge surface meshes [Turk and Levoy, 1994; Soucy and Laurendeau, 1995] and only work on very clean input data. Point interpolation techniques [Bernardini et al., 1999; Edelsbrunner and Mücke, 1994] use (a subset of) the original input points and reconstruct the missing connectivity information, hence lack approximation capabilities and are thus unsuitable for noisy input. The vast majority of surface reconstruction methods construct an implicit function from which the surface is extracted afterwards. The reason is the fundamental difficulty in directly working with the explicit representation. Instead of explicitly handling geometric and topological changes by evolving the surface, an implicit function implicitly handles these changes.

An explicit function $G\colon \mathbb{R}^2 \mapsto \mathbb{R}$ defines an explicit surface embedded in 3D space. For example, consider a height-field approach, $G(x, y) = z$, which is parameterized over the (x, y)-plane and yields a height value z. The surface is thus defined by $\{(x, y, G(x, y))^\top\}$. In contrast, an implicit function $F\colon \mathbb{R}^3 \mapsto \mathbb{R}$ is parametrized over 3D-space and yields a scalar whose interpretation is application-specific. An implicit surface, or *isosurface* with respect to an *isovalue* c is then given by the c-level set of F, i.e., $\{\mathbf{x} \mid F(\mathbf{x}) = c\}$.

Consider the intuitive example of the *signed distance function* $F(\mathbf{x})$, which defines for every point in space $\mathbf{x} \in \mathbb{R}^3$ the signed distance to the surface. The isosurface of interest is then given by the zero-level set of F: $\{\mathbf{x} \mid F(\mathbf{x}) = 0\}$. The sign of the distance values indicates whether a point is *in front of* or *behind* the surface and defines the surface orientation.

Consider the surface of a sphere as an example. Such a shape is conveniently expressed using an isosurface $\{\mathbf{x} \mid \sqrt{\mathbf{x} \cdot \mathbf{x}} = r\}$ with the radius r of the sphere as isovalue. Here, the surface corresponds to the r-level set of the implicit function.

The use of an implicit function for surface reconstruction was pioneered by Hoppe et al. [1992]. Given points $\mathbf{p}_i$ with normals $\mathbf{n}_i$, a set of tangent planes T_i with centers $\hat{\mathbf{p}}_i$ and normals $\hat{\mathbf{n}}_i$ are associated with every input point $\mathbf{p}_i$. The implicit function is then defined by

$$F(\mathbf{x}) = (\mathbf{x} - \hat{\mathbf{p}}_i) \cdot \hat{\mathbf{n}}_i. \tag{2.26}$$

This formulation computes the orthogonal distance from $\mathbf{x}$ to plane i whose center $\hat{\mathbf{p}}_i$ is closest to $\mathbf{x}$. Because small changes in the query $\mathbf{x}$ can result in a different nearest plane center, the implicit function is clearly discontinuous which can lead to reconstruction artifacts. Notably, this implicit function is defined everywhere without the need for interpolation, which stands in contrast to many approaches which discretize the implicit function.

A discretized implicit function has been successfully used for range image integration by Curless and Levoy [1996] in a method called *VRIP*. It is based on a volumetric approximation of the signed distance function. The approach is different in that the implicit function F is constructed as the weighted sum of individual signed distance functions d_i and weight functions w_i, each generated by one range image. The total implicit function is given by

$$F(\mathbf{x}) = \frac{\sum_i w_i(\mathbf{x}) d_i(\mathbf{x})}{\sum_i w_i(\mathbf{x})} \tag{2.27}$$

$$W(\mathbf{x}) = \sum_i w_i(\mathbf{x}). \tag{2.28}$$

A discretization of the function is motivated by the ability to *incrementally update* the implicit function, by adding one range image at a time. Although this function is continuous because both d_i and w_i are continuous, w_i has compact support which leads to undefined regions of F with zero weight. In this regard the method presented in Chapter 5 shares many similarities with VRIP.

A variational formulation is presented by Calakli and Taubin [2011]. They reconstruct an approximation to the signed distance function F by minimizing an energy functional $\mathcal{E}(F)$ which drives F to be zero at the sample positions, and the derivative ∇F to coincide with the sample normals. A well-conditioned solution is found by regularization of the second order derivatives and forcing the norm of the Hessian matrix $\mathtt{H}(F)$ to be close to zero.

Most approaches reconstruct the signed distance function F and extract the isosurface as the zero-level set of F. In contrast, Kazhdan et al. approximate the *indicator function*, which is one inside, and zero outside the solid. Using a Monto-Carlo approximation to Stokes' Theorem they compute the Fourier coefficients from oriented surface samples and solve for the surface using the inverse Fast Fourier Transformation [Kazhdan, 2005]. In a more recent method, Kazhdan et al. formulate the reconstruction of the indicator function as a Poisson problem [Kazhdan et al., 2006; Kazhdan and Hoppe, 2013], which translates into solving a sparse linear system of equations. At least in theory, since the indicator function is discontinuous at the surface, finding an isovalue for surface extraction is an ill-posed problem. In practice, the implicit function is evaluated at the input samples to find a suitable isovalue.

2.4.4 Isosurface Extraction

Given an implicit function F, the isosurface corresponding to the c-level set of F can be extracted using various algorithms. This process is often called *contouring* the volumetric data into a *polygonal* representation. In the case of a regular sampling of the implicit function, *Marching Cubes* [Lorensen and Cline, 1987] is probably the most well-known algorithm. The algorithm inspects eight voxels v_i in "cube configuration" and determines for each voxel whether it is inside the surface ($v_i < 0$) or outside the surface ($v_i \geq 0$). Because there are 8 voxels in each cube with 2 states (inside or outside), there are $2^8 = 256$ cube configurations which can be enumerated in a look-up table. Each configuration corresponds to a triangulation within

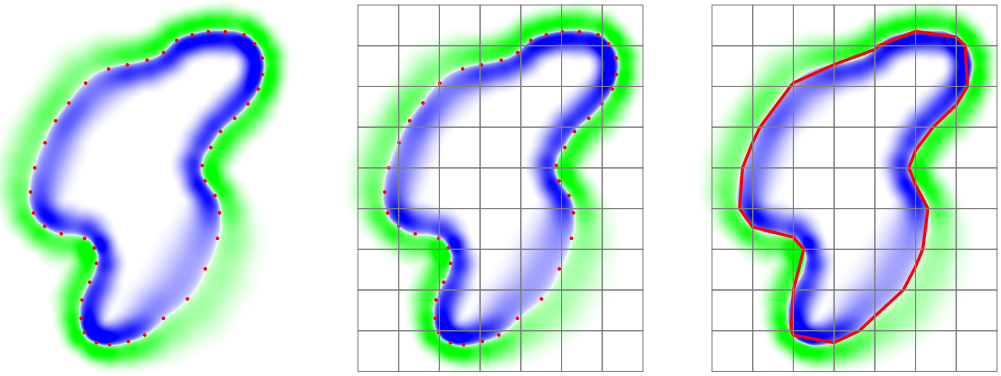

Figure 2.23: *Left:* An implicit function reconstructed from the input samples. *Middle:* The implicit function is regularly sampled for isosurface extraction. *Right:* The resulting piecewise linear isosurface is extracted with *Marching Squares* (the 2D version of Marching Cubes).

the cube and the union of all per-cube triangulations yields the final surface. The resulting surface is guaranteed to be a valid 2-manifold, possibly with boundaries. The surface is also watertight if and only if the value of all voxels at the volume's boundary are on either side of c. See Figure 2.23 for a 2D surface reconstruction example using the Marching Squares algorithm.

In a regular lattice, Marching Cubes can process each cube individually. In the case of an octree, however, different decisions are made on either side of a cube face (because of depth disparity in the octree), which leaves cracks in the surface. A standard solution is to perform conforming splits [Westermann et al., 1999] by decomposing coarser nodes into tetrahedral elements until the depth disparities are resolved. This, however, results in a large increase of triangles. Many approaches focus on adapting the octree hierarchy, e.g. by restricting the depth disparity to one, and patching cracks between nodes at different depths [Westermann et al., 1999]. In the dual setting, isovertex positions are defined inside the nodes and connected to isovertices in adjacent nodes [Ju et al., 2002; Schaefer and Warren, 2005], which can lead to non-manifold surfaces. A solution to this problem is presented by Kazhdan et al. [2007]. This technique yields a crack-free and highly adaptive mesh directly from the octree hierarchy and is applicable to many hierarchical surface reconstruction techniques [Kazhdan et al., 2006; Calakli and Taubin, 2011; Kazhdan and Hoppe, 2013; Fuhrmann and Goesele, 2014].

2.4.5 Sample Scale and Implications for Reconstruction

The vast majority of the surface reconstruction algorithms idealize the input samples as infinitely small points. In fact, every sample is the result of an integration process over a certain surface area. See Figure 2.24 for an illustration of sample scale. The specific properties of the sample scale depend on the scanning technology and the sample formation process. Let us consider a few examples.

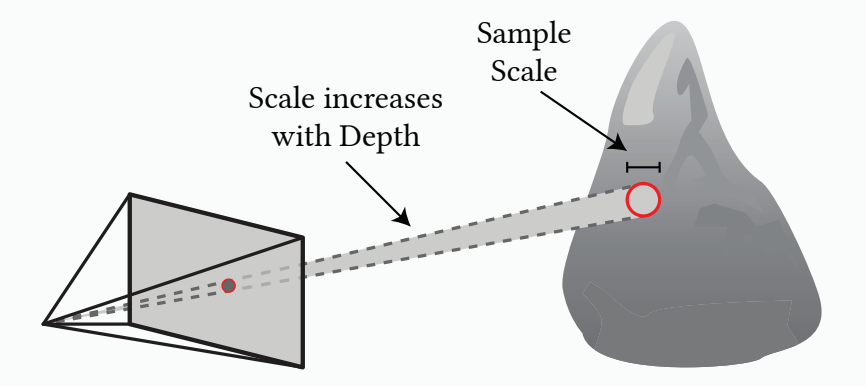

Figure 2.24: Illustration of a sample in a depth map, projected into 3D space onto the surface. The actual size of the sample depends on the acquisition process.

- Laser triangulation scanners project a laser point on the 3D object and determine the 3D position of the laser point by triangulation using a visual sensor. Here, the laser point has a certain extent on the 3D object as the beam diameter is not infinitely small. Furthermore, the visual sensor uses a (usually unknown) point spread function, which effectively blurs the incoming light information.

- Structured light scanners project a pattern on the 3D object. Depending on the projector resolution and potentially out-of-focus effects, a single pixel leads to samples with a certain extent on the 3D object. The pattern is interpreted by a visual sensor with a point spread function, which again blurs the visual information, thus pixels are not infinitesimal "point samples".

- Mid- and long-range scanners use laser light pulses and measure the time until the reflection of the pulse arrives back at the sensor. These time-of-flight systems also accumulate light information over the area of the laser point. Furthermore, since the sample spacing in these systems is usually much larger than the sample size aliasing artifacts can occur.

- Multi-View Stereo systems triangulate visual correspondences. The visual information in the images has been filtered by a point spread function and is subject to sensor discretization, so even accurate correspondences are based on filtered visual information. Additionally, the majority of MVS techniques establish correspondences using patches, which have an additional low-pass filtering effect.

Treating point samples as idealized, infinitely small samples discards valuable information about the size (or *scale*) of a sample. This can have a negative effect on the reconstruction and often leads to dramatic differences in quality. Obtaining a very reasonable approximation of a sample's scale is surprisingly simple: In most cases this does not require internal knowledge of the scanning technology used. As

described in Section 2.4.2, scale information is directly available from depth maps and scans in triangulated form. The approximation of scale can often be improved by using information about the scanning system. For example, compensating for the patch size in Multi-View Stereo can be achieved by multiplying the scale value with the radius of the patch.

The effects of disregarding sample scale are three-fold: First, depending on the scale of the sample, the 3D position of the sample corresponds to a low-pass filtered, smoothed version of the surface. Hence, the reconstruction algorithm tends to reconstruct the filtered version of the original surface [Klowsky et al., 2012]. Second, although the true position of a sample is unknown and can likely not be recovered, scale information indicates which samples should not be combined with each other in order to avoid degrading higher resolution geometry with low-resolution information [Mücke et al., 2011; Fuhrmann and Goesele, 2011]. Third and most importantly, without scale it is impossible to distinguish between actual detail and redundancy in the samples [Fuhrmann and Goesele, 2014]. A very dense set of samples can either be interpreted as a very detailed region at a high sampling rate, or a region sampled with many but redundant low-resolution samples. The former case justifies a high-resolution reconstruction, while the latter case suggests to use the redundancy for noise reduction. Without scale, these decisions need to be made globally by means of setting parameters.

In Chapter 4 and 5 we present surface reconstruction algorithms which incorporate scale information in the reconstruction. Redundancy can automatically be detected, and an appropriate reconstruction behaviour can automatically be determined from the sample scale without setting any parameters.

2.5 Post-Processing

In this section we cover some post-processing steps that can be applied to the global surface mesh. In particular, the surface triangulation contains many small and ill-shaped triangles, which is a typical artifact caused by the Marching Cubes algorithm and variants thereof. The resolution of the surface is often adapted to the reconstruction scale, and not to the geometric properties of the surface, which results in many small triangles even in planar regions. A global remeshing step can improve the distribution and shape of the triangles and significantly reduce the amount of triangles at a negligible loss of quality. Finally, a visually faithful geometric representation is achieved by texturing the surface using high-resolution image information.

2.5.1 Mesh Cleanup

Because the Marching Cubes algorithm operates independently within each cube, it introduces many unnecessarily small and degenerated triangles. Degenerated triangles contains at least one very small angle and can be classified in two types: *Caps* and *needles* [Botsch and Kobbelt, 2001], see Figure 2.25. These triangles barely contribute any geometric information because their surface area is close to zero, and

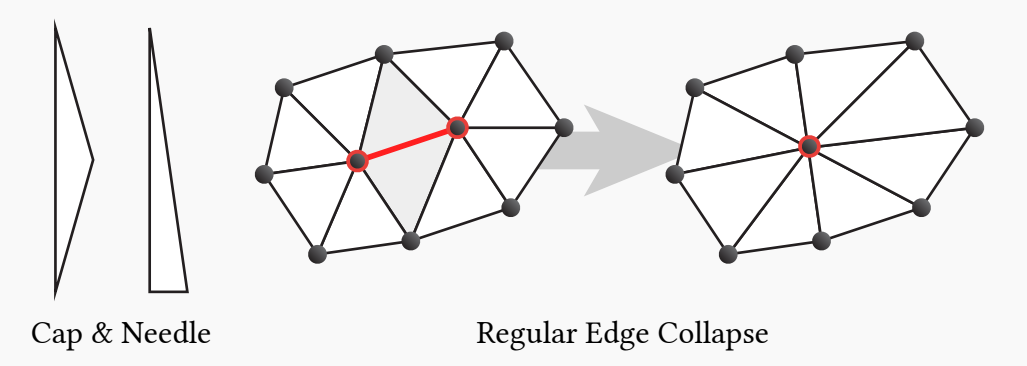

Figure 2.25: *Left:* Two types of ill-shaped triangles. *Caps* have two small angles and no short edges, and *Needles* have one small angle and two long edges. *Right:* The *edge collapse* is a fundamental operation in mesh processing. A regular edge collapse removes one edge, one vertex and two triangles from the mesh. Collapsing a boundary edge removes one edge, one vertex and one triangle only.

they should be removed to reduce memory consumption, to increase rendering performance and to avoid numerical problems when processing the mesh.

Edge Collapse Operation

A very common mesh decimation operation is the *edge collapse* primitive, see Figure 2.25. In order to avoid topological changes by the collapse, it is useful to ensure that the geometry does not change too much. To this end, the original normals $\mathbf{n}_i$ of all triangles adjacent to v_1 and v_2 (excluding the triangles which are to be removed) are compared with the normals of the new triangles $\tilde{\mathbf{n}}_i$ after to collapse. If the angle between the normals is above a threshold t_{angle}, no collapse should be performed. This can be expressed in terms of the dot product between the normals

$$\mathbf{n}_i \cdot \tilde{\mathbf{n}}_i < \cos(t_{\text{angle}}). \tag{2.29}$$

A useful threshold is $\cos(t_{\text{angle}}) = 0.95$, which corresponds to $t_{\text{angle}} \approx 18.2°$.

Removing Marching Cubes Artifacts

While needle triangles can be eliminated by collapsing their short edge, removing cap triangles turns out to be a much harder problem. Collapsing any of their edges or subdividing the long edge can produce new cap triangles. A procedure for removing degenerated triangles is presented by Botsch and Kobbelt [2001]. In their approach, a global mesh slicing operation is used. However, a much simpler procedure can be applied that eliminates most degenerated triangles in a typical Marching Cubes mesh and usually reduces the number of triangles in the mesh by about 40%.

In the first step, needle triangles are removed by collapsing their short edge. In the second step, vertices with exactly three adjacent triangles are removed. This particular configuration is the main source for cap triangles and is easily removed

in a local manner. Finally, another pass that removes needle triangles is performed, as new needles may be introduced by the previous operation. This procedure is illustrated in Figure 5.9 in Chapter 5.

Needle triangles can be identified using a threshold t_{needle}, which is the ratio of the shortest edge $\mathbf{e}_{\text{shortest}}$ to the second shortest edge $\mathbf{e}_{\text{middle}}$ of the triangle

$$t_{\text{needle}} = \frac{\|\mathbf{e}_{\text{shortest}}\|_2}{\|\mathbf{e}_{\text{middle}}\|_2}. \tag{2.30}$$

For both equilateral triangles and Caps this ratio is close to 1 and decreases with increasing amount of degeneration. A threshold value of $t_{\text{needle}} \leq 0.4$ has been used for all of our results.

2.5.2 Surface Remeshing

The mesh cleanup procedure removes a large number of degenerate triangles from the mesh simply by considering trigonometric properties of the triangles. Surface remeshing, on the other hand, globally creates a new surface discretization which improves the triangle quality while maintaining the fidelity to the original surface. Figure 2.26 shows the original and the remeshed version of the *Mannequin* model. In this atypical example the number of vertices has been increased using subdivision techniques [Loop, 1987] to create a much smoother surface.

Usually, surface remeshing is concerned with triangle quality and global vertex distribution. Locally, the shape of the triangles is to be improved to obtain well-shaped, equilateral triangles. It has been shown that the (weighted) Centroidal Voronoi Tessellation produces an isotropic vertex distribution that leads to very regular triangles [Alliez et al., 2003; Surazhsky et al., 2003; Yan et al., 2009]. Globally, the vertex distribution is optimized either to be as uniform as possible, or according to geometric properties such as surface curvature. In the latter case, the aim is to distribute more vertices in regions of higher curvature to minimize the discretization error and to reduce rendering artifacts. Obtaining a good initial global vertex distribution is key to efficiency. A bad initial vertex distribution leads to very slow convergence behavior of the relaxation procedure that optimizes the vertex positions [Fuhrmann et al., 2010].

Triangle shapes are relevant for the numerical stability of mesh processing applications. On the other hand, a suitable vertex distribution can result in considerable vertex and triangle reduction. The vertex distribution in a mesh obtained by a surface reconstruction algorithm is often controlled by the input sample resolution or density. This can lead to over-tessellation in almost planar regions with low curvature, which can well be approximated with larger triangles while maintaining fidelity to the original mesh. Producing a mesh with fewer, larger elements is useful to reduce storage requirements, and to improve rendering performance.

A popular method to colorize triangle meshes is to use per-vertex colors. The color values are interpolated over the mesh triangles. Such a method relies on a dense vertex sampling in order to faithfully represent the object appearance. Most

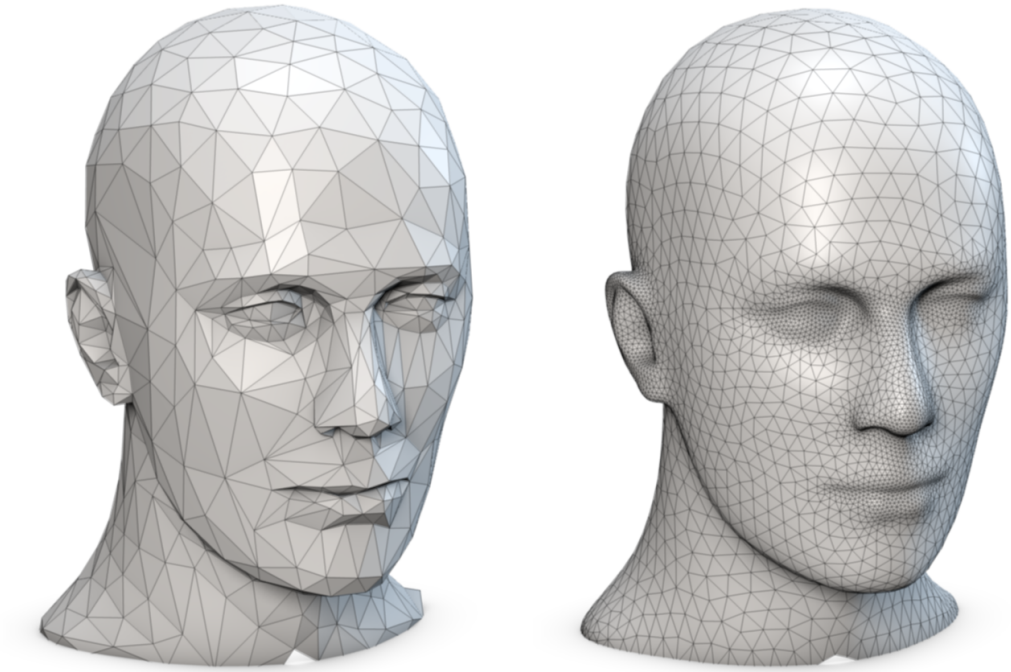

Figure 2.26: The original *Mannequin* model with 700 vertices and the remeshed subdivision surface with 5k vertices [Fuhrmann et al., 2010]. Note the regular shape of the triangles and the adaptive vertex sampling distributing more samples in regions of higher curvature.

remeshing and simplification algorithms only consider preservation of geometric quantities and will thus destroy per-vertex attributes. Few simplification methods attempt to preserve the attributes of meshes [Cignoni et al., 1998; Hoppe, 1999]. While it is certainly possible to incorporate other per-vertex attributes in computation of a *density field* which guides the global distribution of vertices during remeshing, it is appropriate to transfer these attributes to dedicated textures. Textures allow more control over the degree of detail while keeping the pure geometry as small as possible. Remeshing is the tool of choice to prepare the mesh and to provide a suitable input for surface texturing. The computation time of surface texturing falls significantly with fewer triangles and often leads to cleaner results.

2.5.3 Surface Texturing

Surface texturing can be seen as the last step in the reconstruction pipeline, which generates a surface with associated textures ready for rendering. The input to this process are the input images and the corresponding reconstructed camera parameters, as well as a globally consistent surface mesh. The textures are then generated by means of projecting the image information onto the geometry. The task of surface texturing is faced with several challenges. Because camera parameters can be

Figure 2.27: *Left:* The input geometry with the individual regions each colored by one view, the textured result before applying seam leveling, and the final textured result. *Right:* The final texture atlas produced by Waechter et al. [2014].

slightly inaccurate and the images are subject to different resolution, image blending leads to blurry results. A common approach is to select a single, suitable image for a surface region [Lempitsky and Ivanov, 2007], which causes seams at the region boundaries because of exposure and color differences in the images.

In order to obtain a sharp and high-resolution texture, a two-step approach is usually applied [Arikan et al., 2014]. The first step is a *view selection* that finds suitable high-resolution, in-focus images that orthogonally observe the surface to texture the surface regions. Also, in order to minimize the number of seams across the mesh, it is favorable to maximize the size of the region that is textured from one image. In the second step the remaining seams between the regions are eliminated by employing a *seam leveling*, or color adjustment algorithm. Both steps can be solved by minimizing a non-linear optimization problem [Waechter et al., 2014]. Figure 2.27 illustrates the view selection, seam leveling, and the resulting texture atlas. In contrast, the Floating Texture approach [Eisemann et al., 2008] repositions textures during runtime to compensate for inaccurate camera calibration and to obtain crisp textures, and thus avoids costly pre-processing.

2.6 Conclusion

In this chapter we discussed the background of image-based reconstruction. In particular, the pinhole camera (Section 2.1) has been introduced as the prevailing camera model. The extrinsic and intrinsic parameters of this camera model are recovered using Structure from Motion (Section 2.2), either with incremental or global techniques. A dense scene reconstruction is computed with Multi-View Stereo (Section 2.3) by means of selecting suitable neighboring views, and estimating geometry by establishing dense correspondences. Finally, a global surface is computed (Section 2.4), cleaned and textured (Section 2.5) to obtain a mesh for presentation purposes.

Chapter 3

MVE – The Multi-View Environment

Abstract

We present MVE, the Multi-View Environment. MVE is an end-to-end multi-view geometry reconstruction software which takes photos of a scene as input and produces a surface triangle mesh as the result. The system covers a Structure from Motion algorithm, Multi-View Stereo reconstruction, generation of extremely dense point clouds, and reconstruction of surfaces from point clouds. In contrast to most image-based geometry reconstruction approaches, our system is focused on reconstruction of multi-scale scenes, an important aspect in many areas such as cultural heritage. It allows to reconstruct large datasets containing some detailed regions with much higher resolution than the rest of the scene. Our system provides a graphical user interface for structure-from-motion reconstruction, visual inspection of images, depth maps, and rendering of scenes and meshes.

Contents

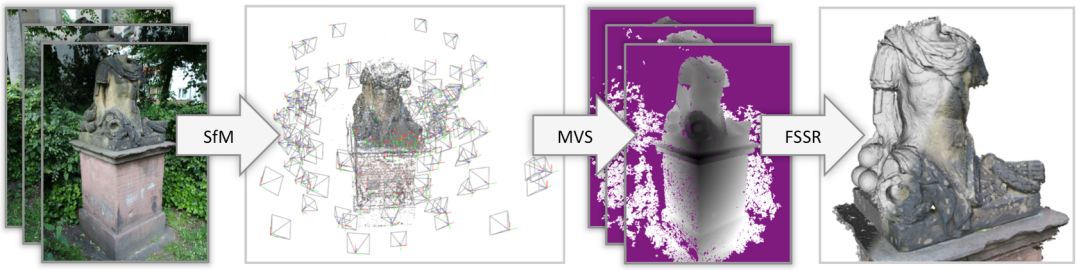

Figure 3.1: Our multi-view reconstruction pipeline. Starting from input images, Structure from Motion techniques are used to reconstruct camera parameters and a sparse set of points. Depth maps are computed for every image using Multi-View Stereo. Finally, a colored mesh is extracted from the union of all depth maps using a surface reconstruction approach.

3.1 Introduction

Acquiring 3D geometric data from natural and man-made objects or scenes is a fundamental field of research in computer vision and graphics. 3D digitization is relevant for designers, the entertainment industry, and for the preservation as well as digital distribution of cultural heritage objects and sites. In this chapter, we introduce *MVE*, the *Multi-View Environment*, a new, free software solution for low-cost geometry acquisition from images. The system takes as input a set of photos and provides the algorithmic steps necessary to obtain a high-quality surface mesh of the captured object as final output. This includes Structure from Motion (SfM), Multi-View Stereo (MVS) and surface reconstruction.

Geometric acquisition approaches are broadly classified into active and passive scanning. Active scanning technologies for 3D data acquisition exist in various flavors. Time of flight and structured light scanners are known to produce geometry with remarkable detail and accuracy. But these systems require special hardware and the elaborate capture setup is expensive. Real-time scanning systems such as the Kinect primarily exist for the purpose of gaming, but are often used for real-time geometry acquisition. These systems are based on structured light which is emitted into the scene. They are often of moderate quality and limited to indoor settings because of interference with sunlight. Finally, there is some concern that active systems may damage objects of cultural value due to intense light emission.

Passive scanning systems do not emit light, are purely based on natural illumination, and will not physically harm the subject matter. The main advantage of these systems is the cheap capture setup which does not require special hardware: A consumer-grade camera (or just a smartphone) is usually enough to capture datasets. These systems are based on finding visual correspondences in the input images; thus the geometry is usually less complete, and scenes are limited to static, well-textured surfaces. The inexpensive demands on the capture setup, however, come at the cost of much more elaborate computer software to process the unstructured input. The

standard pipeline for geometry reconstruction from images involves three major algorithmic steps:

- Structure from Motion (SfM) reconstructs the extrinsic camera parameters (position and orientation) and the camera calibration data (focal length and radial distortion) by finding sparse but stable correspondences between images. A sparse point-based 3D representation of the subject is created as a byproduct of camera reconstruction.
- Multi-View Stereo (MVS), which reconstructs dense 3D geometry by finding visual correspondences in the images using the estimated camera parameters. These correspondences are triangulated yielding 3D information.
- Surface Reconstruction, which takes as input a dense point cloud or individual depth maps, and produces a globally consistent surface mesh.

It is not surprising that software solutions for end-to-end passive geometry reconstruction are rare. The reason lies in the technical complexity and the effort required to create such integrated tools. Many projects cover parts of the pipeline, such as *VisualSfM* [Wu, 2013] or *Bundler* [Snavely et al., 2006] for Structure from Motion, *PMVS* [Furukawa and Ponce, 2010] for Multi-View Stereo, and *Poisson Surface Reconstruction* [Kazhdan and Hoppe, 2013] for mesh reconstruction. A few commercial software projects offer an end-to-end pipeline covering SfM, MVS, Surface Reconstruction and Texturing. This includes *Arc3D*, *Agisoft Photoscan* and *Acute3D Smart3DCapture*. We offer a complete pipeline as an open source software system free for personal use and research purposes.

Our system gracefully handles many kinds of scenes, such as closed objects or open outdoor scenes. It avoids, however, filling holes in regions with insufficient data for a reliable reconstruction. This may leave gaps in models but does not introduce artificial geometry, common to many global reconstruction approaches. Our software puts a special emphasis on multi-resolution datasets which contain both detailed and less detailed regions, and it has been shown that inferior results are produced if the multi-resolution nature of the input data is not considered properly [Klowsky et al., 2012; Fuhrmann and Goesele, 2011, 2014].

In the remainder of this chapter, we will first give an overview of our system (Section 3.2). The practical applicability of our system is demonstrated in a hands-on guide in Section 3.3. We then show reconstruction results on several datasets in Section 3.4 and demonstrate the superiority of our pipeline in comparison to alternative state of the art software. We briefly describe our software framework in Section 3.5 and finally conclude in Section 3.6.

3.2 System Overview

Our system consists of three main steps, namely Structure from Motion (SfM) which reconstructs the parameters of the cameras, Multi-View Stereo (MVS) for establishing dense visual correspondences, and a final meshing step which merges the MVS

geometry into a globally consistent, colored mesh. In the following, we give a concise overview of the process, using the dataset of *Anton's Memorial* as an example for a cultural heritage artifact, see Figure 3.1. For a more detailed explanation of the approaches, we refer the interested reader to Szeliski's textbook [Szeliski, 2010].

3.2.1 Structure from Motion

Structure from Motion is one of the crowning achievements of photogrammetry and computer vision which started its roots in [Armstrong et al., 1994; Hartley, 1994; Pollefeys et al., 1998] and opened up to a wider audience in [Pollefeys et al., 2003; Snavely et al., 2006]. In essence, SfM reconstructs the parameters of cameras solely from sparse correspondences in an otherwise unstructured image collection. The recovered camera parameters consist of the extrinsic calibration (i.e. the orientation and position of the camera), and the intrinsic calibration (i.e. the focal length and radial distortion of the lens). Although, at least in theory, the focal length can be fully recovered from the images for non-degenerate configurations, this process can be unstable. However, a good initial guess for the focal length is usually sufficient and can be optimized further.

Feature detection: First, machine-recognizable features are detected in the input images (Figure 3.2, left) and matched in order to establish sparse correspondences between images. Differences in the images require invariance of the features with respect to certain transformations, such as image scale, rotation, noise and illumination changes. Our system implements and jointly uses both *SIFT* [Lowe, 2004] and *SURF* [Bay et al., 2008] features which are amongst the top performing features in the literature.

Feature matching: The detected features are matched between pairs of images (Figure 3.2, right). Because corresponding points in two images are subject to the epipolar constraints of a perspective camera model [Luong and Faugeras, 1995], enforcing these constraints removes many false correspondences. The pairwise matching results are then combined and expanded over several views, yielding feature tracks. Each track corresponds to a single 3D point after SfM. Depending on the size of the scene, matching can take a long time, because every image is matched to all other images, resulting in a quadratic algorithm. As an expedient, the state after matching (containing the feature detections and the pairwise matching) can be saved to file and reloaded later in case the remaining procedure is to be repeated under different parameters. This state is called the *prebundle*.

We also investigated in accelerating the matching time of our system. Common approaches include matching fewer features per image, reducing the number of pairs in a pre-processing step, or accelerating the matching itself using parallelism. We use a practical combination of the approaches: By matching a few low-resolution features, one can identify image pairs that potentially do not match, and reject the candidates before full-resolution matching is performed. Although low-resolution

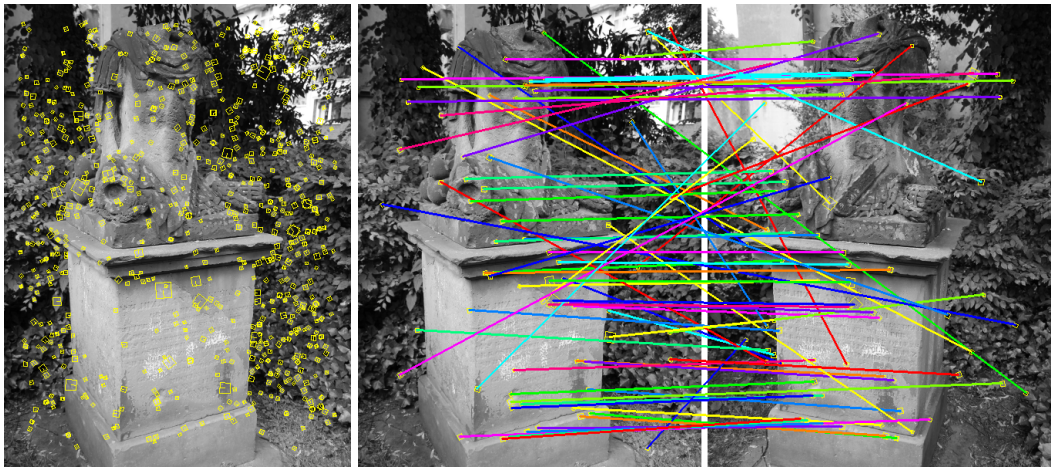

Figure 3.2: Feature detection (left) and feature matching between two views (right). The horizontal lines are mostly good matches, and more slanted lines are outliers. Enforcing two-view constraints will remove most outliers.

matching rejects some good image pairs, we could not observe any loss of quality in the reconstruction. It has been shown by Wu [2013] that this can considerably accelerate the matching time.

Incremental reconstruction: The relative pose of a good initial image pair is estimated, and all feature tracks visible in both images are triangulated. What follows is the incremental SfM, where suitable next views are incrementally added to the reconstruction, until all reconstructable views are part of the scene (Figure 3.3). Lens distortion parameters are estimated during reconstruction. The performance of subsequent algorithms is considerably improved by removing the distortion from the original images.

Not many software solutions for SfM have been published, probably because the theoretical concepts and algorithmic details are involved. Freely available software for this purpose includes *Bundler* [Snavely et al., 2006] and *VisualSfM* [Wu, 2013].

3.2.2 Multi-View Stereo

Once the camera parameters are known, dense geometry reconstruction is performed. MVS algorithms exist in various flavors [Seitz et al., 2006]. Some approaches work with volumetric representations [Kolev et al., 2012] and usually do not scale well to large datasets. Others reconstruct global point clouds, e.g. the popular *PMVS* implementation by Furukawa and Ponce [2010]. Scalability issues further motivated work that clusters the scene into smaller manageable pieces [Furukawa et al., 2010]. Although *PMVS* is widely used, we aim at creating much denser point clouds for mesh reconstruction in order to preserve more details in the final result. A third line of work directly reconstructs global meshes [Vu et al., 2012] and couples MVS and surface reconstruction approaches in a mesh evolution framework.

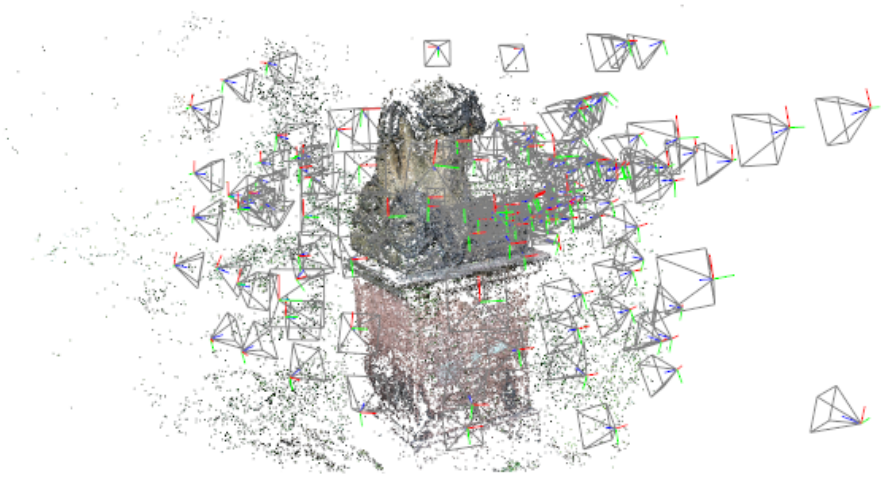

Figure 3.3: Structure from Motion reconstruction showing the sparse point cloud and the camera frusta.

We use the Multi-View Stereo for Community Photo Collections approach by Goesele et al. [2007] which reconstructs a depth map for every view (Figure 3.4). Although depth map based approaches produce lots of redundancy because many views are overlapping and see similar parts of the scene, it effortlessly scales to large scenes as only a small set of neighboring views is required for reconstruction. In a way, this can be seen as an out-of-core approach to MVS. Another advantage of depth maps as intermediate representation is that the geometry is parameterized in its natural domain, and per-view data (such as color) is directly available from the images. The excessive redundancy in the depth maps can be a burden; not so much in terms of storage but processing power required for depth maps computation. On the positive side, this approach has proven to be capable of producing highly detailed geometry, overcoming the noise in the individual depth maps [Fuhrmann and Goesele, 2011, 2014].

3.2.3 Geometry Reconstruction

Merging the individual depth maps into a single, globally consistent representation is a challenging problem. The input photos are usually subject to large variations in viewing parameters. For example, some photos show a broad overview of the scene while others show small surface details. The depth maps inherit these multi-scale properties which leads to vastly different sampling rates of the observed surfaces.

Many approaches for depth map fusion have been proposed. The pioneering work by Curless and Levoy [1996] renders locally supported signed distance fields (SDF) of the depth maps into a volumetric representation. Overlapping SDFs are averaged, which effectively reduces noise, but also quickly eliminates geometric details if depth maps with different resolution are merged. To this end, Fuhrmann and Goesele [2011] present a solution based on a hierarchical SDF which avoids averaging geometry at different resolutions. We use the follow-up work by Fuhrmann

Figure 3.4: An input image and the corresponding depth map reconstructed with multi-view stereo. Each depth value encodes the distance from the camera.

and Goesele [2014]. They present a point-based reconstruction approach (*Floating Scale Surface Reconstruction*, *FSSR*), which additionally takes per-sample scale values as input. In contrast to point-based approaches that do not use scale, such as *Poisson Surface Reconstruction* [Kazhdan and Hoppe, 2013], the method is able to automatically adapt the interpolation and approximation behavior depending on sample scale and redundancy without explicit parameter settings. An important aspect of *FSSR* is that it does not interpolate regions with insufficient geometric data. Instead, it leaves these regions empty which is useful for incomplete or open (outdoor) scenes. This stands in contrast to many global approaches that often hallucinate geometry, requiring manual cleanup.

In order to generate the input samples for *FSSR*, all depth maps are triangulated and colored using the input image. The connectivity information is used to compute a normal for each vertex. Additionally, the lengths of all edges emanating from a vertex are averaged and used as scale value for the vertex. The union of all vertices from all depth maps is then used as input to *FSSR*. An hierarchical, signed implicit function is constructed from the samples, and the final surface (Figure 3.5) is extracted as the zero-level set of the implicit function using a hierarchical variant of the *Marching Cubes* algorithm [Kazhdan et al., 2007].

3.3 Reconstruction Guide

We now demonstrate a practical walk-through of a reconstruction session starting with photo acquisition all the way to the final geometry. Specifically, we point out best practices and explain how to avoid common pitfalls.

Figure 3.5: The final surface reconstruction with color (left) and shaded (right). Notice how even the engraved text is visible in the geometry.

Capturing photos: A dataset reconstructs best if a few simple rules are observed. First, in order to successfully reconstruct a surface region, it must be seen from at least five views. This is a requirement of the MVS algorithm to reliably triangulate any 3D position. Photos should thus be taken with a good amount of overlap. Usually, unless the dataset becomes really large, more photos will not hurt quality, but there is a tradeoff between quality and performance. As a rule of thumb, taking twice as many photos as one might think is a good idea. In order for triangulation to work, parallax is required. The camera should be re-positioned for every photo. (This is exactly opposite to how panoramas are captured, where parallax in the images must be avoided.) This is also important for SfM: Triangulating a feature track with insufficient parallax results in a small triangulation angle and a poorly conditioned 3D position. Figure 3.6 shows some input images of our example dataset.

Figure 3.6: Five out of 161 input photos of the *Anton's Memorial* dataset.

Creating a scene: A view is a container that contains per-viewpoint data (such as images, depth maps and other data). A scene is a collection of views, which make up the dataset. A new scene is created using either the graphical interface of our software, *UMVE*, or the command line tool *makescene*. Technically, the scene appears as a directory in the file system (with the name of the dataset). It contains another directory `views/` with all views stored as files with the extension `.mve`. Creating a new scene will solely create the `views/` directory for now. Importing photos will create a `.mve` file for every photo. This process will also import meta information from the images (EXIF tags), which is required to get a focal length estimate for every photo. If EXIF tags are not available, a default focal length will be assumed. This, however, can lead to SfM failures if the default value is a bad guess for the actual focal length. Figure 3.7 shows our graphical interface *UMVE* after importing images into a new scene.

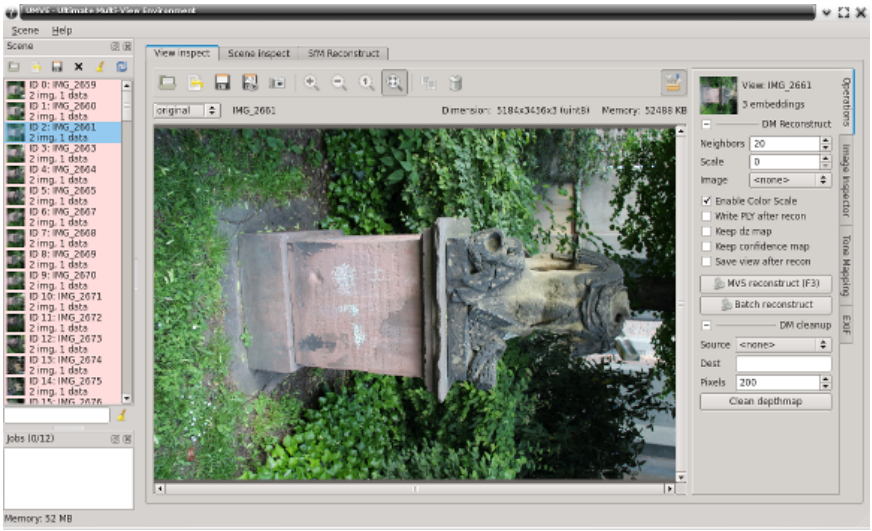

Figure 3.7: *UMVE* after creating a new scene from images. The left pane contains a list of views. A view appears in red if no camera parameters have (yet) been reconstructed. The central part is the view inspector where the individual images of the selected view can be inspected. The right-hand side contains various contextual operations, such as UI elements to start the MVS reconstruction.

SfM reconstruction: The SfM reconstruction can be configured and started using *UMVE*, or the command line tool *sfmrecon*. The UI guides through feature detection, pairwise matching and incremental SfM. What follows is the SfM reconstruction starting from an initial pair, and incrementally adding views to the reconstruction. Finally, the original images are undistorted and stored in the views for the next step. Figure 3.8 shows a rendering of the SfM reconstruction with the sparse point cloud and the camera frusta. Note how dense the frusta are spaced around the object to achieve a good reconstruction.

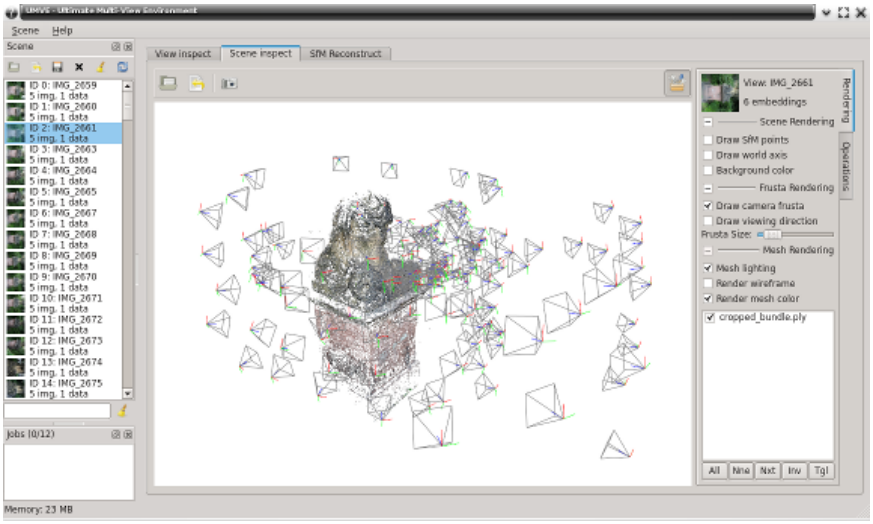

Figure 3.8: *UMVE* rendering the SfM reconstruction. The central element is the 3D scene inspector. The right hand side offers various rendering options.

MVS reconstruction: Given images with camera parameters, dense geometry is reconstructed using MVS. This can be done with either *UMVE* or the command line tool *dmrecon*. The most important parameter is the resolution level at which depth maps are reconstructed. A level of 0, or L_0, reconstructs at the original image size, L_1 corresponds to half the size (quarter the number of pixels), and so on. Looking at the resolution of recent digital cameras, a full-size L_0 reconstruction is rarely useful as finding dense correspondences gets more difficult, often leading to sparser depth maps at much higher computational cost. Using smaller images (we often use L_2), the process is faster and depth maps become more complete. See Figure 3.9 for a depth map computed at L_2.

Surface reconstruction: The *scene2pset* tool combines all depth maps in a single, large point cloud. At this stage, a scale value is attached to every point which indicates the actual size of the surface region the point has been measured from. This additional information enables many beneficial properties using the *FSSR* surface reconstruction approach [Fuhrmann and Goesele, 2014]. Then, the *FSSR* tools compute a multi-scale volumetric representation from the points (which does not require setting any explicit parameters) and a final surface mesh is extracted. The mesh can appear cluttered due to unreliable regions and isolated components caused by inaccurate measurements. The mesh is thus cleaned by deleting small isolated components, and removing unreliable regions from the surface. See Figure 3.10 for the final reconstruction.

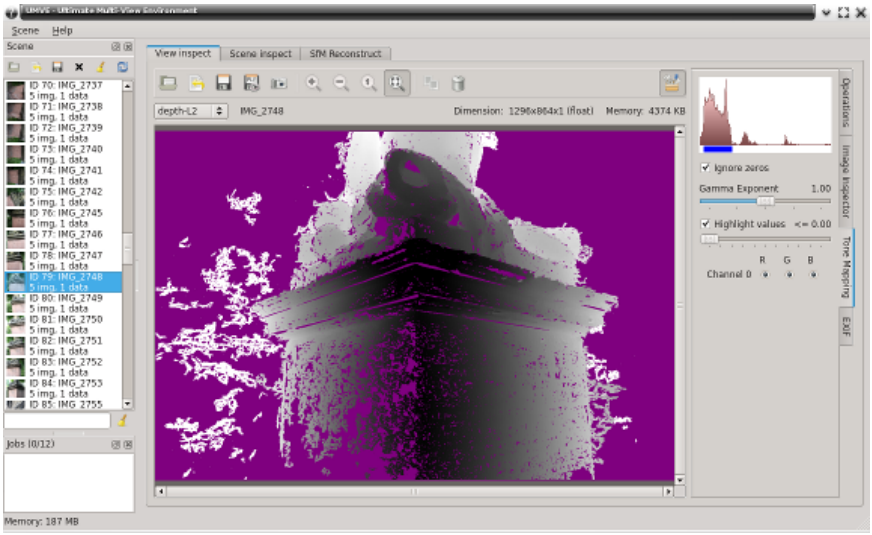

Figure 3.9: *UMVE* displaying the reconstructed depth map. The mapping of depth values to intensities can be controlled using the histogram in the upper right corner. Purple pixels correspond the unreconstructed depth values.

3.4 Reconstruction Results

In the following, we show results on a few datasets we acquired over time. We report reconstruction times for all datasets in Table 3.1. A complete reconstruction usually takes several hours, depending on the size of the dataset and the available computing resources.

Der Hass: The first dataset is called *Der Hass* and contains 79 images of a massive stone sculpture, see Figure 3.11. This is a relatively compact dataset with uniform scale as the images have the same resolution and are evenly spaced around the object. Notice that although the individual depth maps contain many small holes, the final geometry is quite complete. Here, redundancy is key as all of our algorithms are completely local and no explicit hole filling is performed.

Notre Dame: Next, we reconstruct the façade of *Notre Dame* in Paris from 131 images downloaded from the Internet. We demonstrate that our pipeline is well suited even for Internet images: The features we use are invariant to many artifacts in the images, such as changing illumination. The MVS algorithm [Goesele et al., 2007] uses a color scale to compensate for changing image appearance and is well suited for community photo collections. The surface reconstruction [Fuhrmann and Goesele, 2014] handles the unstructured viewpoints well; it will, however, not produce a particularly good model colorization. This is because the original images have non-uniform appearance and color values are computed as per-vertex averages. In Figure 3.12, e.g., the portal appears slightly brighter as it has been reconstructed from brighter images.

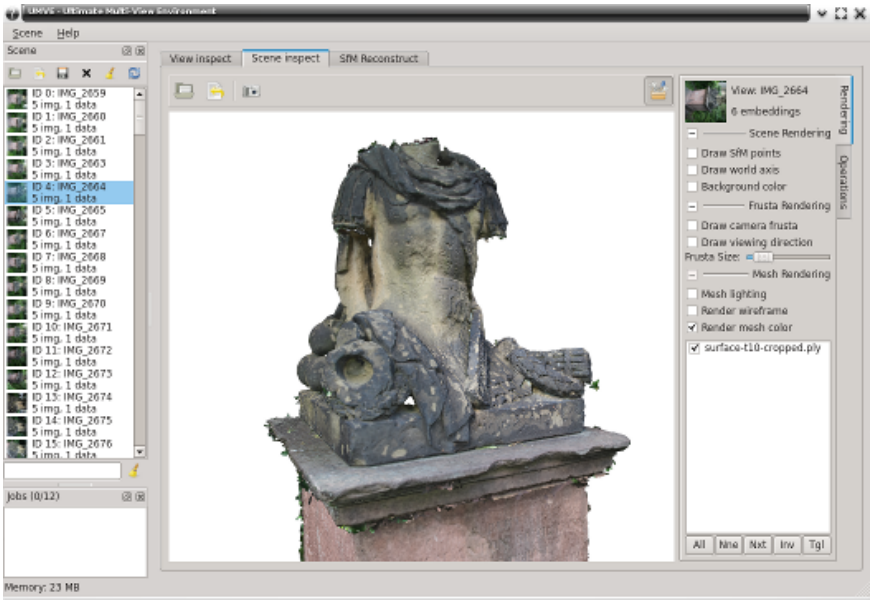

Figure 3.10: *UMVE* rendering the reconstructed surface. The reconstructed vegetation has been manually cropped from the geometry with a bounding box in order to isolate the statue.

Citywall: We conclude our demonstration with the *Citywall* dataset in Figure 3.13. The 363 input images depict an old historic wall with a fountain. This dataset demonstrates the multi-scale abilities of our system. While most of the views show an overview of the wall, some photos cover small details of the fountain. These details are preserved during reconstruction yielding a truly multi-resolution output mesh.

3.4.1 Runtime Performance

In Table 3.1, we present timings for all datasets in this chapter. The reconstructions have been computed on an Intel Xeon Dual CPU system with 6×2.53GHz per CPU. Usually 4GB of main memory are sufficient for the smaller datasets. For large datasets, we recommend at least 8 GB of main memory (such as for the *Citywall* dataset, where surface reconstruction is quite demanding). Since most parts of the pipeline are parallelized, multiple CPUs will considerably improve the computation time. Currently we do not perform computations on the GPU as only few steps of our pipeline would benefit from GPU accelleration.

All datasets listed here use exhaustive matching; however, we investigated reducing the time for feature matching. In order to quickly reject candidates that are unlikely to match, we first match a few hundred low-resolution features instead of the full set of featues. We evaluated this approach on the *Citywall*, the largest of our datasets. As can be seen in Table 3.1, this reduces the matching time by about $2/3$, rejecting about $2/3$ of all potential image pairs.

Figure 3.11: The *Der Hass* dataset. The top row shows 4 out of 79 input images and a depth map. The bottom row shows the reconstruction with and without color.

3.4.2 Limitations

A practical limitation in the presented system is the memory consumption in some parts of the pipeline. For example, surface reconstruction keeps all points in memory for the evaluation of the implicit function. Since our algorithm is purely local, we plan to implement out-of-core solutions to further scale the approach. Datasets with many images pose a bottleneck in runtime performance of SfM. This is because every image is matched to all other images, resulting in a quadratic algorithm in the number of images. Although low-resolution matching reduces runtime performance, our system is not limited to, but suitable for a few hundred images.

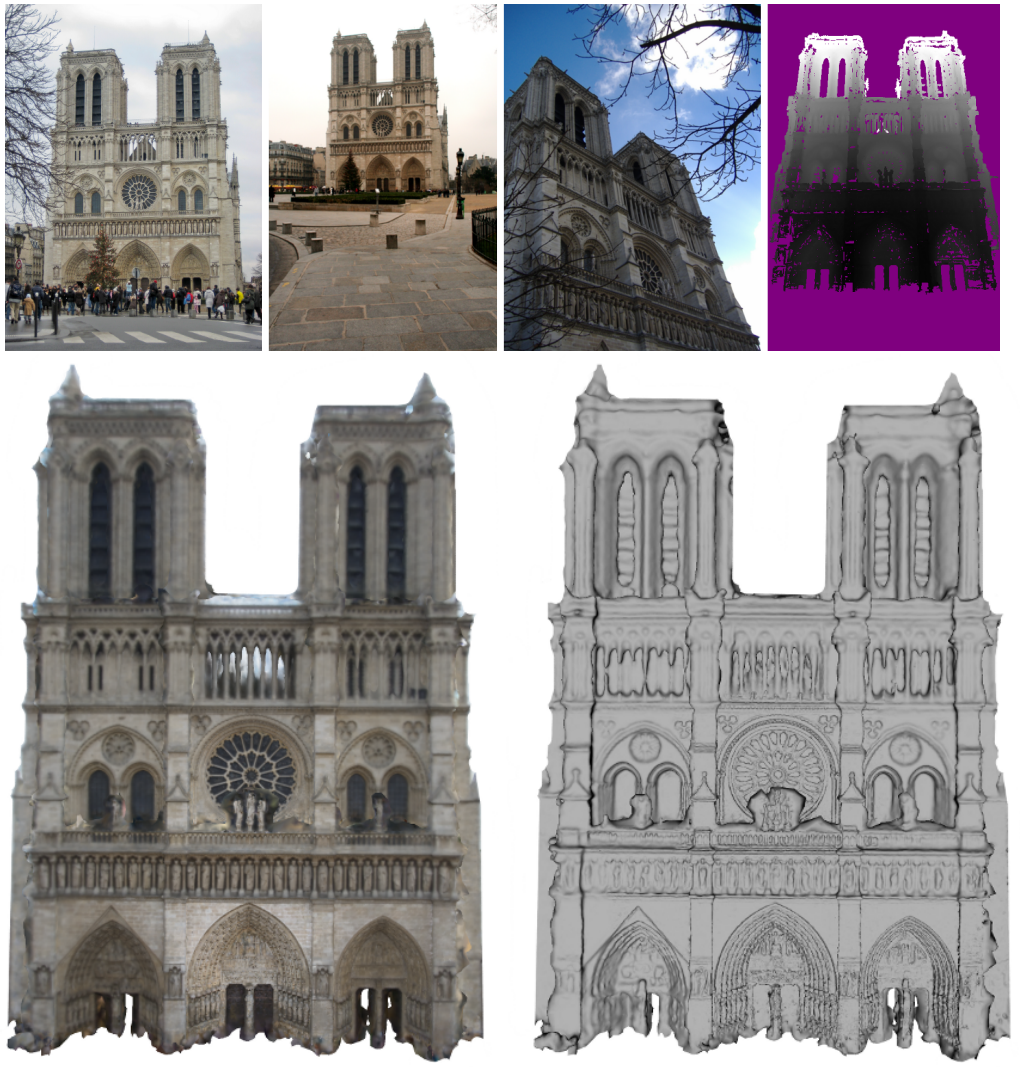

Figure 3.12: The *Notre Dame* dataset. The top row shows 3 images and 1 depth map from a total of 131 input images. The bottom row shows the reconstruction with color (left) and shaded (right). Images taken by Flickr users *April Killingsworth*, *Alex Indigo* and *Maria Boismain*.

A general limitation of multi-view reconstruction approaches is the lack of texture on the geometry. Since stereo algorithms rely on variations in the images in order to find correspondences, weakly textured surfaces are hard to reconstruct. This is demonstrated in Figure 3.14. Although most of the depth map has been reconstructed, Multi-View Stereo fails on the textureless forehead in the relief.

3.5 Software

The principles behind our software development make our code base a versatile and unique resource for practitioners (use it) and for developers/researchers (extend it).

Figure 3.13: The *Citywall* dataset. The top row shows 3 out of 363 input images and one depth map. The middle row shows the full reconstruction in color, and the bottom row shows the fountain and a small detail on the fountain.

One notable example of an extension (also relevant in the context of cultural heritage) is the texturing approach by Waechter et al. [2014]. The core functionality of MVE is available as a small set of easy to use, cross-platform libraries. These libraries solely build upon `libjpeg`, `libtiff` and `libpng` for reading and writing image formats, and do not require any other external dependencies. We strive for a user-friendly API and to keep the code size at a maintainable minimum. The correctness of many components in our code is backed by unit tests. Our GUI application requires (aside from our own libraries) the widely used QT framework for the user interface. We ship with our libraries a few command line applications for the entire pipeline to support computation on server machines without a graphical interface. MVE is tested and operational under Linux, MacOS and Windows. The source code is available from our website[1].

[1] http://www.gris.informatik.tu-darmstadt.de/projects/multiview-environment/

Dataset	Images	SfM [min]	MVS [min]	FSSR [min]
Der Hass	79	33+5	61	17
Notre Dame	131	109+8	32	26
Anton Memorial	161	180+10	121	88
Citywall	363	945+56	267	203
Citywall LRM	363	345+55	254	181

Table 3.1: Runtime performance for various datasets. The SfM timings are broken down into feature detection with matching and incremental SfM. The bottom row (*Citywall LRM*) shows the timing using low-resolution matching considerably reducing matching time.

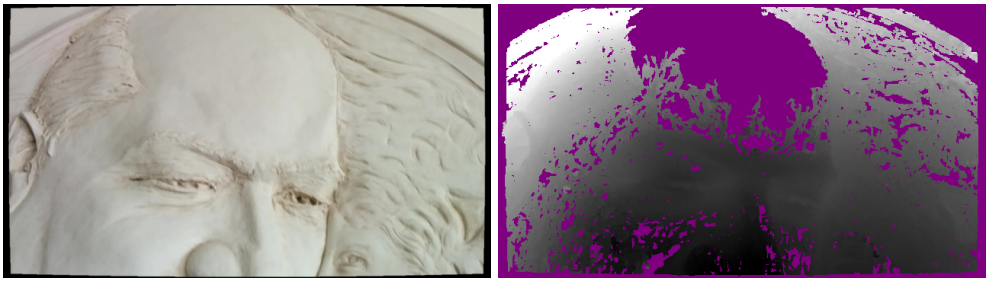

Figure 3.14: One image of a dataset with a weakly textured relief, and the corresponding depth map. The depth map contains a big hole in the homogeneous region as visual correspondences cannot reliably be established.

3.6 Conclusion

In this chapter we presented *MVE*, the *Multi-View Environment*, a free and open 3D reconstruction application, relevant to the cultural heritage community. It is versatile and can operate on a broad range of datasets. This includes the ability to handle quite uncontrolled photos and is thus suitable for reconstruction amateurs. Our focus on multi-scale data enables to put an emphasis on interesting parts in larger scenes with close-up photos. We belief that the effort and expert knowledge that went into *MVE* is an important contribution to the community.

Chapter 4

Fusion of Depth Maps with Multiple Scales

Abstract

Multi-view stereo systems can produce depth maps with large variations in viewing parameters, yielding vastly different sampling rates of the observed surface. We present a new method for surface reconstruction by integrating a set of registered depth maps with dramatically varying sampling rate. The method is based on the construction of a hierarchical signed distance field represented in an incomplete primal octree by incrementally adding triangulated depth maps. Due to the adaptive data structure, our algorithm is able to handle depth maps with varying scale and to consistently represent coarse, low-resolution regions as well as small details contained in high-resolution depth maps. A final surface mesh is extracted from the distance field by construction of a tetrahedral complex from the scattered signed distance values and applying the Marching Tetrahedra algorithm on the partition. The output is an adaptive triangle mesh that seamlessly connects coarse and highly detailed regions while avoiding filling areas without suitable input data.

Contents

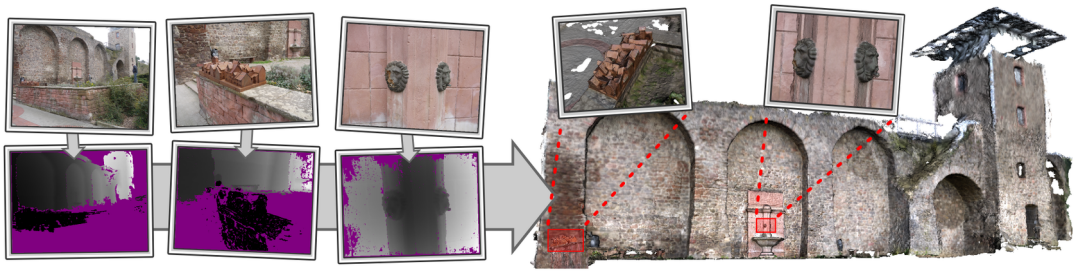

Figure 4.1: Input photographs (top left) depicting objects at different levels of detail. Multi-view stereo yields depth maps (bottom left), which inherit these multi-scale properties. Our system is able to fuse such depth maps and produce an adaptive mesh (right) with coarse regions as well as fine scale details (insets).

4.1 Introduction

Surface reconstruction is an important problem with huge practical applications and a long history in computer graphics. The goal is to build high quality 3D surface representations from captured real-word data. Important applications include the preservation of cultural heritage, model reverse engineering, and prototyping in the multi-media industry. Typical inputs to surface reconstruction algorithms are either unorganized points or more structured data such as depth maps. In this work we will focus on the latter kind of data, which is produced by range scanners and some multi-view stereo algorithms. To fully capture an object of interest, multiple overlapping depth maps are necessary, each covering parts of the object surface. In a general acquisition framework, these depth maps need to be aligned into a common coordinate system and fused into a single, non-redundant surface representation. This process is called *integration* or *fusion* of depth maps.

One source of depth maps are multi-view stereo (MVS) systems, which recently attained renewed interest [Seitz et al., 2006]. These algorithms reconstruct the scene geometry from photographs of the scene by regaining the 3D information lost during capture. Current structure-from-motion systems [Snavely et al., 2006, 2008] are able to recover the camera parameters of thousands of photographs under very uncontrolled conditions. This enables modern MVS algorithms to make use of the massive amount of Internet imagery for geometry reconstruction [Goesele et al., 2007; Furukawa and Ponce, 2010].

We desire to construct surface representations from the depth maps delivered by these acquisition systems, which is still an unsolved problem and difficult for various reasons. In particular, the photographs may be at different resolutions and show large variations in viewing parameters. The resulting depth maps inherit these properties and imply vastly different sampling rates of the surface. As in almost all acquisition processes, individual depth map samples are not ideal point samples. Instead, they represent the surface at a particular scale depending on viewing distance, focal length and image resolution. The extent of individual pixels when projected

into 3D space can therefore dramatically vary in size. We call this the *pixel footprint*. The issue of scale and pixel footprints is crucial and requires particular care when mixing samples at different scales. To our knowledge, this has not been solved convincingly in the surface reconstruction literature.

Apart from the scale issue, each depth map may be *locally incomplete*, i.e., contain regions without reconstructed depth values, a common artifact of depth maps produced by multi-view stereo (see Figure 4.1). Additionally, the individual depth values possess errors and deviate from the ground truth surface. These errors depend on the technology used to create the depth maps. Naturally, MVS approaches generate a different kind of (and much larger) error than most active range scanning systems. Given a set of depth maps of an object, some regions of the surface are typically seen by more than one depth map. We want to make use of the redundancy to suppress, or average out noise in the individual depth samples. In contrast to techniques that produce water-tight surfaces, we want to support incomplete representations and leave unobserved regions empty while closing small holes. One well known approach that handles these issues but does not take scale information into account, is volumetric range image integration (VRIP) [Curless and Levoy, 1996]. Our proposed approach is volumetric, builds on some ideas developed in VRIP but solves the scale issue while sharing advantageous properties. In particular, our contributions are

- a method to construct a discrete, multi-scale signed distance field capable of representing surfaces at multiple levels of detail, yielding a *hierarchical signed distance field*,
- a processing approach that supplements uncertain signed distance values at high resolution with data from a coarser scale, *regularizing* the distance field,
- defining a *continuous signed distance field* from the hierarchical, incomplete and scattered signed distance values by building a bounded tetrahedral complex, and
- a surface extraction approach based on Marching Tetrahedra [Doi and Koide, 1991] to produce output surfaces that are adaptive to the scale of the input data.

The remainder of this chapter is organized as follows: We first give an overview of previous work (Section 4.2) before we describe our main concept in Section 4.3. We show how to construct the hierarchical signed distance field and present a regularization technique by combining data at different resolutions in Section 4.4. Our approach for surface extraction is described in Section 4.5. We evaluate the proposed algorithm and present results in Section 4.6. Finally we conclude our work in Section 4.7.

4.2 Related Work

Surface reconstruction is an important topic for which a large variety of techniques have been proposed over the last decades. Most reconstruction techniques aim at generating a piecewise linear surface representation (such as a triangle mesh) from the input data. Methods can be classified into reconstruction from unorganized points and techniques that use the underlying structure of the data. Examples of the former class include the classical work by Hoppe et al. [1992], Moving Least Squares surfaces [Levin, 1998], RBF-based techniques [Carr et al., 2001; Ohtake et al., 2006], Poisson surface reconstruction (PSR) [Kazhdan et al., 2006], and Voronoi-based reconstruction [Alliez et al., 2007]. These methods operate in the most general setting and do not make any assumptions about the spatial structure of the data. The motivation to deal with a more specific type of input, namely depth maps, is that the acquisition process often provides us with additional information such as connectivity. Although we can always fall back to unorganized point-based reconstruction techniques by projecting all pixels of the depth maps in 3D space to produce unorganized point samples, intuition suggests that we should make use of the additional information to improve upon the results.

In fact, most methods that deal with *depth map integration* apply a depth map triangulation step first, where depth samples are connected in image space to form a triangulated surface, which is then lifted to 3D space. We can classify these methods into *parametric surface representations*, *surface-based methods*, and *volumetric methods*.

Early work in the field of depth map fusion impose an object-centered coordinate system for surface integration. Chen and Medioni [1991] apply a global reparametrization of the depth maps into a unified parameter space. Integration is then simply a matter of averaging in the overlapping areas. Similarly, Higuchi et al. [1994] integrate all data points into an object-centered parametric representation and fit a deformable mesh in order to obtain a smooth model. These techniques assume a simple topology such as a cylinder or a sphere, are therefore *parametric*, and restrict the input to very simple and compact models.

Surface-based methods such as Mesh Zippering [Turk and Levoy, 1994] or the co-measurements approach by Pito [1996] select one depth map for each surface region, remove redundant triangles in overlapping regions, and glue the remaining meshes together by connecting the boundaries. These methods can handle noise by local surface averaging of positions, but are very fragile in the presence of outliers and typically fail in regions of high curvature. Interestingly, these methods can, at least in theory, handle arbitrary scales since they attempt to fuse triangulated depth maps directly, and do not re-parameterize the data. Thus these methods work with natural pixel resolution.

Hilton et al. [1996] introduced the idea of representing the surface implicitly using a signed distance field computed from the individual depth maps. Curless and Levoy [1996] took this idea further by taking into account the direction of the sensor uncertainty to model the anisotropic behavior of sensor noise of the acquisition

device. As in most volumetric methods, the final surface can then be extracted as zero-level set of the implicit function using standard techniques such as Marching Cubes [Lorensen and Cline, 1987]. Hilton and Illingworth [1997] propose a method to reduce memory consumption of implicit functions by constructing an adaptive signed distance field stored in an octree. The octree level is adapted to surface curvature bounding the approximation error. This approach still expects depth maps with similar scale and adapts the octree with respect to geometric properties only.

Zach et al. [2007] cast the problem of depth map integration as a global optimization problem, minimizing an energy functional consisting of a total variation regularization with an L_1 data fidelity term. L_1 is more robust than L_2 data fidelity in the presence of noise and outliers, but is also very expensive: Although their method produces impressive results, it is restricted to small and compact objects sampled over regular volumes because computation time and memory consumption quickly become prohibitive.

It is worth noting that Point Set Surfaces [Alexa et al., 2001] based on Moving Least Squares [Levin, 1998] can produce implicit functions, which can be evaluated over a hierarchy with respect to approximation error, the local feature size, or even local scale information. Such a technique, however, requires that all sample points (from all depth maps) are located in memory. Another disadvantage is that a Point Set Surface can only define a smooth closed surface. Guennebaud and Gross [2007] describe a technique to define object boundaries but this requires an additional clipping normal for each boundary input point. Similar drawbacks also apply to most other point-based reconstruction techniques.

In this work, we address both multi-resolution input depth maps as well as creating a final surface that is adaptive with respect to the scale of the input data. While most volumetric methods operate on regular grids, some techniques use hierarchical data structures, including [Kazhdan et al., 2006; Hilton and Illingworth, 1997; Soucy and Laurendeau, 1992]. Sometimes these (and similar) methods are said to be "multi-resolution approaches", which typically means that the resulting mesh is adaptive. However, input data at various scales is not addressed. Further, we show why methods such as VRIP [Curless and Levoy, 1996], Poisson Surface Reconstruction [Kazhdan et al., 2006] and Point Set Surfaces [Guennebaud and Gross, 2007] cannot be modified in an obvious way to handle multi-scale input.

4.3 Concepts

In the noise-free case, i.e., if all samples are perfect, we would like to reconstruct a surface from the samples corresponding to the highest resolution information available at that location. Thus, adding infinitely many depth maps with lower resolution should not change the reconstruction. In contrast, many existing techniques converge towards the lower resolution surface if more and more low resolution depth maps are added. This issue is illustrated in Figure 4.2.

In VRIP, this behavior is very pronounced because the surfaces (obtained by triangulating the individual depth maps) are resampled into the volume without taking

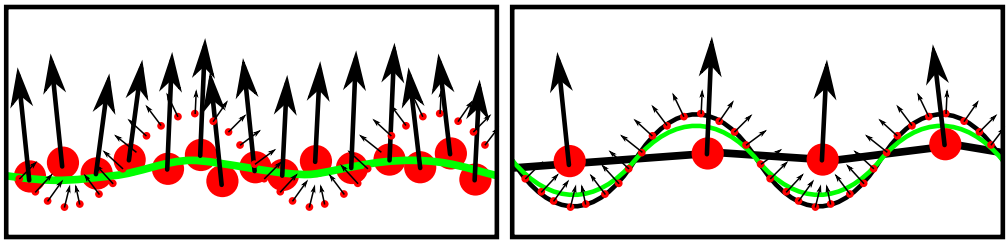

Figure 4.2: *Top:* Point samples (red) with normals for a surface sampled at low and high resolution (black curves). The reconstructed surface (green) degenerates only slightly because the density of high resolution samples dominates. *Bottom:* Adding more and more low resolution samples causes the surface to converge towards the coarse geometry.

the scale into consideration. Combining a single low resolution depth map with a high resolution depth map considerably influences the high-resolution geometry. In PSR, this issue is much less apparent because PSR considers the *density* of the samples, and the *amount* of samples contained in a depth map varies with the scale of the depth map. However, the same behavior can be observed when adding the same *density* of low resolution samples and high resolution samples. Thus, in general, PSR has the same convergence behavior for n low resolution depth maps with $n \to \infty$.

A weighting scheme that leaves out the contribution of coarse information is hard to realize: In a regular volume like in VRIP, the required information is simply not present because the implicit surface is not represented at different scales. In point-based techniques, a single unreliable high-resolution sample potentially prohibits the use of coarser information essential for reliable reconstruction.

A more formal view on the issue of scale is given by scale space theory [Lindeberg, 1998]. Given an image, the scale space of the image is constructed by introducing a parameter t of scale and convolving the image signal with a Gaussian filter with variance $t = \sigma^2$, thus representing the image as a one-parameter family of smoothed images, which is called the *scale-space representation* of the image. This theory also applies to 3D images such as signed distance fields (SDF). We assume that the SDF of a surface at a given resolution can be approximated by a low-pass filtered SDF of the same surface at higher resolution. The scale space parameter t has the interpretation that image structures of size $t \geq 1$ (in pixels) have largely been eliminated at scale t^2. This interpretation suggests that we should be careful when combining depth maps at different scales. In particular, if we want to keep structures of a specific size in a depth map, say size t, we should avoid naïvly combining it with another depth map at scale t^2.

Another important aspect of our work is that noise in the depth maps is typically coherent between samples from a single depth map, but differs between samples from different depth maps at a different scale. In particular, this is difficult for point-based reconstruction techniques, since they can no longer exploit the fact that the consistency in noise is tied to proximity. In fact, points that are spatially close

together may have very different amount of noise and should therefore be processed independently. In our system, we assume a linear correlation between the scale and the expected noise of a sample: For example, if the scale of two samples differs by a factor of two, the depth uncertainty also doubles.

We assume that the input depth maps to our system are band-limited such that they can be triangulated without significant aliasing artifacts. While this is the case for depth maps from most acquisition systems (such as multi-view stereo and structured light scanners), some technologies produce depth maps with different characteristics. For example, LIDAR scanners typically produce samples with a sample spacing that is substantially larger than their footprint. This is due to the very small point spread function of the LIDAR beam and these samples need additional filtering to suppress aliasing artifacts in the triangulated surface.

In the next section we describe the underlying ideas and properties of our technique. We decided to approach the reconstruction problem using a volumetric representation of the input data. Note, however, that the principle behind our solution applies to other representations as well.

4.4 Signed Distance Field

One of our key ideas is to separately aggregate the contributions of the individual depth samples at their corresponding scale. We are therefore able to select a suitable scale for final surface extraction and avoid mixing up different scales. In order to do this, we aggregate geometric information in the form of a signed distance field (SDF) in scale space, i.e., the 3D Euclidean space plus one dimension of scale. We associate a scale with each depth sample, which then only contributes at that specific scale parameter in scale space at its 3D position. We explain how we define the scale of a sample in the next section.

So far averaging of information would not be possible because overlapping regions in the depth maps rarely have exactly the same scale. A common solution is to discretize the scale space into octaves, which yields a hierarchical representation. The levels of the octaves correspond to a doubling of scale, and all samples within a single octave are combined to produce average surfaces. In the (unlikely) case that all depth maps contribute to a single octave only, the dataset has uniform scale, and our technique gracefully degrades to the VRIP algorithm.

Assigning each sample to exactly one scale can lead to artifacts near the boundaries of the octaves, because contributions are distributed between two neighboring octaves. We therefore transfer geometric information from the coarser octaves to the finer octaves, thus *regularizing* the fine geometry using the coarser one. Unreliable measurements, such as a surface seen at a grazing angle, are pruned from the hierarchical SDF (hSDF). Finally, we extract the isosurface from the hSDF by triangulating the zero-crossing corresponding to the finest geometric information available.

4.4.1 Construction

We take as input a set of registered depth maps, generated, e.g., by a range scanner or a multi-view stereo approach, optionally with confidence values and colors. The depth maps are triangulated in image space and the triangulation is lifted to 3D. If the depth disparity between two vertices of a generated triangle is above a threshold, we assume a depth discontinuity and discard the triangle. To dynamically choose the disparity threshold, we use our notion of the pixel footprint. The *pixel footprint* $F_\circ$ is the width (or height) of a fronto-parallel square corresponding to the pixel (u, v) in the image, projected to its 3D location $\vec{x}(u, v)$ on the object. We detect a depth discontinuity between two neighboring pixels if the depth disparity is above a threshold $\rho \cdot F_\circ$, where $F_\circ$ is the footprint of the pixel closer to the camera, and ρ is a user-defined constant. Triangles where at least one edge contains a detected depth discontinuity are discarded. We achieve overall good results with $\rho = 5$.

The next step of our algorithm inserts the triangulated depth maps into the hierarchical signed distance field. Our hierarchy corresponds to a primal octree, where each cube has eight voxels in the corners and is subdivided into eight sub-cubes. These sub-cubes create 27 new voxels, eight of these voxels coincide with voxels of the parent node. We explicitly keep these duplicated voxels to represent information at different levels of the hierarchy. Technically, we do not explicitly store the octree hierarchy, but insert all voxels into a map data structure, which maps the voxel index (l, I_l) to the voxel data. The index is composed of the level l and the index $I_l \in \{0, \dots, 2^{3l} - 1\}$ within that level and uniquely determines the position of a voxel with respect to the root node's axis aligned bounding box. Initially, the data structure is empty and voxels are created as they are requested for the first time.

The triangles of each depth map are inserted one after another. For each triangle a decision is made which octree level it affects. Again, we use our notion of the pixel footprint to make that decision. Each vertex of the triangle carries an associated footprint size from the depth map pixel that generated the vertex, and we declare the smallest footprint $F_\circ$ of the three triangle vertices as the representative footprint of the triangle $F_\triangle$. To sample the triangle, we enforce that the footprint $F_\square$ of octree cells is smaller or equal to $F_\triangle \cdot \lambda^{-1}$, where λ is the sampling rate. We typically set $\lambda = 1$ and define $F_\square$ as the edge length of the octree cell (i.e., the spacing between voxels). The appropriate octree level l_T for triangle T is efficiently found by taking the binary logarithm of the root node's footprint $F_\square^R$ divided by the maximum sample spacing $F_\triangle \cdot \lambda^{-1}$:

$$l_T = \lceil \log_2 \left(\frac{F_\square^R \cdot \lambda}{F_\triangle} \right) \rceil \tag{4.1}$$

Once we computed the level l_T of triangle T, we need to identify all affected voxels, i.e., those voxels that fall in a band around the triangle. This is controlled by the *ramp length*, see Curless and Levoy [1996] for details. The ramp length $\gamma \cdot F_\square$ is calculated by multiplying the footprint $F_\square$ of the octree cell with the ramp size factor γ; thus the ramp length is constant for each octree level. The ramp length factor should be chosen according to the expected maximum noise of the data set and the

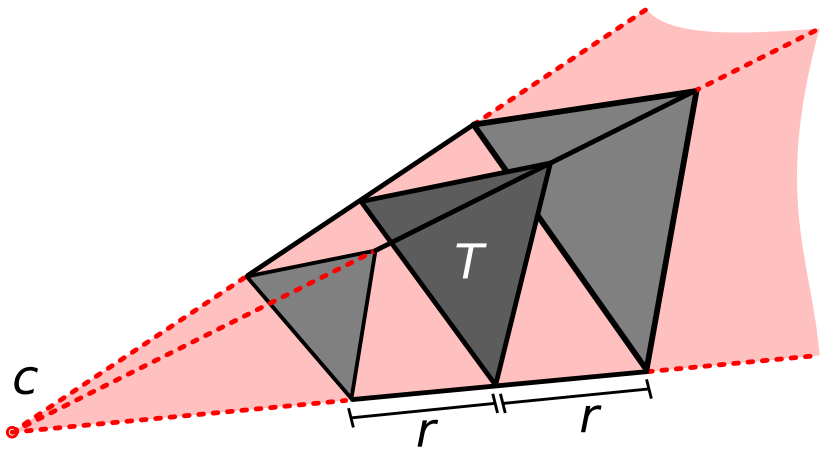

Figure 4.3: Truncated tetrahedron created by shooting rays from the sensor center c through the vertices of triangle T. The ramp length is denoted by r.

scanning technology; reasonable parameter values are between $\gamma = 2$ for clean, range scanned data and $\gamma = 8$ for MVS datasets with heavy noise.

To identify affected voxels, we extrude the triangle T by following the rays from the sensor center through the triangle vertices. We bound the resulting cone and limit the volume to the ramp length around the triangle; this yields a tetrahedron with one corner truncated, see Figure 4.3. To simplify things, we create the bounding box of the truncated tetrahedron and analytically identify the indices of all voxels, yet created or not, inside the bounding box. We calculate the signed distance from each voxel to the triangle by shooting a ray from the camera center through the voxel and either create or update the voxel on hitting the triangle. When creating a new voxel, we assign a weight value in addition to the distance to the voxel. Our weight value is calculated similar to VRIP [Curless and Levoy, 1996]: We multiply individual weights for angle deviation (the dot product between the ray and the hit point normal), a truncated tent weight function of the absolute distance, and the confidence value at the hit point, linearly interpolated from the mesh vertices. When updating a voxel x, we use the following cumulative rules [Curless and Levoy, 1996]:

$$W_{i+1}(x) = W_i(x) + w_{i+1}(x) \tag{4.2}$$

$$D_{i+1}(x) = \frac{W_i(x)D_i(x) + w_{i+1}(x)d_{i+1}(x)}{W_{i+1}(x)} \tag{4.3}$$

where $d_i(x)$ and $w_i(x)$ are the signed distance and weight values from the ith range image, and $D_i(x)$ and $W_i(x)$ are the cumulative signed distance and weight values after inserting the ith range image. For the rest of this chapter, we interchangeably use the terms *weight* of a voxel and *confidence* of a voxel.

A particular surface structure can appear quite differently depending on the scale of the image (or depth map) representing the surface. Thus different geometric representations of the same surface may deviate from each other. This deviation can potentially lead to duplicated surfaces in the output mesh, see Figure 4.4 for an il-

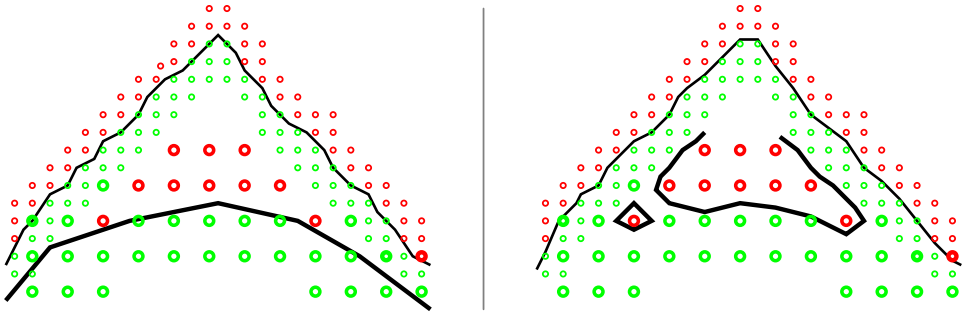

Figure 4.4: Left: A high-resolution and a low-resolution depth map of the same surface and the voxels that sample the depth maps at different scales. Green and red voxels are in front of and behind the surface, respectively. Right: The isosurface extracted from the distance field yields artifacts, which we avoid by inserting at coarser scales.

lustration. To avoid duplicated surfaces, one could use infinitely large ramps, such that voxels at coarser levels are overridden, but this is not feasible in practice. If the deviating surface at level l is, however, within the ramp of the surface of level $l+1$, the duplication is detected and overridden. To make this mechanism work over several scales, we additionally insert the depth maps into a few coarser levels. In our experiments, we always use four coarser levels.

4.4.2 Regularization

The hierarchical signed distance field now contains a sampled representation of all input depth maps at various scales, depending on the footprints of the inserted triangles. The representation is incomplete, i.e., contains holes in unseen or unreconstructed regions. But even in areas where data is available, it might be very unreliable, having a low confidence value caused by, e.g., uncertain reconstruction or surfaces seen at grazing angles. Our goal is to improve these unreliable samples by transferring distance and confidence measures from coarser levels where available.

We make a pass through the hSDF in coarse-to-fine order, searching for occupied voxels with confidence w_l below a threshold τ_0. For each of these sub-confident voxels at level l, we interpolate its distance value from distance values at coarser level $l-1$ if all required voxels at that level are occupied. The number of required voxels varies, depending on the voxel position in the hierarchy: If a voxel at level l coincides with a voxel at level $l-1$, only the coinciding voxel is required for "interpolation". The other possible configurations require two, four, or eight voxels at the coarser level for interpolation.

We perform a weighted blend of the distance and weight values. Since the weight at the coarser level $l-1$ can be arbitrary high, we adapt the reasoning of Mitchell [1987] to this case (similar to Gortler et al. [1996]) and clamp the confidence values to τ_0 to avoid oversmoothing. τ_0 can be seen as a saturation threshold. The blended

voxel x with distance $\tilde{d}_l$ and weight $\tilde{w}_l$ at level l then becomes:

$$\tilde{d}_l = \frac{d_l \cdot w_l + d_{l-1} \cdot (\tau_0 - w_l) \cdot \min(1, \frac{w_{l-1}}{\tau_0})}{w_l + (\tau_0 - w_l) \cdot \min(1, \frac{w_{l-1}}{\tau_0})} \tag{4.4}$$

$$\tilde{w}_l = w_l + (\tau_0 - w_l) \cdot \min(1, \frac{w_{l-1}}{\tau_0}) \tag{4.5}$$

If the blended confidence $\tilde{w}_l$ remains below a second threshold $\tau_1 \leq \tau_0$, we delete the voxel from the octree since it is not reliable enough for reconstruction. Voxels that could not be updated due to missing information at level $l-1$ with confidence $w_l < \tau_1$ are deleted.

The *saturation threshold* τ_0 as well as the *confidence threshold* τ_1 need to be chosen according to the dataset. The confidence threshold is similar to the one in VRIP, that is often used to remove clutter in the reconstruction. If the dataset contains little redundancy, i.e., most regions are observed by one or two depth maps only, reasonable values are $\tau_0 = 0.5$ and $\tau_1 = 0.1$. For scenes with higher redundancy where regions are seen by more depth maps, the thresholds can be increased.

After all voxels have been processed, there are still duplicated voxels at different levels in the hierarchy. Since all unconfident voxels have been deleted, we ultimately trust in the remaining voxels at the highest resolution. Hence we take another pass through the octree in fine-to-coarse order and delete for each occupied voxel at level l all coinciding voxels at coarser levels $\{l - i \mid 1 \leq i \leq l\}$.

4.5 Extracting the Isosurface

In the previous step we converted the hierarchical signed distance field to a scattered signed distance field by deleting unconfident and duplicated voxels. Each voxel has an associated distance value as well as optional per-voxel attributes. Although the 3D positions of the voxels are structured as they are derived from a primal octree, isosurfacing turns out to be a difficult problem.

Most prior methods apply the Marching Cubes (MC) algorithm [Lorensen and Cline, 1987] to the implicit function, but this only works for regular samplings. Several approaches have been developed to get around this limitation, some of them require knowledge of the original signed distance function, others demand restrictions on the octree topology, i.e., require that the level difference between adjacent leaf nodes must not be greater than one. Dual methods pose less restrictions but require hermite data [Ju et al., 2002] or can introduce topological artifacts [Schaefer and Warren, 2005]. A more recent technique [Kazhdan et al., 2007] solves these issues but requires an octree where each non-leaf node has all eight children allocated. Schroeder et al. [2004] use a unique global point numbering (vertex indices) to produce compatible triangulations across cell boundaries. We are, however, not aware of a suitable modification that generalizes the method to incomplete octrees. So none of the direct methods we know of applies to our case.

4.5.1 Creating the Complex

We therefore use a more general approach and consider our voxels as scattered samples of the signed distance field. We apply a global Delaunay tetrahedralization [Doi and Koide, 1991] to all voxel positions. This yields a tetrahedral complex that covers the convex hull of all voxels. The downside of this approach is that the shape of the data domain (which is typically not convex) is not taken into account, and some tetrahedra connect unrelated parts of the distance field with each other. Thus erroneous interpolation between unrelated regions becomes possible, creating phantom surfaces.

To remove these tetrahedra, we define a neighborhood relation on the voxels. We then delete all tetrahedra that contain at least one edge between non-neighboring voxels. When this relation is carefully designed, we can not only detect those tetrahedra that bridge unrelated parts of the implicit function but also exploit the generic Delaunay tetrahedralization to fill small holes in the surface, by keeping some tetrahedra that would have been removed otherwise.

Consider two voxels A and B at different levels $L(A)$, $L(B)$. Without loss of generality, assume A is the voxel at a coarser level. We define voxels A and B as neighboring if B is contained in the cube that is spanned by the 27-neighborhood of voxel A. Additionally, we enlarge this neighborhood by n voxels in each direction at the finer level of B. Thus, we can express the maximum extent of the neighborhood in each direction at the level of B as $2^{L(B)-L(A)} + n$. The same rule applies for voxels at the same level, which yields a neighborhood distance of $2^0 + n = n + 1$. We typically use $n = 2$ for very conservative hole filling. See Figure 4.5 for an illustration of the neighborhood relation in 2D.

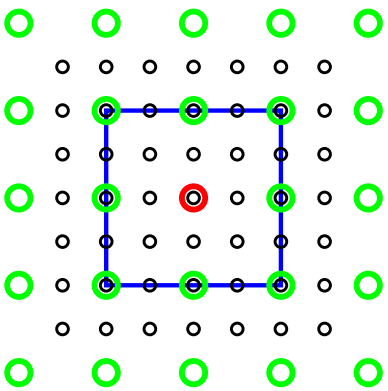

Figure 4.5: The voxel neighborhood in 2D. The neighborhood of the center voxel (red) at level l consists of all voxels at levels $\geq l$ within the blue square (the 9-neighborhood at level l). In addition, the n-ring (here: 1-ring) around the blue square at each level $\geq l$ is also part of the neighborhood of the red voxel.

4.5.2 Extracting the Surface

We now apply the Marching Tetrahedra algorithm [Doi and Koide, 1991] to the resulting tetrahedral mesh. The extracted surface mesh is adaptive, with fewer triangles in regions where only coarse information is present and more triangles in detailed regions modeled by high-resolution depth maps.

Applying Marching Tetrahedra is an attractive choice for isosurface extraction because of its stability, simplicity and performance. The downside of this simple approach, however, is the vast amount of poorly shaped triangles that do not contribute much to the accuracy of the surface. To address this, we optimize the tetrahedralization for surface extraction similar to Schaefer and Warren [2005]. For each edge in the tetrahedral mesh with a zero crossing that is located very close to one of the edge vertices, we pull the vertex along the edge onto the crossing and set its distance value to zero. In combination with a simple modification of the Marching Tetrahedra algorithm to prevent zero-area faces, this results in significantly fewer degenerate triangles, see Figure 4.6.

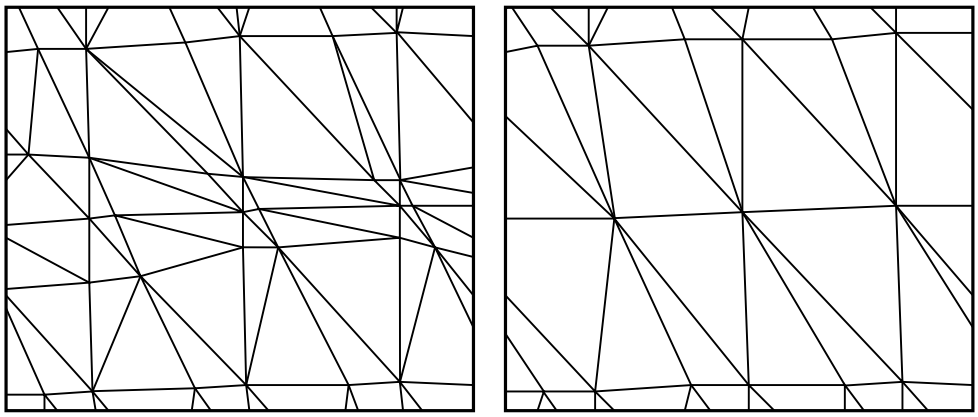

Figure 4.6: Optimized isosurface extraction. The original MT triangulation (left) and the optimized triangulation (right) obtained by pulling vertices of the tetrahedral mesh to the zero crossing.

4.6 Evaluation and Results

We now present some results on various datasets. Figure 4.7 shows a reconstruction of the Bunny dataset from laser scanned range images provided by the *Stanford Scanning Repository*. This dataset has negligible scale variations, thus triangles are typically inserted at a constant level. In this case, our algorithm gracefully degrades to the behavior of and produces very similar results to VRIP. One difference, however, is that the parameters of our method are more intuitive. We simply set the sampling rate $\lambda = 1$ and the ramp size factor $\gamma = 4$, and our algorithm determines the appropriate voxel spacing, which needs to be explicitly specified in VRIP. Since

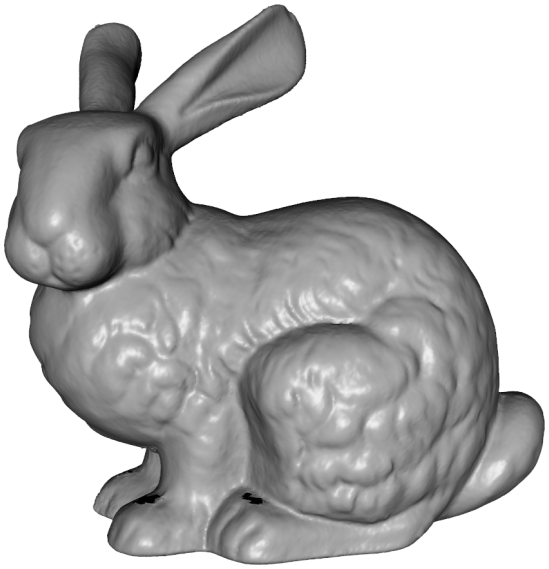

Figure 4.7: The Stanford Bunny dataset reconstructed from 10 range images provided by the Stanford Scanning Repository.

Figure 4.8: The Temple data set from the multi-view stereo evaluation effort by Seitz et al. [2006].

all depth values have more or less the same footprint, we can omit the regularization step, setting $\tau_0 = 0$.

Figure 4.8 (right) shows a reconstruction of the Temple data set from the Middlebury MVS evaluation effort [Seitz et al., 2006]. We first recovered the depth maps from the 312 input images using the MVS system by Goesele et al. [2007], and fused all depth maps using ramp size factor $\gamma = 8$ and sampling rate $\lambda = 1$. Note that we used the same MVS system for all reconstructions.

Our next dataset consists of over 700 photographs of the Cathedral of Notre Dame de Paris downloaded from Flickr with vastly different resolutions and viewing parameters. This dataset is challenging, contains very uncontrolled images taken with different cameras, thus the depth maps are tainted with a lot of noise (see Figure 4.9 for our surface reconstruction result). Since people tend to make most pho-

Figure 4.9: Some input photographs of the facade of Notre Dame de Paris that show the variations in scale (top row), and a surface reconstruction from about 700 depth maps (bottom row). We want to thank the following Flickr users for permission to use their images, ordered by appearance: *Brian Jeffery Beggerly, Eric Wilcox, Yvonne Yuen, Sara Hopkins, and Brian Beaver.*

tos of the center portal, we focused our attention on that region for a comparison with VRIP and Poisson, see Figure 4.10. To make the comparison fair, we specified a bounding box around the center portal for VRIP and Poisson, and reconstructed only within this region. Even though VRIP and Poisson operated on a subset of the data, our reconstruction shows more detail and yields a more crisp result, but also more noise. The influence of coarse depth maps on the VRIP reconstruction quality is clearly visible; for Poisson this effect is less visible but still noticeable.

We captured a dataset called *Stones* (118 photographs) that shows a metal door next to a wall built of stones. Each image represents the geometry at a different scale as we moved closer to the wall while taking the images. The reconstruction consists of various, seamlessly connected scales, from coarse regions to highly accurate geometry on the order of millimeters (see Figure 4.11).

Finally, we evaluate our reconstruction pipeline with a large MVS dataset called *Citywall* (see Figure 4.1) consisting of 561 photographs and corresponding depth maps. We reconstructed each depth map with resolution 500×375. In this scene, the footprint of the individual depth samples varies dramatically. A focus in this scene was the detailed reconstruction of the fountain with its two lion heads and

Figure 4.10: A comparison of VRIP (left), Poisson surface reconstruction (middle) and our reconstruction (right). Both VRIP and Poisson reconstructions are smoother but also less detailed.

Name	DMs	Resolution	Time	Voxel
Notre Dame	715	mixed	7h + 1h + 4m	$103 \cdot 10^6$
Stones	118	1000×750 px	2h + 23m + 2m	$41 \cdot 10^6$
Citywall	564	500×375 px	6h + 1h + 4m	$49 \cdot 10^6$

Table 4.1: Statistics of the reconstruction results. The individual timings are for constructing the octree, building the tetrahedral mesh and extracting the isosurface, respectively.

the replica of the historic city, see Figures 4.12 and 4.13.

We compare our full reconstruction with a Poisson surface reconstruction of the fountain. To do this, we clipped all samples with a bounding box around the fountain and provided the clipped point set to PSR. The reconstruction clearly shows an overly smooth result, caused by many low-resolution samples from depth maps taken further away from the surface (see Figure 4.14).

The running time and memory consumption of our algorithm is dependent on the amount, resolution, and scale of the input depth maps. Reconstruction details are given in the Table 4.1. The table shows the name of the dataset, the number of fused depth maps, the resolution of the depth maps (if uniform), the time required for reconstruction, and the number of generated voxels.

4.7 Conclusion

We presented a hierarchical, volumetric approach for depth map fusion that takes into account the scale (or footprint) of the individual depth samples to extract adaptive, high-quality surfaces. Although the basic principle of our algorithm is inspired by VRIP [Curless and Levoy, 1996], the new algorithm is, to our knowledge, the first successful attempt to handle multi-resolution data. Our results show a clear improvement over traditional depth map fusion techniques.

Figure 4.11: The top row shows some input images from the *Stones* dataset, and the middle row shows the reconstructed scene. The bottom row shows a close-up view of our reconstruction with and without texture (left, middle), and the corresponding VRIP reconstruction (right).

Figure 4.12: A detailed view on the fountain of the Citywall dataset.

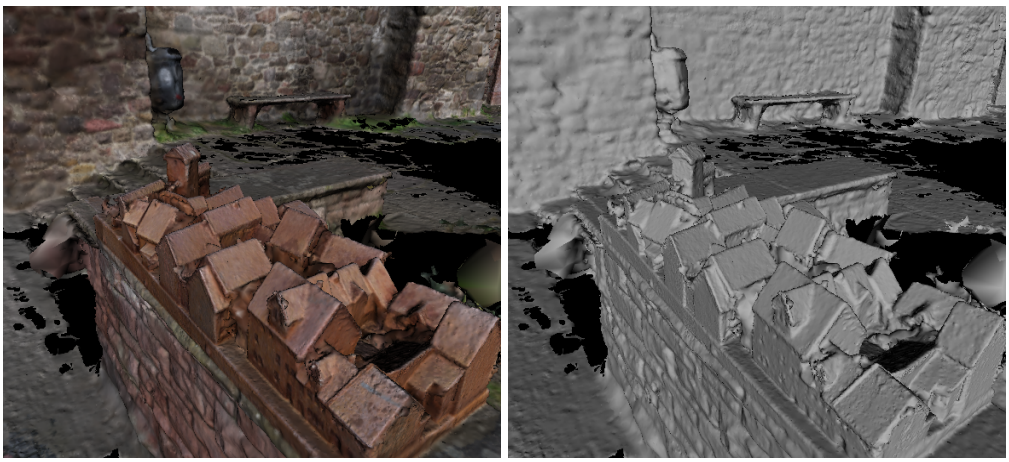

Figure 4.13: A detailed view on a miniature replica of a historic city contained in the Citywall dataset.

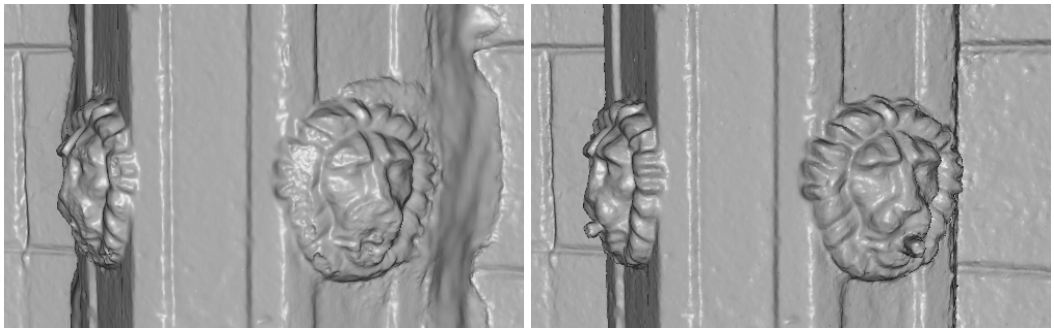

Figure 4.14: Poisson Surface Reconstruction on a small bounding box around the fountain (left). The reconstruction yields a smooth and flat result whereas our result (right) features more detailed geometry (e.g., compare the spout at the mouth).

Chapter 5

Floating Scale Surface Reconstruction

Abstract

Any sampled point acquired from a real-world geometric object or scene represents a finite surface area and not just a single surface point. Samples therefore have an inherent scale, very valuable information that has been crucial for high quality reconstructions. We introduce a new method for surface reconstruction from oriented, scale-enabled sample points which operates on large, redundant and potentially noisy point sets. The approach draws upon a simple yet efficient mathematical formulation to construct an implicit function as the sum of compactly supported basis functions. The implicit function has spatially continuous "floating" scale and can be readily evaluated without any preprocessing. The final surface is extracted as the zero-level set of the implicit function. One of the key properties of the approach is that it is virtually parameter-free even for complex, mixed-scale datasets. In addition, our method is easy to implement, scalable and does not require any global operations. We evaluate our method on a wide range of datasets for which it compares favorably to popular classic and current methods.

Contents

Figure 5.1: Input photographs (top left) depicting objects at different levels of detail. Multi-view stereo yields depth maps (bottom left), which inherit these multi-scale properties. Our system is able to fuse such depth maps and produce an adaptive mesh (right) with coarse regions as well as fine scale details (insets).

5.1 Introduction

Surface reconstruction from sampled data is a long-standing and extensively studied topic in computer graphics. Consequently, there exists a broad and diverse range of methods with various strengths and weaknesses. One well-known example is VRIP [Curless and Levoy, 1996], an efficient and scalable method able to create high quality models. Due to these properties, it was extensively used in the context of the Digital Michelangelo project [Levoy et al., 2000] to merge the captured range images. Since then many new techniques were developed that, e.g., use more advanced mathematical concepts, are able to smoothly interpolate holes, or employ hierarchical techniques. These approaches come, however, often at the cost of limited efficiency, scalability or certain quality issues. Moreover, they frequently treat reconstruction as completely separate from the actual sample acquisition process.

Our goal in this chapter is to present a method that is able to efficiently reconstruct high quality meshes from acquired sample data even for large and noisy datasets using a virtually parameter-free method. Examples of such reconstructions from hundreds of millions of samples are the *Fountain* dataset (Figure 5.1) and the full-sized David statue (Figure 5.12) from the Digital Michelangelo project [Levoy et al., 2000]. Following on earlier work, we attach a *scale* value to each sample which provides valuable information about the surface area each sample was acquired from.

The sample scale can in general be easily derived from the acquisition process (e.g., from the sample footprint in a structured light scan or the patch size in a multi-view stereo algorithm). This definition of scale that has been used in prior work [Mücke et al., 2011; Fuhrmann and Goesele, 2011; Klowsky et al., 2012]. Knowing scale allows us to reliably identify redundancy in the samples and avoid intermingling data captured at different scales (such as in multi-view stereo depth maps reconstructed from images at various distances to the geometry, as shown in Figure 5.1). Without scale information, datasets containing non-uniform redundancy, sample resolution or noise characteristics will, in general, lead to poor reconstruc-

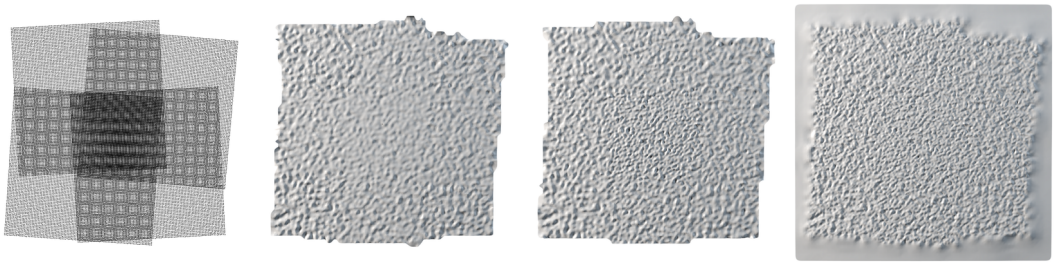

Figure 5.2: Sample density versus sample scale. Noisy input samples from four synthetic, overlapping scans (left). Reconstructed surface with proper scale values (middle left) and with scale values estimated from the sample density (middle right). In the former case, redundancy is properly exploited for noise reduction. In the latter case, however, higher sample density leads to higher frequency noise in the reconstruction. Similarly, the Poisson Surface Reconstruction [Kazhdan and Hoppe, 2013] (right) also suffers from higher frequency noise.

tions. Many methods do adapt the reconstruction resolution to the input data in some way. These decisions, however, are often based on the density of the input data. Figure 5.2 shows a common case that demonstrates why density and scale are not always related: An increased sample density is often caused by data redundancy. Being able to detect this redundancy makes the difference between proper noise reduction and reconstructing higher frequency noise.

Conceptually, our method is based on reconstructing an implicit function F from the input samples. F has spatially continuous scale (*floating scale*), i.e., the scale at which surface details are represented by F varies continuously as defined by the scale of the input samples. We then define a discrete, scale-adaptive sampling of F and extract an isosurface corresponding to the zero-level set of F. The implicit function F is constructed as the sum of compactly supported basis functions. But unlike, e.g., Radial Basis Functions [Carr et al., 2001] or Smooth Signed Distance Reconstruction [Calakli and Taubin, 2011] our method does not require the solution of a global problem, is computationally tractable, and the implicit function can, given the samples, readily be evaluated. The compact support leads to an approach that reconstructs open meshes and leaves holes in regions where data is too sparse for reliable reconstruction. This is useful for scenes which cannot be completely captured, such as outdoor scenes. This stands in contrast to methods such as Kazhdan and Hoppe [2013], which perform excellent hole-filling but often hallucinate geometry in incomplete regions, requiring manual intervention.

Our contributions are:

- The reconstruction of a continuous, signed implicit function with spatially continuous scale (*floating scale*) using a simple mathematical formulation,
- a virtually parameter-free approach that selects the appropriate reconstruction scale and automatically adapts the interpolation and approximation behavior depending on the redundancy in the data,

- no costly aggregation of samples in a pre-processing step so that the implicit function can, given the input samples, readily and rapidly be evaluated, and
- an efficient and scalable method that does not require any global operations (such as applying graph cuts or solving large systems of equations).

In the remainder of this chapter we first review related work (Section 5.2). We then formally introduce our surface reconstruction approach (Section 5.3) and perform experiments on synthetic and real-world data (Section 5.4). Next, we describe the isosurface extraction (Section 5.5) and evaluate our approach (Section 5.6). We finally discuss the limitations of our approach and conclude with an outlook on future work (Section 5.7).

5.2 Related Work

We give an overview of closely related surface reconstruction algorithms with a focus on how they handle scale, whether and which parameters they require, and to what extent they use costly global optimizations to reconstruct the final mesh.

Volumetric Range Image Processing (VRIP) [Curless and Levoy, 1996] averages surfaces (regardless of scale) in a regular grid using a volumetric approach based on the signed distance function. Averaging a high resolution and a low resolution surface yields an average surface quickly blurring the high resolution information. Our method is similar in that it also uses the weighted average of locally estimated functions to define the implicit surface compactly around the input data. While VRIP's implicit function is approximately a signed distance function, the interpretation of our function is more abstract and values do not represent distances. In contrast to VRIP, (Screened) Poisson Surface Reconstruction [Kazhdan et al., 2006; Kazhdan and Hoppe, 2013] uses the density of the samples as indicator for scale. Thus a denser set of samples is assumed to originate from a surface sampled at a higher resolution. However, the sampling rate is not necessarily related to the sample resolution, and an increased sampling rate may simply be caused by data redundancy (see Figure 5.2). As a consequence, Poisson Surface Reconstruction starts fitting to the sample noise and hallucinates geometric detail. Mesh Zippering [Turk and Levoy, 1994] selects a triangulated depth map for each surface region, eroding redundant triangles. It is worth noting that such an approach works with meshes at pixel resolution and is thus, at least in theory, able to select high resolution surface parts and could avoid averaging with low resolution surfaces. In practice Mesh Zippering is fragile and fails in the presence of noise and outliers.

Using basis functions for surface reconstruction is a common approach, e.g., for rendering of atomic structures [Blinn, 1982] or in the area of mesh-free particle-based simulation [Yu and Turk, 2013]. A scalar field is defined as the sum of radially symmetric or anisotropic basis functions, possibly with finite support, and triangulated or rendered at a fixed isovalue. Radial Basis Functions (RBFs) have been used for surface reconstruction from (oriented) point clouds [Turk and O'Brien, 1999] but

their work is limited to small problems and closed surfaces. Another inherent difficulty lies in defining off-surface constraints to avoid the trivial solution. Although advances made RBFs much more tractable to real world data in terms of size and handling of noise [Carr et al., 2001], RBF fitting is global in nature and a large linear system of equations must be solved to obtain the parameters of the basis functions. Similarly, Calakli and Taubin [2011] present a variational approach to reconstruct a smooth signed distance function which requires the global solution of a linear system of equations.

A local approach is presented by Ohtake et al. [2003] who fit local shape functions to oriented points and employ weighting functions to blend together the local representations. The approach requires parameters such as the support radius for fitting the local shape functions and an error threshold that controls the refinement of the hierarchal decomposition. All of these parameters, as well as the choice of the local shape functions, depend on the density, redundancy and noise characteristics of the input samples. Their approach is "multi-scale" in the sense that features are reconstructed at different resolution, however, multi-scale input samples are not considered. The method is related to ours in that it constructs the implicit function as a weighted sum of local functions. In contrast, their functions fit multiple points using local shape priors over an octree hierarchy, whereas our functions are defined on a per-sample basis. Shen et al. [2004] present an approach based on an implicit moving least-squares formulation. One key distinction of [Ohtake et al., 2003] is that not only point constraints are considered when fitting the input data: Integrated constraints are used over the polygons which allows the method to either interpolate or approximate polygonal data.

Mixed-Scale: Although there exists a wealth of surface reconstruction literature, few authors consider samples at different scales as input. Integrating scale in the reconstruction process allows us to identify and use redundancy to suppress noise, and to distinguish between high and low resolution samples. Given sufficient high resolution information, any amount of additional low resolution information should not degrade the high resolution reconstruction.

Mücke et al. [2011] splat Gaussians for every input sample into a grid to produce a 3D confidence map. They use normalized Gaussians so that every sample contributes the same confidence but, depending on the scale of the sample, distribute the confidence over differently sized regions. The final surface is extracted as the maximum confidence cut through a graph defined by the grid. The downside of this approach is the unsignedness of the map, and the exact maximum of the function cannot be obtained by interpolation. The global graph cut optimization is also a limiting factor. We draw inspiration from this approach in that we also use basis functions whose size change with the sample scale. In contrast, our implicit function is signed, the zero level-set can be triangulated with sub-voxel accuracy, and we do not require any global optimization.

Fuhrmann and Goesele [2011] present a multi-scale depth map fusion method. The distance fields of triangulated depth maps are rendered into a hierarchical signed

distance field and, in contrast to VRIP, only surfaces at compatible scales are averaged. Low resolution information is discarded in regions with sufficiently high resolution information. The final surface is extracted as the zero level-set of the implicit function. Although our work is inspired by the same basic idea of reconstructing multi-resolution data, the approaches are quite different. Where Fuhrmann and Goesele [2011] assume triangulated depth maps with known sensor positions as input, we rely on oriented, scale-enabled surface samples. Instead of a discretized representation of the implicit function both spatially and in scale, our implicit function can be evaluated anywhere without interpolation in scale and space, solely from the input samples. Thus scale selection becomes more flexible and is not limited to neighboring octree levels. Like VRIP, Fuhrmann and Goesele [2011] cannot extract surfaces in regions without data. Our implicit function extends beyond the input samples to some degree, which enables us to fill small holes and obtain more complete reconstructions. Finally, our isosurface extraction does not require a global Delaunay tetrahedralization, and is thus more efficient and produces meshes with fewer output triangles.

5.3 Floating Scale Implicit Function

In this section we describe the choice of our implicit function. We assume that N input samples are given and equipped with a position $\mathbf{p}_i \in \mathbb{R}^3$, a normal $\mathbf{n}_i \in \mathbb{R}^3$, $\|\mathbf{n}_i\| = 1$, and a scale value $s_i \in \mathbb{R}$. Optional attributes are the sample's confidence $c_i \in \mathbb{R}$ and a color $\mathbf{C}_i \in \mathbb{R}^3$. We will treat color reconstruction only as subordinate aspect of our work.

In the first step an implicit function $F(\mathbf{x}) : \mathbb{R}^3 \mapsto \mathbb{R}$ is defined as the weighted sum of basis functions f_i. Every sample in the input set contributes a single basis function which is parameterized by the sample's position and normal, as well as its scale value. This step does not require any preprocessing and F can readily be evaluated. The final surface is then given as the zero level set of F. In order to make the approach computationally tractable, the basis function weights w_i are compactly supported such that only a small subset of all samples need to be evaluated to reconstruct F at a position $\mathbf{x} \in \mathbb{R}^3$. Due to the compact support of the basis functions, the set $\{\mathbf{x} \mid F(\mathbf{x}) = 0\}$ essentially defines a surface everywhere beyond the support of the samples. We therefore only consider the zero level set inside the support where the weight function W is strictly positive, i.e.

$$\{\mathbf{x} \mid F(\mathbf{x}) = 0 \wedge W(\mathbf{x}) > 0\}. \tag{5.1}$$

5.3.1 Implicit Function

Like many approaches in literature, we reconstruct a signed implicit function which is positive in front of and negative behind the surface (similar to a signed distance

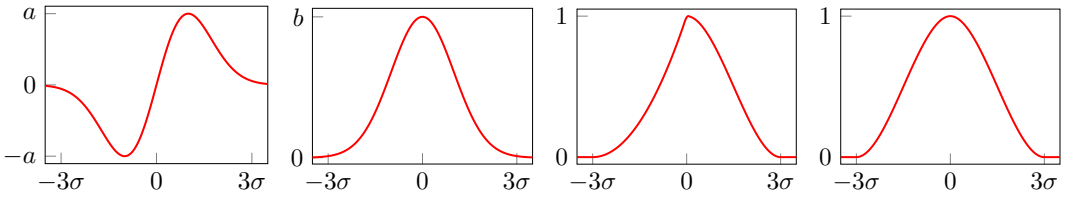

Figure 5.3: The 1D components of the basis function f_x, f_y, and weight function w_x and w_y. The peaks are $a = 1/\sigma e^{-0.5}$ and $b = 1/\sigma\sqrt{2\pi}$.

function). This function F is defined as a weighted sum of basis functions:

$$F(\mathbf{x}) = \frac{\sum_i c_i w_i(\mathbf{x}) f_i(\mathbf{x})}{\sum_i c_i w_i(\mathbf{x})} \qquad W(\mathbf{x}) = \sum_i c_i w_i(\mathbf{x}) \tag{5.2}$$

Function f_i and weight w_i are parameterized by the ith sample position $\mathbf{p}_i$, normal $\mathbf{n}_i$ and scale s_i. The optional confidence c_i essentially scales the weight function and can easily be omitted by setting it uniformly to $c_i = 1$. In the following, without loss of generality, we define f_i and w_i as a one parameter family of functions depending only on the scale s_i of the sample. The position $\mathbf{p}_i$ and normal $\mathbf{n}_i$ are considered by translating and rotating the input coordinate $\mathbf{x}$

$$\mathbf{x}_i = \mathtt{R}_i \cdot (\mathbf{x} - \mathbf{p}_i) \tag{5.3}$$

with rotation matrix $\mathtt{R}_i = R(\mathbf{n}_i)$ such that $\mathbf{x}$ is transformed into the local coordinate system (LCS) of sample i. The LCS is defined such that the sample's position is located in the origin and the normal coincides with the positive x-axis. Because the normal defines the LCS only up to a one dimensional ambiguity it is important that the basis and weight functions f_i and w_i are defined in a *rotation invariant* manner, such that the reconstruction is invariant to the choice of the LCS orthogonal to the normal. Given a rigid transformation T and reconstruction operator $\mathcal{R}$ acting on a point set P, this property ensures that $T(\mathcal{R}(P)) = \mathcal{R}(T(P))$.

5.3.2 Basis Function

Similar to Mücke et al. [2011], we use basis functions that, for every sample, contribute the same "confidence", or volume, to the implicit function. Depending on the scale of a sample, the volume is distributed over differently sized regions. As basis function f we use the derivative of the Gaussian f_x in the direction of the normal with $\sigma = s_i$ set to the scale of the sample. (We flip the sign of the function because it is defined to be *positive* in front of the surface, i.e. in the direction of the *positive* x-axis.) Normalized Gaussians f_y, f_z are used orthogonal to the normal in y and z direction.

$$f_x(x) = \frac{x}{\sigma^2} e^{\frac{-x^2}{2\sigma^2}} \qquad f_y(x) = f_z(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} \tag{5.4}$$

Figure 5.3 illustrates the function components in 1D. This yields the basis function

$$f(\mathbf{x}_i) = f_x(x)f_y(y)f_z(z) = \frac{x}{\sigma^4 2\pi} \cdot e^{\frac{-1}{2\sigma^2}(x^2+y^2+z^2)} \tag{5.5}$$

The function is rotation invariant around the normal because $f_y f_z$ can be rewritten in terms of the distance $\sqrt{y^2+z^2}$ to the normal. The integral of the function's absolute value is 1 and thus every basis function contributes the same volume to the implicit function:

$$\iiint |f(\mathbf{x}_i)|\mathrm{d}\mathbf{x}_i = \int |f_x(x)|\mathrm{d}x \int f_y(y)\mathrm{d}y \int f_z(z)\mathrm{d}z = 1 \tag{5.6}$$

Since f_y and f_z are normalized Gaussians, their integrals are 1 by definition. We integrate the absolute function $|f_x|$ because the point-symmetric parts cancel each other out. $|f_x|$ does not require explicit normalization and $\int |f_x| = 1$. Figure 5.4 (left) illustrates the function in 2D.

5.3.3 Weighting Function

In the following we design a polynomial weighting function w that has compact support, falls smoothly off to zero and gives more weight to the regions in front of the surface. The justification behind this is related to free space constraints and occlusions as discussed by Curless and Levoy [1996] and Vrubel et al. [2009]: If a sample has been observed, the existence of a surface between the observer and the sample is not possible. Behind the sample, however, we cannot be sure of the existence of a surface and want to reduce the weight quickly. We observe that $f(\mathbf{x}_i)$ has negligible influence beyond 3σ, and thus we chose 3σ as the point beyond which the weighting function vanishes. The weighting function

$$w(\mathbf{x}_i) \quad = \quad w_x(x) \cdot w_{yz}(\sqrt{y^2+z^2}) \tag{5.7}$$

is composed of a non-symmetric component in x-direction

$$w_x(x) \quad = \quad \begin{cases} \frac{1}{9}\frac{x^2}{\sigma^2} + \frac{2}{3}\frac{x}{\sigma} + 1 & x \in [-3\sigma, 0) \\ \frac{2}{27}\frac{x^3}{\sigma^3} - \frac{1}{3}\frac{x^2}{\sigma^2} + 1 & x \in [0, 3\sigma) \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

and a rotation invariant component in y- and z-direction

$$w_{yz}(r) \quad = \quad \begin{cases} \frac{2}{27}\frac{r^3}{\sigma^3} - \frac{1}{3}\frac{r^2}{\sigma^2} + 1 & r < 3\sigma \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

$$r \quad = \quad \sqrt{y^2+z^2}. \tag{5.10}$$

Note that the function w_{yz} is the positive domain of w_x where x is replaced with the distance to the normal r. The individual 1D components of the weighting function are illustrated in Figure 5.3, and a 2D illustration is shown in Figure 5.4 (right).

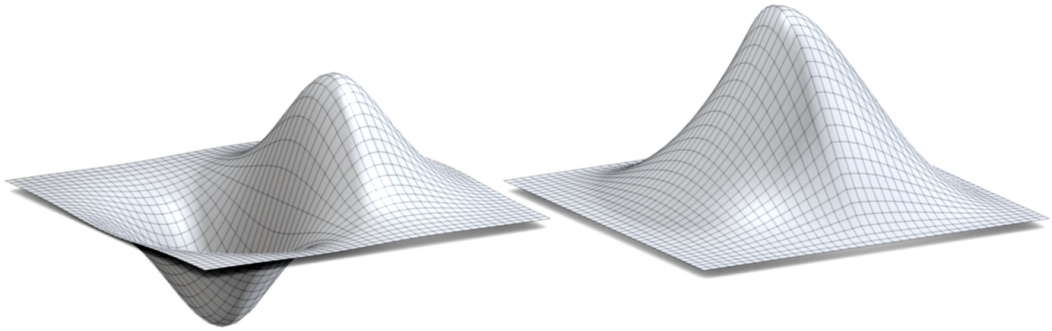

Figure 5.4: The basis and weighting functions in 2D in the interval $[-3\sigma, 3\sigma]^2$. Left: $f(x, y) = f_x(x)f_y(y)$. Right: $w(x, y) = w_x(x)w_y(y)$.

5.4 Analysis in 2D

There are many possible choices for both basis and weighting function. We chose the Gaussian family of functions as basis function which empirically provides excellent approximation and extrapolation behavior. We will now demonstrate these properties of the implicit function on simple synthetic 2D datasets as well as real world data. In Section 5.6 we discuss alternative choices for both the basis function and weighting function.

We visualize the implicit function using a color mapping where positive values are colored green and negative values are blue. Bright colors correspond to small values of F (near and also far from the isosurface) and darker colors to large values, such that the isosurface is directly visible in the images. A gray color is used outside of the support, where W is zero. Samples are indicated in red.

5.4.1 Synthetic Data

When designing a reconstruction algorithm, we are concerned with the interpolation, extrapolation and approximation characteristics of the reconstruction operator. On the one hand the isosurface should pass through the input samples, in particular if the points are sparse and accurate. The implicit function should gracefully fill the gaps between the samples using smooth extrapolation. On the other hand, if many redundant and noisy point samples are available, it should approximate the samples and average out the noise instead of over-fitting the data. The presented formulation of the implicit function automatically adapts to the data as demonstrated by the following 2D experiments.

We provide 2D point samples (of a curve) with normals. In the first experiment the scale is computed for each sample as the average distance to the two nearest neighbors of the sample. We then multiply the computed scale with several factors, see Figure 5.5. With a small factor, the function interpolates the samples but extrapolation becomes less smooth. With an increasing factor, the reconstruction will approximate the points and provide a smoother extrapolation, but a less accurate

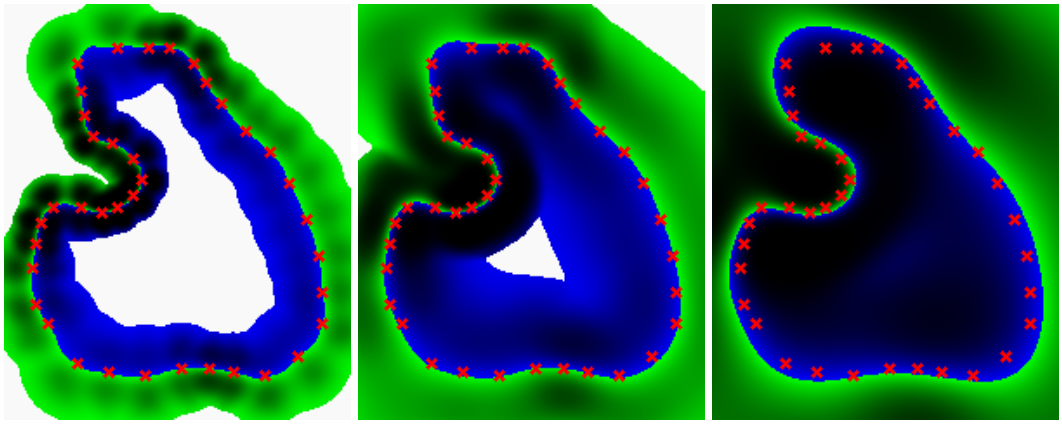

Figure 5.5: Reconstructions with increasing scale factor. The larger the factor the more approximative is the reconstruction. Scale factors are 0.5 (left), 1.0 (middle) and 2.0 (right).

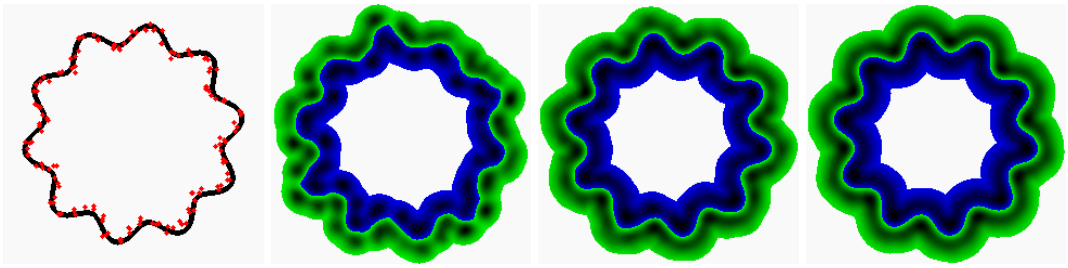

Figure 5.6: Reconstructions with increasing redundancy converging towards the original function. The left image shows the original function and 150 noisy samples. The reconstructions have been computed from 50, 500 and 5000 noisy samples, respectively.

interpolation.

In Figure 5.6 we consider the sampling of a function with added noise to positions and normals. The noise is uniform and about 5% of the bounding box of the samples. As we increase redundancy (adding more noisy samples while keeping the scale of the samples constant), the technique starts to increasingly approximate the data, reducing the noise, until the reconstruction converges towards the original function.

5.4.2 Real-World Data

In practical cases the noise characteristics of the input data change considerably depending on how the samples have been acquired. Our intention is to demonstrate the ability of our reconstruction operator to handle both clean and extremely noisy datasets without tuning any parameters (such as noise characteristics or the octree level).

In the following experiments we use the Stanford Bunny and the Middlebury Temple dataset (see Section 5.6 for details how these datasets were created). For

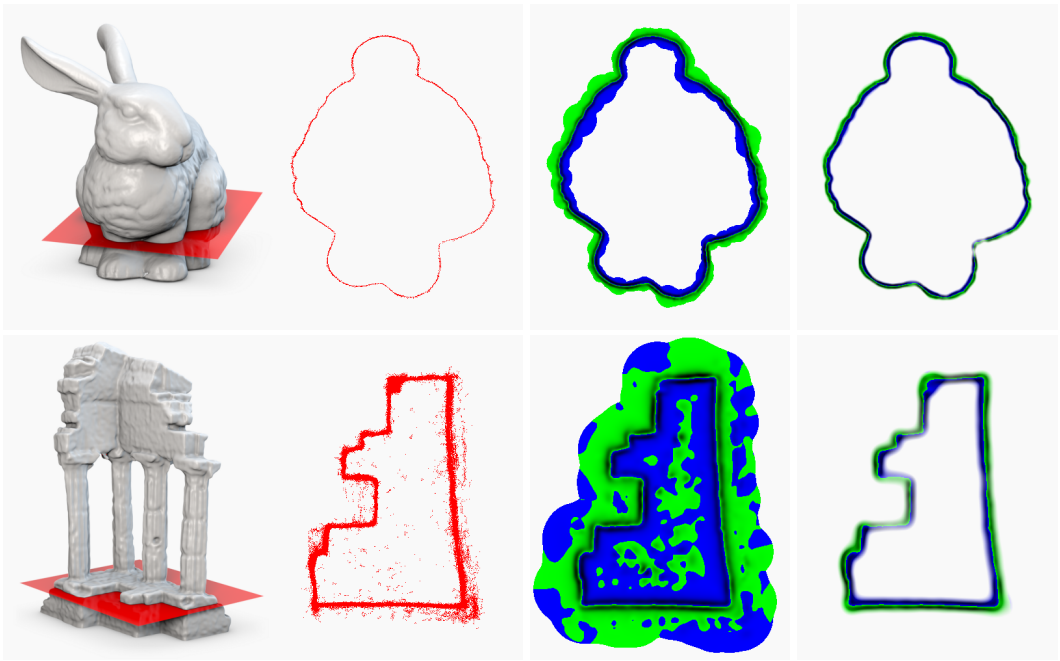

Figure 5.7: Reconstruction with real world data. Top row: Slice of the Stanford Bunny. Bottom row: Slice of the Middleburry Temple. Illustration of the slice and the 2D input point set (left), normalized implicit function (middle) and the weighted implicit function (right).

each dataset, we intersect the set of samples with a plane and select only samples whose distance to the plane is below a threshold. This yields 2D datasets with 2690 samples for the Bunny, and 157275 samples for the Temple, see Figure 5.7. While the Stanford Bunny contains clean, range scanned samples, the Middlebury Temple is a very noisy multi-view stereo (MVS) reconstruction with many outliers. The presence of isolated outliers as well as noise in both normals and sample positions lead to many isovalue crossings further away from the true surface. The weighting function, however, indicates which parts of the implicit function are important. In practice, only isosurfaces above a certain weight are extracted, which removes spurious isolated components. Note that this weight threshold is not a parameter to our reconstruction operator. In fact, we postpose cleaning the geometry until after the surface mesh has been extracted.

5.5 Sampling the Implicit Function

In this section we detail how the implicit function F is efficiently evaluated to extract the isosurface $F(\mathbf{x}) = 0$. All input samples are first inserted into an octree data structure according to their scale value. The resulting octree hierarchy prescribes a sampling of F by considering the positions in the corners of the octree leaf nodes. The implicit function is then evaluated at these positions. We call these sample

positions *voxels* to distinguish from the input sample points. Finally, the isosurface is extracted from the octree using a variant of the Marching Cubes algorithm.

5.5.1 Octree Generation

In order to avoid aliasing when sampling the implicit function or evaluating the function too far from the isosurface, we set bounds on the voxel spacing according to the samples' scale values. Recall that sample i has scale value s_i and the radius of the sample's support is $3s_i$. We impose

$$S_\ell \leq s_i < S_{\ell-1} \quad \Leftrightarrow \quad S_\ell \leq s_i < 2S_\ell \tag{5.11}$$

where ℓ is the octree level at which the sample will be inserted, and S_ℓ is the side length of an octree node at level ℓ (i.e., the voxel spacing). This forces a sample to be inserted into an octree node with a side length S_ℓ of at most s_i but usually smaller:

$$1/2\ s_i < S_\ell \leq s_i. \tag{5.12}$$

We start with an empty octree without nodes. The first sample i is inserted in a newly created root node with a side length of s_i and centered around the sample's position $\mathbf{p}_i$. When inserting subsequent samples, three cases can occur:

1. The new sample is outside the octree. In this case the octree is iteratively expanded in the direction of the new sample until the new sample is inside the octree. The sample is then inserted using cases 2 or 3.

2. The new sample's scale is larger than the scale of the root node. Again, octree expansion is used to create new, larger root nodes until the root has a scale according to (5.12).

3. The new sample's scale is smaller than the scale of the root node. In this case the tree is traversed, possibly creating new nodes, until a node with a scale according to (5.12) is reached.

Once a node is determined, the sample is inserted into that node.

5.5.2 Evaluating the Implicit Function

After inserting all samples in the octree, the octree is prepared for evaluation of the implicit function. We enforce that nodes can be classified into either *inner nodes* or *leafs*. Inner nodes have all eight children allocated, and leafs have no children allocated. The current octree, however, has *mixed nodes* where only some of the children are allocated. We make the octree *regular* by allocating the remaining unallocated children of nodes which are not leafs. This creates new leafs and eliminates mixed nodes.

A list of voxels (points at which the implicit function is evaluated) is created by iterating all leaf nodes. Each leaf node generates eight voxels in the corners of the

node. This is a *primal sampling* as opposed to a *dual sampling* where voxels are positioned in the center of the node. Since neighboring leaf nodes share common voxels, every voxel is identified with a unique ID and inserted into a unique set. The implicit function is then evaluated at the voxel positions.

In order to evaluate the implicit function at position $\mathbf{x}$, we design an efficient query on the octree that selects only samples which influence the implicit function at $\mathbf{x}$: The octree is recursively traversed and for every node a check is performed if the node can possibly contain a sample which influences $\mathbf{x}$. From Equation (5.11) we know that node N contains samples with a scale of at most $2S_N$, where S_N is the side-length of N. Thus, $\mathbf{x}$ cannot be influenced by any sample in N if

$$\|\mathbf{x} - \text{center}(N)\| - \sqrt{3}\frac{S_N}{2} > 3 \cdot 2S_N. \tag{5.13}$$

The left side of the inequality is the worst case (smallest) distance from $\mathbf{x}$ to any point in the node, and the right side is the largest possible influence radius of a sample in N, i.e. 3 times the largest sample scale $2S_N$. If the inequality holds, the node can be skipped without descending into child nodes. Otherwise, all samples i in the node are considered if $\|\mathbf{x} - \mathbf{p}_i\| < 3s_i$. The implicit function $F(\mathbf{x})$ can then be evaluated according to Equation (5.2) using all selected samples that influence $\mathbf{x}$.

Scale Selection: Limiting the number of samples for evaluating the implicit function will have two effects: It speeds up the algorithm, but more importantly, it can actually improve the quality of the reconstruction. On the one hand, the error to the ground truth geometry is decreased by exploiting redundancy to account for the sample noise. On the other hand, the surface error is increased by mixing samples with different scales: As the formation of a sample usually happens through some kind of integration process over a surface area, every sample corresponds to a low-pass filtered version of the original surface depending on the scale of the sample [Klowsky et al., 2012]. Mixing high and low resolution samples will thus have the effect of degrading the isosurface towards low-pass filtered geometry. This is demonstrated with the synthetic experiment in Figure 5.8 (see also the supplemental material).

Our approach to this problem is based on the idea of balancing the positive effect of redundancy (Figure 5.6) with the negative effect of mixing high and low resolution samples (Figure 5.8). These two properties are orthogonal to each other: Noise reduction improves precision along the surface normal whereas low resolution samples have an impact along the tangent of the surface. Making a tradeoff between the two is not straightforward. Fuhrmann and Goesele [2011] discard low resolution samples by locally selecting the highest supported resolution from the discretized scale-space representation. Similarly, we also discard low resolution samples. We do, however, not discretize scale and can therefore choose a continuous cut-off scale using the following heuristic.

To evaluate the implicit function at voxel $\mathbf{x}$, consider the set of samples whose (compact) support overlaps with $\mathbf{x}$. We now determine a cut-off scale value $s_{\max}$

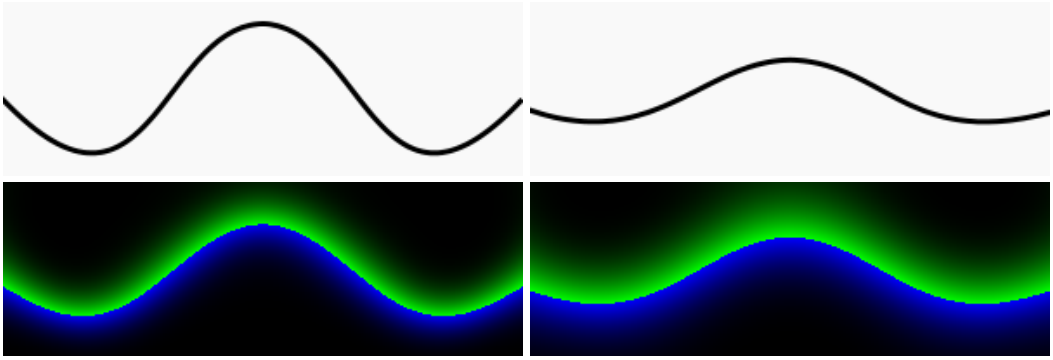

Figure 5.8: Synthetic experiment: The top row shows the high resolution (HR) surface (left) and a low-pass filtered version of the high resolution (LR) surface (right). The bottom row shows the result of mixing 100 HR samples with 1000 LR samples (left) and mixing 100 HR samples with 10000 LR samples (right), causing the isosurface to degrade towards the low-pass filtered geometry.

and only consider samples i with $s_i < s_{\text{max}}$ to reconstruct the implicit function at $\mathbf{x}$. Conceptually, we define $s_{\text{max}} = s_{\mathbf{x}} \cdot f_{\text{noise}}$, where $s_{\mathbf{x}}$ is a reference scale and f_{noise} can be chosen according to the noise properties of the data. In our implementation, the reference scale $s_{\mathbf{x}}$ is chosen as a robust 10th percentile of those scale values affecting $\mathbf{x}$. (Finding the nth percentile is a linear operation and does not require sorting all samples.) We set $f_{\text{noise}} = 2$ in all of our experiments.

5.5.3 Isosurface Extraction

At this point, the samples are no longer required and the isosurface can be extracted from the implicit function defined at the octree voxels. This is, however, more complicated than with a regular grid. In the regular case, each cube can be processed individually using Marching Cubes [Lorensen and Cline, 1987] and the result is guaranteed to be watertight. In the case of an octree, however, different decisions are made on either side of a cube face (because of depth disparity in the octree), which leaves cracks in the surface. We use the isosurface extraction algorithm proposed by Kazhdan et al. [2007] which yields a crack-free and highly adaptive mesh directly from the octree hierarchy.

The resulting surface contains many degenerate triangles, which is typical for Marching Cubes-like algorithms. To obtain a well-behaved mesh we apply a simple cleanup procedure, see Figure 5.9. We first identify needle triangles, which are erased by collapsing the short edge. A check that the normals of adjacent triangles do not change too much prevents topological artifacts. Afterwards cap triangles are removed by collapsing vertices with only three adjacent faces. A final pass of needle removal is performed as new needles may be created by the previous operation. This simple procedure usually reduces the number of triangles in the mesh by about 40%.

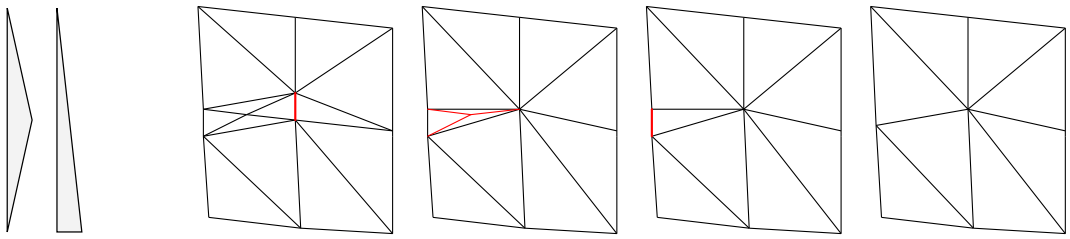

Figure 5.9: Two types of degenerate triangles, caps and needles (left). The mesh cleanup procedure (right) with the initial mesh, needles cleanup, caps cleanup, and another needles cleanup. The edges to be collapsed are shown in red.

5.5.4 Color Reconstruction

We use a simple approach to evaluate a second implicit function that yields a color value for every position $\mathbf{x}$. The implicit function has form (5.2) but uses simpler basis functions. f_i is replaced with the constant sample color $\mathbf{C}_i$ and the weight function w_i is replaced with a narrow 3D Gaussian with $\sigma = {}^1\!/_5 \cdot s_i$. Here, σ is chosen so small to avoid blurring the color and to obtain a crisp texture. Although this weighting function does not have compact support, the weight evaluated at $\pm 3 s_i$ away from the sample is in the order of 10^{-10} and thus negligible.

5.6 Results

We perform a thorough evaluation of our approach on three types of datasets. In Section 5.6.1 we compare our results on controlled data with Mesh Zippering [Turk and Levoy, 1994] and VRIP [Curless and Levoy, 1996]. We use the Middlebury benchmark in Section 5.6.2 to rank our reconstruction on multi-view stereo data. Finally, in Section 5.6.3, we show the performance of our algorithm on mixed-scale data. For all datasets we also compare with the quasi-standard reconstruction algorithm, (Screened) Poisson Surface Reconstruction (PSR) [Kazhdan and Hoppe, 2013]. Instead of comparing to an exhaustive number of algorithms, we limit ourselves to PSR as one representative algorithm that uses point density to estimate per-sample scale in the reconstruction process. Extensive comparison of PSR with other algorithms has been performed by Kazhdan and Hoppe [2013].

5.6.1 Range Scanner Data

The availability of both range data and final reconstructions in the Stanford Scanning Repository allow us to qualitatively compare our reconstructions with those from the website performed with Mesh Zippering [Turk and Levoy, 1994] and VRIP [Curless and Levoy, 1996]. We obtained the input point sets to our system by aligning the range data using transformations provided by the Stanford Scanning Repository. This yields one mesh per range scan in the global coordinate system. Normals are

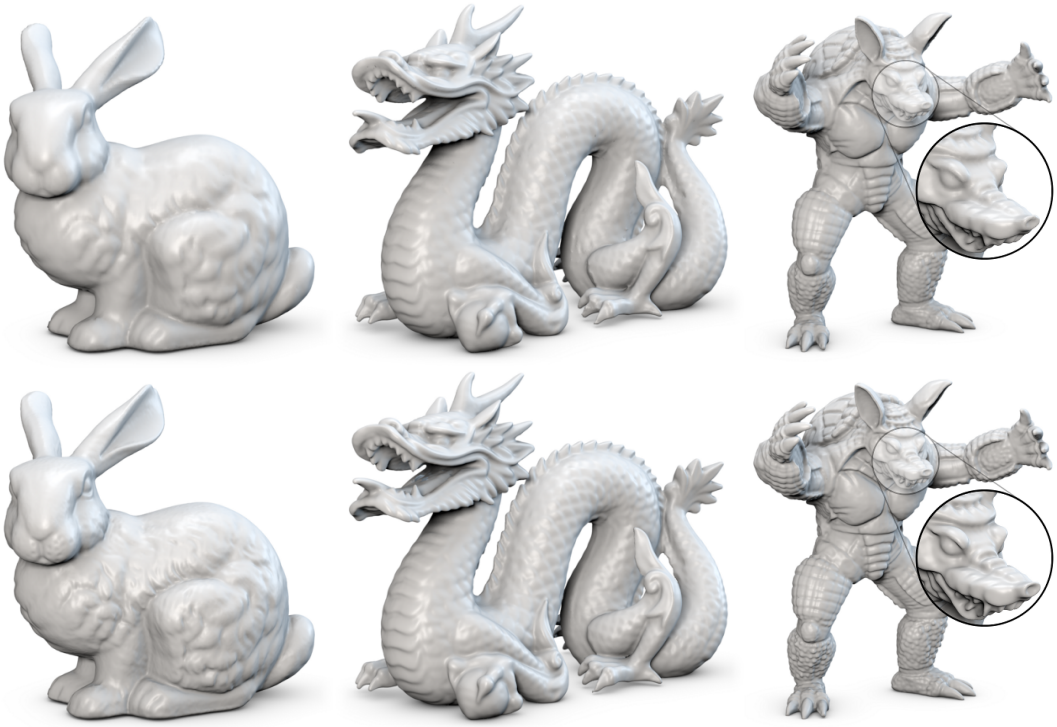

Figure 5.10: Reconstruction of the Stanford models. The left column shows the Bunny reconstruction using Mesh Zippering [Turk and Levoy, 1994] (top) and our reconstruction (bottom). The middle and right column show the Dragon and Armadillo reconstructed with VRIP [Curless and Levoy, 1996] (top) and with our algorithm (bottom). Our algorithm reveals more detail on all models.

computed for every vertex from the adjacent triangles. The per-vertex scale value is set to the average length of all edges emanating from the vertex. Connectivity information of the range scans is discarded afterwards.

Stanford Models: Figure 5.10 compares several reconstructions from the Stanford Scanning Repository with our own reconstructions. The Stanford Bunny dataset contains 10 range scans, the Dragon 71 and the Armadillo a total of 97 range scans. Our algorithm is able to make use of the redundancy in the data without blurring the result, which reveals details unavailable in the Mesh Zippering and VRIP reconstructions. The surfaces created with PSR look visually very close to our reconstruction, so we omit a visual comparison here. Instead, we provide a quantitative evaluation in Table 5.1. For this evaluation we split the input point set and use 90% of the samples for reconstruction, and the remaining 10% of the samples to evaluate the RMS error and mean distance to the reconstructed surface. Our method shows performance on par with PSR on these datasets.

Incomplete Data: We now demonstrate the behavior of our method on data with holes and boundaries. Due to the local nature of the basis functions, the implicit

Figure 5.11: Reconstruction behavior with incomplete data. The input point set from a single range image (left) and our reconstruction leaving holes in regions with insufficient sampling (right).

function is undefined beyond the support of the samples. Although the implicit function is able to close small gaps in the sampling of the surface, it does not close larger holes. Figure 5.11 illustrates this behavior on a single range scan of the Stanford Bunny.

Michelangelo's David: To showcase the scalability of our approach, we reconstruct Michelangelo's David provided by the Stanford 3D Scanning Repository, see Figure 5.12. The dataset is a VRIP reconstruction of non-rigidly aligned range scans and contains a total of 472 million input samples. Although our reconstruction required a considerable amount of memory (114 GB RAM) and processing time (4 hours on a machine with 8 AMD Opteron Quad-Core processors), we were not able to process the data with PSR within a memory limit of 250 GB at any octree level

Error comparison on Stanford Datasets			
	Bunny	Dragon	Armadillo
RMS (PSR)	1.419789	2.950294	5.527238
RMS (ours)	1.394920	2.930433	5.439365
Mean (PSR)	0.970039	1.560911	1.706402
Mean (ours)	0.911296	1.512578	1.676785

Table 5.1: Quantitative evaluation on Stanford datasets. 90% of the samples are used for reconstruction, the remaining 10% for evaluating the mean and RMS distance to the reconstructed surface. The measurements are in units of 10^{-4}.

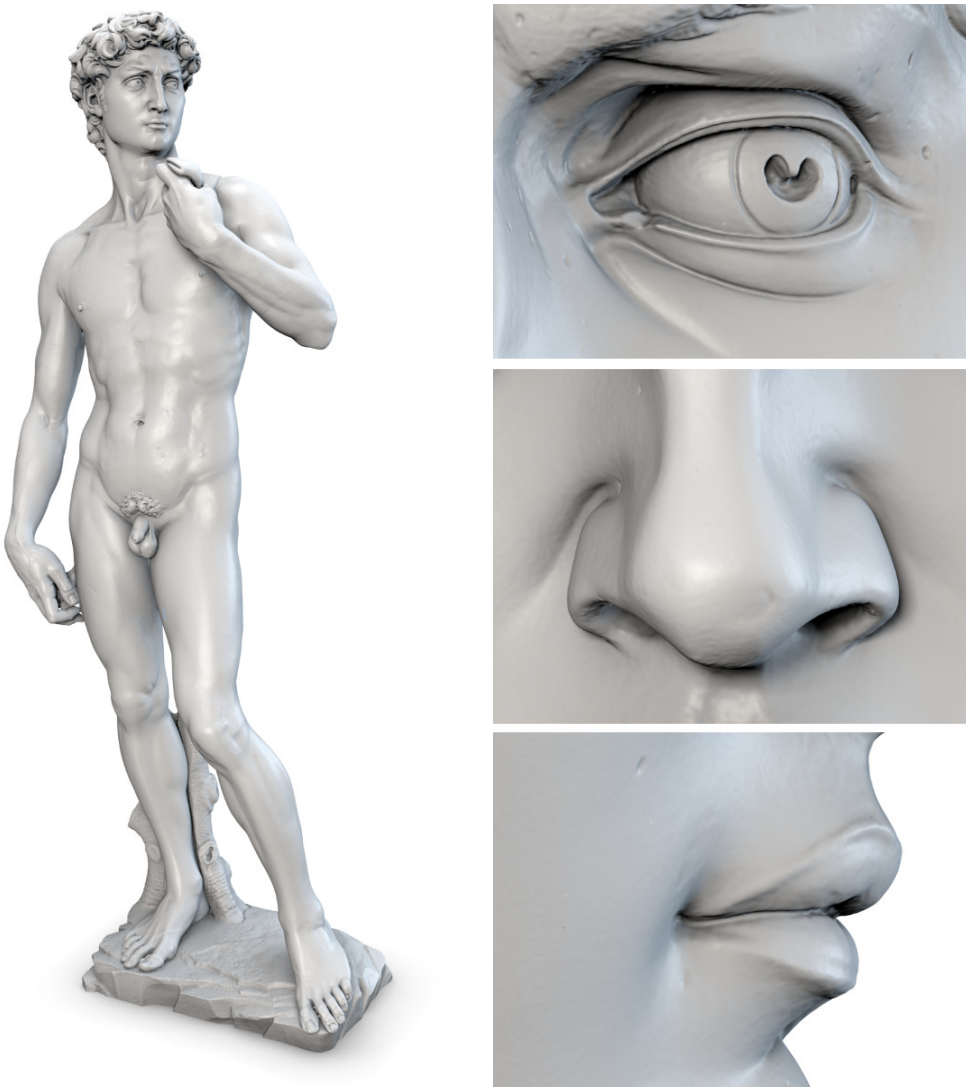

Figure 5.12: Reconstruction of Michelangelo's David from 472M input samples. The dataset is kindly provided by the Stanford 3D Scanning Repository.

larger than 11. We succeeded in running Streaming PSR [Bolitho et al., 2007] at a level of 14, which took about a day, but still resulted in a very low resolution output mesh.

5.6.2 Multi-View Stereo Data

Next, we evaluate our approach on multi-view stereo (MVS) data. We produce the input samples to our algorithm in the following way: A depth map is computed for every input image using the freely available MVS implementation of Goesele et al. [2007]. Similar to the range scanner data, scale is computed for every vertex in the triangulated depth maps as the average length of all edges emanating from that vertex. But in contrast to the scanner data, every pixel in the depth map actually

Figure 5.13: Reconstruction of the Middlebury Temple. The Poisson Surface Reconstruction (left), our colored (middle) and shaded reconstruction (right).

corresponds a surface region larger than the pixel: Every depth value is the result of a photo-consistency optimization on patches of a certain extent, which has a (low-pass) filtering effect on the reconstructed surface [Klowsky et al., 2012]. We used a patch size of 5x5 pixels and, empirically, found that multiplying the scale by 2.5 (i.e. the "radius" of the patch) yields good results. Finally, the union of all vertices from all depth maps is used as the input point set.

The *Temple Full* dataset from the Middlebury benchmark [Seitz et al., 2006] contains 312 images. All MVS depth maps yield a total of 23 million input samples. Our reconstruction is available as *Fuhrmann-SG14* on the Middlebury evaluation page for quantitative comparison. (Note that the final geometry does not only depend on our reconstruction technique, but also on the MVS algorithm). We visually compare our result with PSR at an octree depth of 10 in Figure 5.13. The PSR reconstruction looks slightly sharper around the edges but also has some geometric artifacts. In contrast to PSR, our algorithm does not require any parameter tuning.

5.6.3 Multi-Scale MVS Data

Our algorithm gracefully handles both clean and uniform scale datasets, but excels in handling multi-resolution datasets. In the following we perform an evaluation on a multi-scale multi-view stereo dataset where images are taken at various distances to the subject. This yields depth maps with vastly different sampling rates of the surface. In contrast to algorithms using point density, our algorithm produces sharp geometry even in the presence of many low resolution samples, and smooth results in low resolution regions. We use PSR as a representative algorithm to demonstrate the shortcomings of traditional methods on multi-scale data. We then compare our results with other multi-scale approaches.

We first register the input images using Structure from Motion software. Similar

Figure 5.14: Multi-scale reconstruction of the *Elisabeth* dataset. The top row shows our reconstruction with color (left), with shading (middle) and with false coloring of the scale (right). The bottom row shows 5 of 205 input images with varying scale.

to the *Temple Full* dataset, we reconstruct dense depth maps using the MVS implementation by Goesele et al. [2007] and use the samples of all depth maps as the input to our algorithm.

Elisabeth Dataset: Due to technical limitations (memory consumption and processing time) with PSR, we prepared a smaller dataset called *Elisabeth* to perform the comparison. The dataset contains high resolution regions with detailed carvings and reliefs, as well as regions captured at a much lower resolution, see Figure 5.14. Although PSR at level 9 produces a smooth result in the low resolution region, it cannot reconstruct the high resolution details. PSR at level 11 reconstructs the fine details but produces a poor result in low resolution regions: It cannot reliably detect redundancy and, due to the too large octree level, reconstructs the noise in the data. A visual comparison of the reconstruction can be found in Figure 5.15.

Fountain Dataset: We now compare our algorithm on an MVS dataset with a much larger extent. We captured 384 photos of an old fountain yielding a total of 196 million input samples (about half the size of the David dataset). While most of the scene is captured in lower resolution, one of the two lion heads is captured with many close-up photos. Figure 5.16 shows some input images as well as an overview of the whole reconstruction spanning more than two orders of magnitude differences in scale. Figure 5.17 shows some geometric details on the fountain.

We compare our reconstruction with two other mixed-scale approaches, namely the work by Mücke et al. [2011] (*SurfMRS*) and Fuhrmann and Goesele [2011] (*DMFusion*) in Figure 5.18. Due to excessive use of memory with SurfMRS on the full point set, we cropped and reconstructed only the detailed region around the fountain for

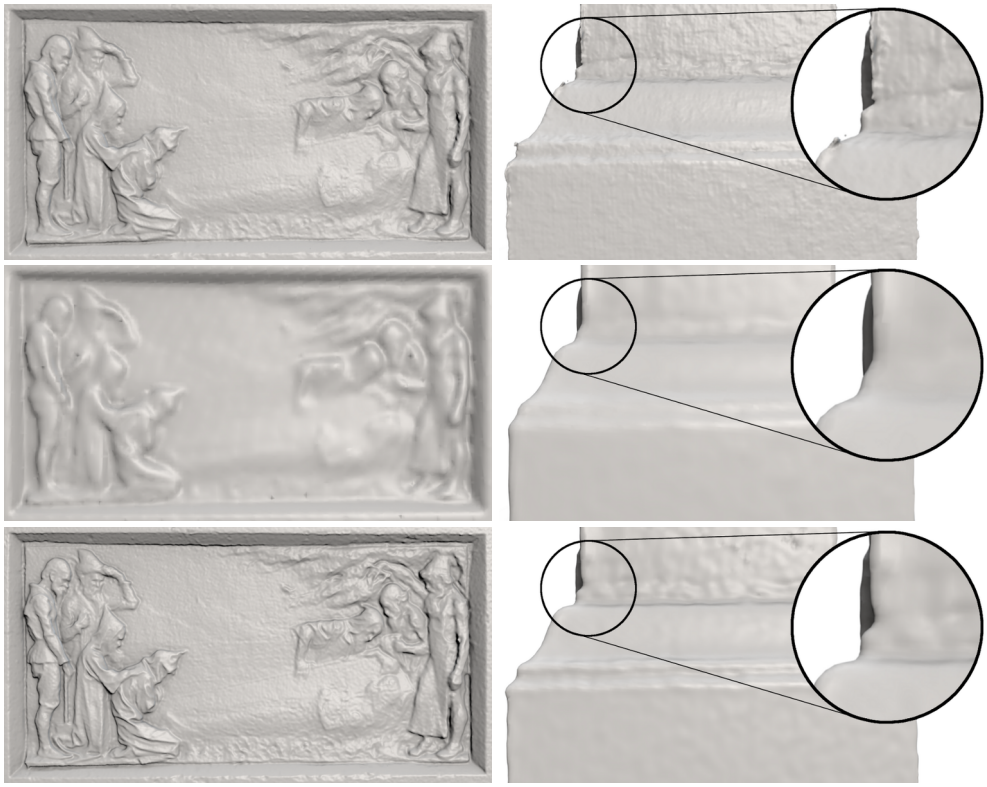

Figure 5.15: Comparison with PSR on the *Elisabeth* dataset. *Top row:* Reconstruction with PSR at level 11, which reconstructs details (left) but produces noise in low resolution regions (right). *Middle row:* PSR at level 9 smoothly reconstructs low resolution regions but fails on the details. *Bottom row:* Our method reproduces both high- and low resolution regions appropriately.

the comparison. Many details are lost in the SurfMRS reconstruction because the graph cut optimization often cuts through details, such as the teeth and the spout at the mouth. While DMFusion leaves small holes in the surface, our algorithm is able to deliver a watertight result. Although all algorithms managed to properly distinguish between low and high resolution regions, our algorithm achieves a more detailed yet smoother reconstruction.

5.6.4 Alternative Basis and Weighting Functions

In the following we present alternative basis and the weighting functions. In particular, we replace our basis function with signed distance ramps similar to VRIP [Curless and Levoy, 1996], and we evaluate the radially symmetric B-spline used in the work of Ohtake et al. [2003] as weighting function.

Figure 5.16: The *Fountain* dataset. The top row shows the full colored reconstruction of the site. The bottom row shows 4 of 384 input images depicting the whole site, the fountain and two details on the fountain.

Basis Function: An approximate signed distance function for sample $\mathbf{p}_i$ with surface normal $\mathbf{n}_i$ is given by

$$f_i(\mathbf{x}) = \langle \mathbf{x} - \mathbf{p}_i \mid \mathbf{n}_i \rangle. \tag{5.14}$$

Because this function does not attenuate orthogonal to the surface normal, it results in a smoother implicit function but less accurate sample interpolation. The integral of the function is unbounded with a constant slope in the direction of the normal, which results in large values if evaluated far away from the surface. This aspect makes the function less useful in multi-scale scenarious because low-resolution samples tend to dominate the implicit function and degrade geometric details. This is demonstrated in Figure 5.19, which shows a high-resolution region of a large multi-scale dataset.

Weighting Function: While we advocate the use of a weighting function with non-symmetric behavior in the direction of the normal, simpler choices are possible. Ohtake et al. [2003] use the compactly supported, radially symmetric, quadratic B-spline $B(\frac{3}{2}\frac{\|\mathbf{x}_i\|}{3\sigma} + 1.5)$ with radius 3σ and centered around the origin. For most datasets this weighting function produces very comparable results. However, similar to Vrubel et al. [2009], the non-symmetric weighting function suppresses more

Figure 5.17: Details in the *Fountain* dataset.

Figure 5.18: Comparison of details in the *Fountain* dataset using SurfMRS (left), DM-Fusion (middle) and our result (right).

artifacts caused by noise and outliers, as demonstrated on the Temple dataset in Figure 5.20.

5.6.5 Runtime Performance

In this section we report runtime and memory performance of our system. Table 5.2 lists datasets with the number of input samples, and the time required for the reconstruction. The reconstruction time is split into sampling the implicit function, which consumes most of the time, and isosurface extraction. We also report the peak memory usage of the system, which is measured as the maximum resident memory size of the process. All benchmarks are performed on a Intel Xeon Dual CPU system with 6×2.53GHz cores per CPU. The reported wall time for evaluating the implicit function uses all cores. Isosurface extraction, however, is limited to a single core.

Figure 5.19: Comparison of reconstructions using signed distance ramps (left) and our basis function using Gaussians (right). The distance ramps are particularly harmful in mixed-scale datasets.

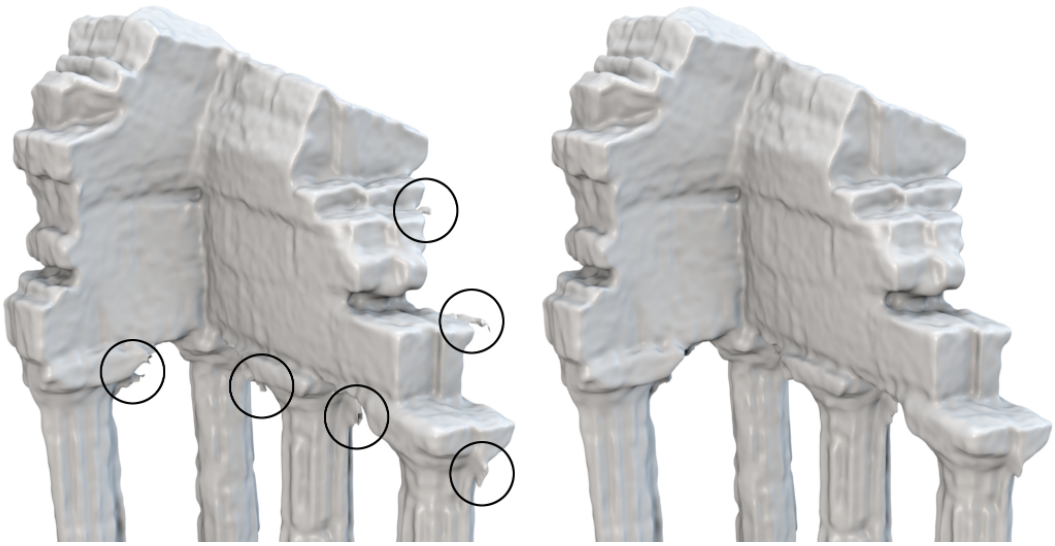

Figure 5.20: Reconstruction using the radially symmetric B-Spline weighting function (left) and our non-symmetric weighting function (right). Our weighting function produces fewer artifacts caused by noise and outliers in the input data.

Dataset Name	Number of Samples	Recon. Time	Peak Memory	Output Vertices
Bunny	362 K	30s + 9s	320 MB	277 K
Dragon	2.3 M	83s + 17s	603 MB	455 K
Armadillo	2.4 M	63s + 13s	553 MB	293 K
David	472 M	247m + 38m	114 GB	81.9 M
Temple	22.8 M	5m + 5s	1.96 GB	176 K
Elisabeth	39.3 M	19m + 1m	4.39 GB	2.3 M
Fountain	196 M	178m + 6m	19.9 GB	10.2 M

Table 5.2: Runtime performance for various datasets. The timings are broken down into implicit function evaluation and surface extraction. The peak memory is measured as the maximum resident memory size of the process.

5.7 Discussion and Conclusion

We presented a point-based surface reconstruction method that considers the scale of every sample and enables an essentially parameter-free algorithm. It can handle both very redundant and noisy as well as controlled datasets without any parameter tuning. This flexibility comes at the price of providing a scale value for every input sample, which is typically easily obtained. The method has been shown to compute highly detailed geometry, gracefully degrades given imperfect input data such as noisy points and normals, outliers, large holes or varying point density. The mathematical concept behind the approach is very simple and will likely inspire more research in this direction. For example, studying the impact of various basis functions on the reconstruction properties can lead to new reconstruction operators.

Our approach requires normals and scale information for every input sample. Several approaches for estimating normals have been proposed, e.g. by Hoppe et al. [1992] and Alliez et al. [2007]. Scale values, however, cannot reliably be inferred without information about the formation of the samples. In the special (but unlikely) case, where the sample density is globally related to the scale of samples (which is assumed in many methods), the scale values can be computed from the sample spacing, for example using the average distance to the k nearest neighbors. If this is not the case, the estimation of scale will fail, and surface reconstruction can produce undesirable results. In particular, the algorithm loses the ability to exploit redundancy for noise reduction and thus reconstructs high frequency noise, as demonstrated in Figure 5.2.

Although our implementation scales well to huge datasets and the runtime performance is competitive with state-of-the-art methods, sampling the implicit function is a time-consuming step because the Gaussians which we use as basis functions are expensive to evaluate. Even though increasing redundancy does not considerably increase memory consumption, it does increase computation time. The reason is that more samples influence each voxel and more basis functions need to be eval-

uated.

Due to the local nature of our algorithm, the implicit function is not defined beyond the support of the samples. Although our approach is able to close small gaps in the surface sampling, it cannot close larger holes and leaves these regions empty. This is suitable for open scenes or geometric objects which are only partially captured. On the other hand, this behavior stands in contrast to many global approaches which perform excellent hole-filling but often hallucinate low resolution geometry in incomplete regions, requiring manual intervention.

We believe that the approach is particularly well suited for out-of-core implementation and distributed reconstruction because of the local nature of our formulation. We would like to investigate this direction in future work. This opens the door for high-quality city-scale surface reconstruction projects, impossible with current state-of-the-art approaches.

Chapter 6

Direct Resampling for Isotropic Surface Remeshing

Abstract

We present a feature-sensitive remeshing algorithm for relaxation-based methods. The algorithm first resamples the reference mesh with an exact vertex budget. The vertex distribution can be either uniform or non-uniform according to a density function. The newly created samples on the mesh surface are triangulated by constructing a mutual tessellation. The algorithm then optimizes the positions of the mesh vertices by building weighted centroidal Voronoi tessellation. We achieve a precise isotropic placement of the samples using Lloyd's relaxation method, but other relaxation schemes are applicable.

The proposed algorithm handles diverse meshes of arbitrary genus and guarantees that the remeshed model has the same topology as the input mesh. The density function can be defined by the user or derived automatically from the estimated mesh curvature. A subset of the mesh edges may be tagged as sharp features to preserve the characteristic appearance of technical models. The new method can be applied to large meshes and produces results faster than previously achievable.

Contents

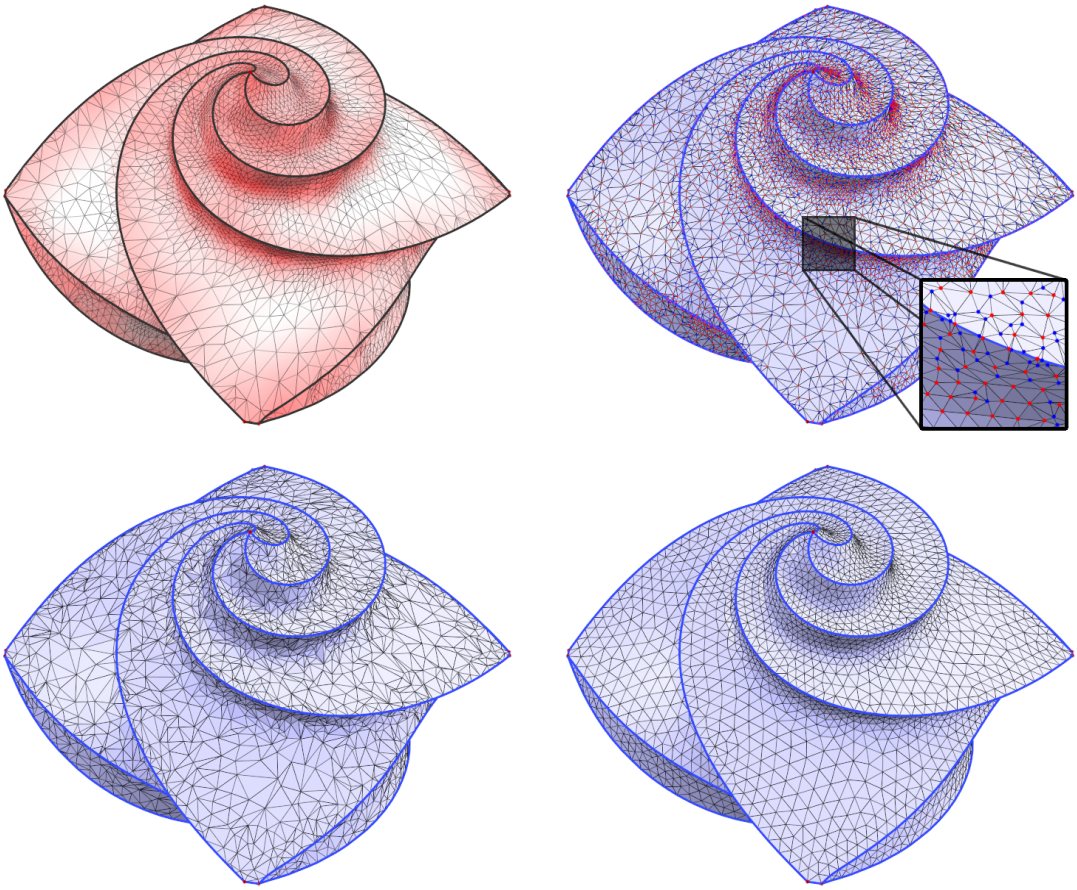

Figure 6.1: Stages of the remeshing algorithm: The final vertex distribution is controlled by a density function defined over the original mesh (top left). The mesh is resampled and meshed using a mutual tessellation (top right) that contains both, old vertices (red) and the new vertices (blue). Old vertices are deleted (bottom left) and Lloyd relaxation is applied to achieve a precise isotropic sample placement (bottom right), which results in well-shaped triangles.

6.1 Introduction

A very popular discrete surface representation is the triangle mesh which is simple to handle and can be rendered efficiently by modern graphics hardware. Modelling a surface in CAD applications as well as digitizing real-world objects typically results in a triangle mesh at some point. These different sources lead to meshes with strongly varying characteristics. For example CAD meshes often have few and skinny triangles to represent a certain surface as exactly as possible, whereas meshes from an acquisition pipeline usually contain many small triangles as a result of the scanning process. Depending on the target application such as rendering, numerical simulation, transmission, or compression, meshes often need to undergo complete remeshing prior to usage. Important desired properties are, e.g., the complexity of

the mesh in terms of number of vertices, the regularity of the connectivity, the quality of the triangle shape, and the sample distribution on the surface.

The goal of this work is to provide a simple but flexible remeshing framework that is capable of handling input meshes with diverse characteristics and produces remeshes with high-quality triangles shapes. The output of our remeshing algorithm is a new triangle mesh with either uniform or adaptive vertex distribution and almost equilateral triangles; thus, our method excels in the area of high-quality remeshing. The process is controlled by various parameters, for example the number of vertices in the remesh, the degree of adaptiveness to surface curvature, and whether remeshing should preserve surface features such as creases and sharp corners. We argue that an optimal remeshing approach should be

- *fast and efficient* to be able to handle large meshes,
- *simple but effective* for ease of implementation,
- *general and robust* to handle meshes with arbitrary genus and boundaries, and
- *accurate* so that the resulting mesh is as close as possible to the input mesh and the vertex sampling follows the prescribed density function.

As shown in the following section, none of the currently available remeshing approaches fulfill all of these criteria. We therefore introduce a novel remeshing algorithm that is designed to meets these criteria. In particular,

- it employs a direct resampling strategy and does not rely on global parametrization for meshing,
- it is applicable to large meshes since resampling provides a beneficial initial vertex distribution,
- it is fast because we use efficient algorithms and CPU parallelization in the relaxation framework, and
- it is accurate because it refers to the input mesh geometry during remeshing.

The algorithm can be roughly divided into three stages: Preprocessing, resampling, and precise vertex placement (see Figure 6.1). In the first stage, the mesh curvature is estimated and features are extracted. The second stage creates a new mesh with an exact vertex budget and proper sample distribution. The algorithm proceeds with the third stage that produces a precise isotropic placement of the samples by constructing a weighted centroidal Voronoi tessellation (WCVT).

This chapter is organized as follows: Section 6.2 gives an overview of related work. Section 6.3 discusses preliminaries and shortly explains the pre-processing step. Section 6.4 describes how the reference mesh is resampled and robustly meshed to build an initial mesh. The new vertices are then re-distributed to achieve a precise isotropic placement as explained in Section 6.5. We present results in Section 6.6 and conclude with a discussion and thoughts about future work in Section 6.7.

6.2 Related Work

Remeshing of surfaces is an evolving field with a long history. It is used in many applications, e.g., in mesh editing, animation, simulation, compression, or for the generation of progressive meshes with levels of detail. Creating high-quality meshes strives to achieve two goals: *isotropy*, i.e., almost equilateral triangles and an overall *good vertex distribution*, typically adapted to surface curvature. High-quality meshes can be created by constructing the centroidal Voronoi tessellation (CVT) on the surface of the mesh. The CVT is a Voronoi tessellation whose vertices, or *sites*, are centroids of their corresponding Voronoi cells [Du et al., 1999]. In a weighted CVT (WCVT) the sites coincide with the *center of mass* rather than the geometric centroid. Such a tessellation can be constructed by applying Lloyd's relaxation method [Lloyd, 1982] to the sites. Due to its slow convergence, it requires, however, a good initial placement of the samples.

Alliez et al. [2003] proposed an isotropic remeshing algorithm that first distributes a given number of vertices over the mesh surface. Meshing and relaxation, however, is performed in a global parameter domain. Such a global parametrization may not exist in general and involves many delicate problems, such as cutting closed and high-genus meshes, numerical instabilities, and the final lifting phase which brings the results back to the 3D domain.

To avoid the global parametrization of the input mesh, Surazhsky et al. [2003] proposed an isotropic remeshing algorithm based on a local parametrization approach [Surazhsky and Gotsman, 2003]. The method first brings the mesh to the required amount of vertices using edge-collapse and vertex-split operations. Lloyd relaxation is then applied in a framework that parametrizes only a small part of the reference mesh required to relocate a vertex to the centroid of its associated Voronoi cell. One drawback is that the mesh, once at the exact vertex budget, may not necessarily exhibit a sample distribution that complies with the density function. Since this makes Lloyd's iteration practically infeasible, Surazhsky et al. propose an area-equalization procedure with much faster convergence to approximate the required sample distribution. However, this approximation still uses local vertex relocations, may get stuck in local optima and does not give any guarantees on the produced sampling.

To alleviate the problems with these two approaches, Fu and Zhou [2009] propose to first apply a Poisson disc sampling on the input mesh and then apply the relaxation framework from Surazhsky et al. [2003]. While this approach achieves excellent local distribution after sampling, it is computationally extremely costly even for meshes with only a few thousand vertices. Yan et al. [2009] avoid explicit 2D parametrization. Instead, they repeatedly build the Restricted Voronoi Diagram for the mesh surface and apply a quasi-Newton optimization method which achieves faster convergence than Lloyd relaxation alone. The approach tolerates input meshes with degenerate triangles which is problematic for parametrization-based methods; they back-project samples to the surface but consequently need to obey sampling theorems to preserve homeomorphism. The proposed method is

strictly tailored towards building the CVT and does not support other relaxation schemes.

Similar to Fu and Zhou [2009], we aim at combining the advantages of Alliez et al. [2003] with the local parametrization framework as proposed by Surazhsky et al. [2003] to construct the CVT directly on the mesh. We argue, however, that the precise initial *local* sample distribution provided by the Poisson disc sampling is not necessary. Similar distributions can quickly be achieved using a few Lloyd iterations. Our algorithm is significantly faster and therefore applicable to much larger meshes than the approach by Fu and Zhou [2009].

6.3 Preliminaries

The input to the remeshing framework is an orientable 2-manifold triangle mesh $\mathcal{M}_o$ of arbitrary genus, any number of boundaries and possibly multiple connected components. Unless normals are provided with the input data, we estimate them using tessellation invariant angle-weighted pseudo-normals [Thuermer and Wuethrich, 1998]. We consider the input mesh to approach a C^1-continuous surface everywhere except at boundaries and feature-tagged edges. Feature edges receive special treatment during remeshing and will be preserved in the final mesh $\mathcal{M}_f$. The vertex distribution of $\mathcal{M}_f$ will be either uniform or comply with a density function defined over the reference mesh $\mathcal{M}_o$.

To maintain fidelity to the reference mesh during relaxation, the vertices of $\mathcal{M}_f$ are restricted to the surface of $\mathcal{M}_o$ at all times: A *barycentric coordinate* $b_T = (b_1, b_2, b_3), b_i \geq 0, \sum b_i = 1$ with respect to a triangle $T = \triangle_{v_1,v_2,v_3}$ uniquely defines a position $v = \sum b_i v_i$ on T. We use the notation $(b,\ T)_{\mathcal{M}}$ to refer to any position on mesh $\mathcal{M}$ and call that pair a *barycentric reference* onto $\mathcal{M}$.

6.3.1 Pre-Processing

To make the method applicable to technical data sets, e.g. models of mechanical parts which typically contain sharp edges, a set of features may either be specified by the user or procedurally extracted. We use a naive thresholding of the dihedral angle and manual tagging but more robust techniques can be applied [Jiao and Heath, 2002]. We also consider boundary edges (with only one adjacent face) as feature edges and handle this case equally. A feature skeleton is built by chaining all these edges together; such a skeleton may contain both open and closed backbones, where open backbones are terminated by corner vertices and edges of closed backbones form a loop.

A density function can be specified or estimated from the mesh geometry. We use simple formulas from Dyn et al. [2001] to approximate the Gaussian and absolute mean curvature at the mesh vertices, combine and clamp them to remove outliers and apply a contrast exponent γ to control the degree of adaptiveness. Gradation of the density function is influenced by iteratively applying a weighted Laplacian

smoothing operator on the density values. See Alliez et al. [2003] for more details on how density gradation influences the final mesh.

The mesh is then prepared for resampling by applying a simple and fast algorithm by Sander et al. [2007] which re-arranges the triangles of the mesh to improve vertex cache efficiency. We will show later why this is a useful property for our resampling procedure. Finally, the input mesh $\mathcal{M}_o$ is taken as reference mesh for the remeshing algorithm and remains unchanged for the rest of the pipeline.

6.4 Building the Initial Mesh

We aim at distributing a user-defined number of samples over the mesh surface such that the sample distribution complies with the prescribed density function. The samples are first partitioned between the surface and the feature skeleton. Both parts are then processed separately: The surface is sampled by drawing random barycentric coordinates for each triangle. The feature skeleton is rebuilt from scratch by accurately sampling the backbones according to the density function. The vertex positions of the new skeleton are exact and remain unchanged during remeshing. The new samples from both the smooth parts and the skeleton are finally meshed together using a mutual tessellation [Turk, 1992].

6.4.1 Sample Partitioning

We first integrate the density function over the surface and the feature edges and obtain two mass quantities D_s and D_f. In the uniform case, this corresponds to the area and length of the surface and the features, respectively. To partition the vertex budget S between the surface and the features, we apply formulas from Alliez et al. [2003] and obtain the number of samples S_s and S_f to distribute on the surface and the feature skeleton.

6.4.2 Triangle Sampling

To sample the surface, we traverse the triangles and deduce the number of samples S_T for each triangle T, $S_T = \frac{S_s}{D_s} \cdot D_T$, where D_T is the mass of T. We round S_T to the nearest integer. This creates a signed quantization error, which is propagated to the next unprocessed face in order.

The task of traversing all triangles for sampling raises the question of a suitable *processing path* over the mesh. One could use a random ordering of the triangles which produces reasonable results with minimal effort, but teleports the local quantization residual to arbitrary locations. Alliez et al. [2003] generalized the concept of error diffusion to obtain a connected processing path over the mesh triangles. In this method, the local quantization error is diffused to adjacent faces which keeps the error local. In contrast, we decided to take advantage of the fast reordering algorithm from Sander et al. [2007] to optimize the triangle ordering for spatial locality in the pre-processing step. This creates an adequate flow over the mesh triangles

and delegates the local quantization residual to the next face in processing order. In contrast to the error diffusion approach, we do not need to store the propagated error for multiple unprocessed triangles at the processing boundary.

For efficiency we distinguish between non-uniform and uniform triangle sampling, see Figure 6.2.

Non-uniform triangle sampling

Given the density values $d_1, d_2, d_3 > 0$ at the corresponding vertices of T, we interpolate the density $d(b) = d_1 b_1 + d_2 b_2 + d_3 b_3$. Without loss of generality, we assume that T lies in the xy-plane. We normalize the density $\tilde{g} := d / \int_T d$ and use $\tilde{g}$ as joint distribution of b_1, b_2. We compute the marginal density g_1 and the cumulative density function (CDF) F_1

$$g_1(b_1) = \int_T \tilde{g}(b_1, b_2) \mathrm{d}b_2 \qquad F_1(x_1) = \int_0^{x_1} g_1(b_1) \mathrm{d}b_1. \tag{6.1}$$

The CDF is inverted by solving a cubic equation. We then draw a sample from a uniform distribution on $[0, 1]$ and transform it into a sample b_1 with distribution g_1. With this sample, the conditional distribution of b_2 is given as $g_2(b_2 \mid b_1) = \frac{\tilde{g}(b_1, b_2)}{g_1(b_1)}$. We compute the CDF

$$F_2(x_2 \mid b_1) = \int_0^{\min(x_2, 1-b_1)} g_2(b_2) \mathrm{d}b_2 \tag{6.2}$$

and solve the resulting quadratic equation to invert it. Again, a uniformly distributed sample is transformed into a sample b_2 with distribution $g_2(\cdot \mid b_1)$, see [Hormann et al., 2004] for details.

Uniform triangle sampling

To generate uniform random barycentric coordinates we draw two uniformly distributed random numbers $\tilde{x}_1$, $\tilde{x}_2$ in $[0, 1]$. We reorder these values by assigning $x_1 = \min(\tilde{x}_1, \tilde{x}_2)$, $x_2 = \max(\tilde{x}_1, \tilde{x}_2)$, which leads to distributions with expected values $1/3$ and $2/3$ for x_1 and x_2, respectively. The lengths of the three intervals between $0, x_1, x_2, 1$ are then taken as barycentric coordinate $b = (x_1, x_2 - x_1, 1 - x_2)$. This yields samples with a uniform distribution over any triangle.

6.4.3 Skeleton Sampling

Instead of randomly sampling the feature skeleton and applying Lloyd relaxation in 1D to feature samples, we calculate the exact sample positions analytically. To emulate the WCVT for the 1D case, we place samples such that each sample is associated with the same amount of mass.

Once the total mass D_f (integrated density) of the feature skeleton and the number of samples S_f to distribute on the skeleton is known, we calculate the optimal

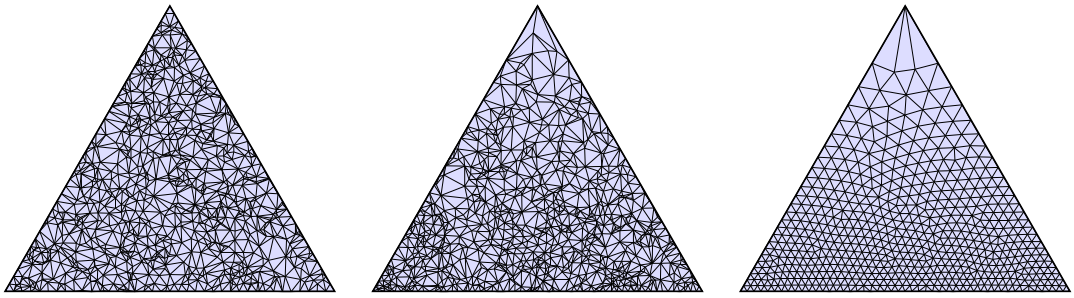

Figure 6.2: Triangle sampling: Uniform (left), adaptive with increasing density from top to bottom (middle) and adaptive sampling after Lloyd relaxation (right).

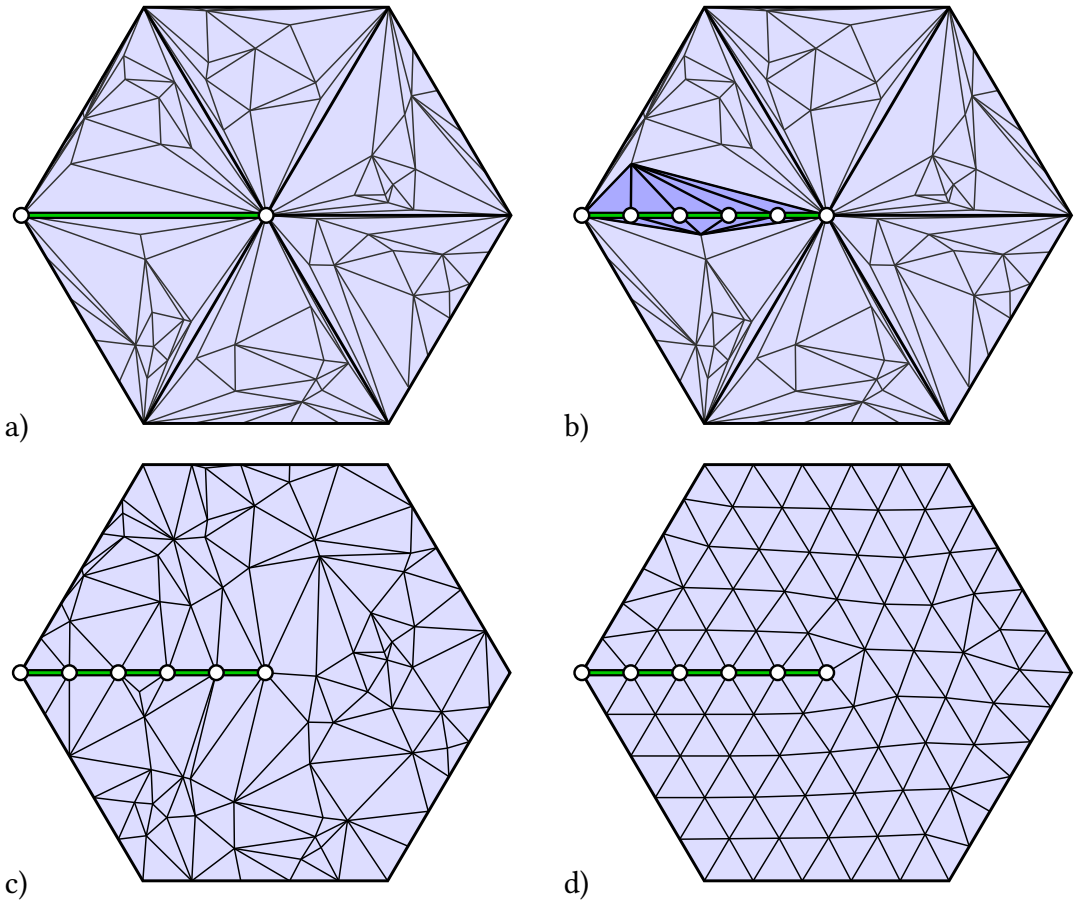

Figure 6.3: Mutual tessellation with features: a) A Delaunay triangulation is constructed locally in each triangle of the original mesh. b) Feature edges are then resampled while inserting new triangles. c) Degeneracies are eliminated by flipping edges afterwards. d) Features remain stable during relaxation.

sample spacing R_f^{-1} (or more formally the *inverse sampling rate*) for the whole skeleton, expressed in mass per sample. The optimal mass for the samples of the skeleton are calculated as follows, where B_o is the number of open backbones, S_f the number of samples to distribute and C the number of corner vertices in the skeleton:

$$R_f^{-1} = \frac{D_f}{B_o + S_f - C} \tag{6.3}$$

To proceed, we derive similar quantities for each backbone: To calculate the optimal number of samples S_B for a backbone we divide the backbone mass by the sample spacing R_f^{-1}, round to the nearest integer, and obtain a non-fractional number of samples to distribute on the backbone. This rounding creates a signed quantization error which is delegated to the next backbone in order. The optimal sample spacing per backbone R_{fb}^{-1} is then derived by dividing the backbone mass by the non-fractional sample amount.

We now aim at placing a sample every R_{fb}^{-1} mass on the backbone. We traverse the edges of the backbone in order, cumulate the mass and place a sample when R_{fb}^{-1} mass has been collected. The exact position of a sample within an edge of length l is then calculated by solving the following equation for $x_2 \in [0, l]$, where D_{left} is the remaining mass to collect from a known position x_1 to a yet unknown position x_2 (D_{left} is equal to R_{fb}^{-1} if the last sample was inserted on the same edge, otherwise D_{left} is the remainder from previous edges):

$$\int_{x_1}^{x_2} d(x)\, dx = D_{\text{left}} \quad \text{with} \quad d(x) = (1 - \frac{x}{l})d_1 + \frac{x}{l}d_2 \tag{6.4}$$

Note that $d(x)$ is the density function with density values d_1 and d_2 at the edge vertices, linearly interpolated over the edge with length l. The new sample is placed on the current edge at position x_2 and the process is repeated for the same edge but starting from x_2, possibly passing edge boundaries until all edges of the backbone are processed.

6.4.4 Meshing the Samples

The sampling process provides a set of surface and feature samples on the reference mesh surface $\mathcal{M}_o$. We must connect these samples to obtain a valid triangle mesh consisting of the new samples only. Several surface reconstruction algorithms can generate connectivity information for a set of points. These techniques, however, do not incorporate the underlying connectivity information of $\mathcal{M}_o$ and cannot guarantee a topologically equivalent surface. Peyré and Cohen [2006] exploit geodesic information to build a Voronoi diagram directly on the mesh and connect samples of neighboring Voronoi cells, but this is computationally expensive. We follow a more efficient approach and create a mutual tessellation [Turk, 1992] of both, the original and the new vertices. The original vertices are deleted afterwards.

We start constructing the mutual tessellation by inserting samples into the triangles of the original mesh. An incremental Delaunay triangulation [Guibas et al.,

1992] is employed using the mesh triangle as *dummy triangle at infinity*. New samples are generated by drawing random barycentric coordinates, as described in Section 6.4.2, until the desired number of samples has been inserted. Each new sample is equipped with a barycentric reference $(b, T)_{\mathcal{M}_o}$ that tracks the exact position on the reference surface. To improve numerical stability, we bound new samples away from already existing samples and edges using ε-checks on the barycentric coordinates. If a sample violates the ε-condition, a new sample is drawn. Note that these checks are independent of the triangle size and thus do not affect the sampling density in regions with smaller triangles. This procedure creates a Delaunay triangulation within each triangle, preserves the edges of the original mesh, but introduces degenerate triangles around these edges, see Figure 6.3.

After merging the surface samples with the mesh, the feature samples are inserted by splitting the skeleton edges of the original mesh at the positions calculated in Section 6.4.3. This is possible because no original edges have been modified in the previous step. Splitting edges may create additional degenerate triangles but is numerically stable: All operations are performed on the vertex positions only and not on possibly degenerate quantities such as triangle areas. Finally, after both surface and feature samples have been inserted into the mesh, a global constrained Delaunay triangulation is restored by flipping edges of the mesh if the angles opposite to the edge sum to more than $180°$. This eliminates all degenerate triangles, see Figure 6.3.

The mutual tessellation is then cleaned by deleting the original vertices. When deleting a vertex, all adjacent triangles are also removed from the mesh, which leaves a hole in the triangulation. This hole is re-triangulated directly in 3D using a procedure similar to Schroeder et al. [1992], and we merely check that newly introduced edges are not already present in the mesh. This guarantees that the mesh topology is preserved. It happens that some original vertices cannot be deleted if re-triangulation fails, thus the exact vertex budget is slightly off. Although this is typically not a big problem in practice, we address this issue by randomly deleting newly introduced vertices (but no feature vertices) until the vertex budget is reached.

Overall, the resampling procedure results in a new mesh with a sampling density that globally complies with the prescribed density function. The new vertices are equipped with vertex references that point to positions on $\mathcal{M}_o$. This information will be used as initialization for vertex relaxation to optimize the mesh.

6.5 Improving Vertex Positions

To improve the sampling of the new mesh, we construct a weighted centroidal Voronoi tessellation (WCVT). One way to do this is Lloyd's algorithm [Lloyd, 1982]. The Lloyd relaxation is a simple iterative method that consists of three basic steps:

- Build the Voronoi diagram of the samples,
- move each sample to the centroid of its Voronoi cell,
- iterate the procedure until convergence.

Lloyd's algorithm is a particularly slow process with bad convergence behaviour, i.e., the main improvement is achieved in the first few Lloyd iterations and performance of convergence quickly decelerates. However, the initial vertex distribution obtained from the sampling process is *globally* correct and *locally* already a good approximation of the density function. This allows for effective, progressive improvement of the mesh with fast convergence.

Building the WCVT

To construct the WCVT, we closely follow the approach of Surazhsky et al. [2003]: Instead of constructing the Voronoi diagram for the whole mesh at once, a Voronoi cell is created locally for each vertex, which is easily derived from the local Delaunay property. A single vertex v and the adjacent triangles are flattened and the Voronoi cell for v is constructed. A density value is assigned to v and to each vertex of the Voronoi cell polygon. The Voronoi cell is triangulated by connecting the polygon vertices with v, and the cell's centroid is calculated by summing the centroids of the individual triangles, weighted with the triangle mass. This yields a new position v^* in the planar domain and v is relocated to that position.

Vertex Relocation

The task of relocating vertex v to v^* boils down to calculating a new barycentric reference $(b^*, T^*)_{\mathcal{M}_o}$ for v^*. To perform the relocation, a patch of $\mathcal{M}_o$ is created that contains a small region required to relocate the vertex, yet large enough to be reused for spatially close relocations. The method in [Surazhsky and Gotsman, 2003] first gathers triangles of the reference mesh in a breadth-first search to create a roundish patch in 3D. The patch is then flattened using a conformal parametrization with fixed circular boundary using Floater's Mean Value Coordinates [Floater, 2003].

Optimizations

Surazhsky also proposed a caching scheme with fast patch lookup that keeps patches for a small amount of time, which drastically improves the performance of the approach. To make effective use of the caching system, we reorder the vertices of the mesh prior to relaxation. We extend the efficient triangle reordering algorithm from Sander et al. [2007] in the following way: Once triangles have been reordered, we traverse the triangles and issue the vertices in order of their first appearance. This creates a processing path over the mesh vertices with spatially low variance. The path allows subsequent relocations to reuse patches that have recently been created and reduces the average cache miss ratio for large meshes that require cache cleanups.

Another observation is that Lloyd relaxation is easily parallelizable on the CPU by partitioning the vertices into equally large sets for processing in separate threads. Due to thread locking mechanisms, performance is not linear with the amount of threads and we achieved best results with 8 threads. We do not delete patches after

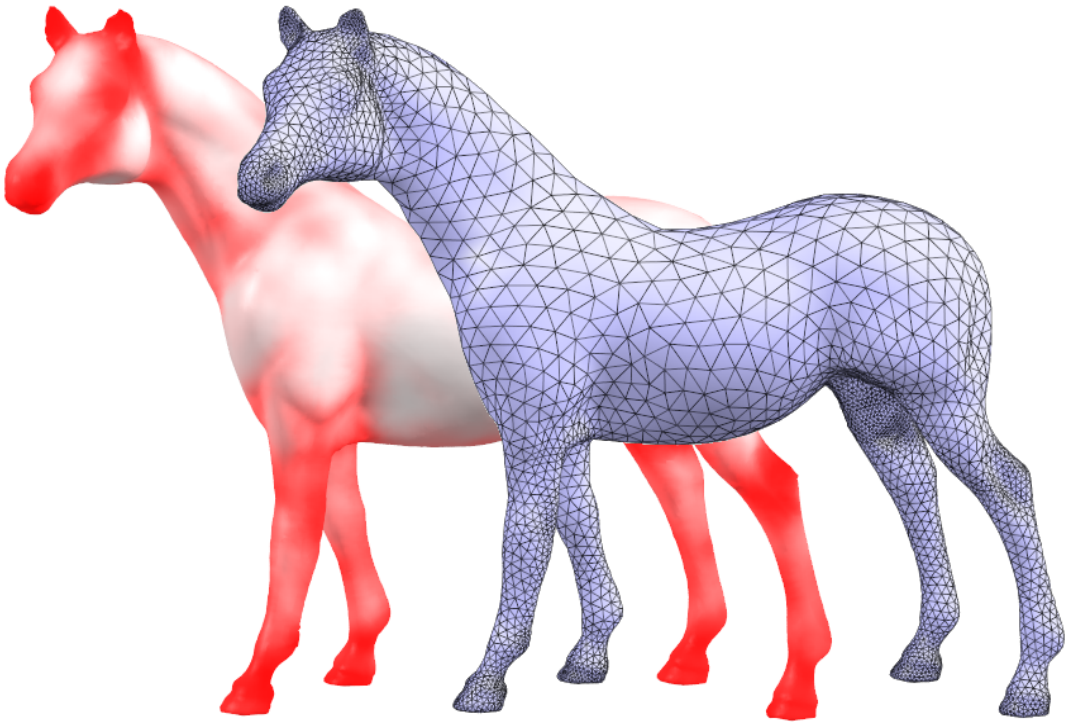

Figure 6.4: The horse model with about 50k vertices and the remeshed model with 6k vertices.

time but threshold memory consumption to trigger cleanups, which is a more thread friendly approach. For each cleanup, cached patches are sorted by access time and a fixed portion of patches (20% in our experiments) with earliest access time is deleted from the cache. Experiments show that a cache of 256 MB is sufficient to handle meshes with about a million vertices without deleting any patch from the cache.

6.6 Results

Figure 6.4 shows an adaptive remesh of the horse model and a visualization of the mesh curvature used as density function for remeshing. The original model has about 50k vertices and was downsampled to 6k vertices. The algorithm took about 2 seconds for resampling and 5 seconds for 100 Lloyd iterations with 8 threads. Figure 6.5 shows an example where the Hygieia model was upsampled from 8k to 10k vertices. Figure 6.6 shows several more models where remeshing to 5k vertices was performed with almost interactive speed.

We also demonstrate our technique on mechanical models. Figure 6.7 shows a remeshing result of the Fandisk mesh that has been resampled to 4k vertices and improved with 100 Lloyd iterations. The final sampling of the skeleton, which only contains open backbones, is a direct result of the resampling procedure and accurately matches with the triangles around the feature creases. Remeshing took 500ms for resampling and 3 seconds for Lloyd relaxation.

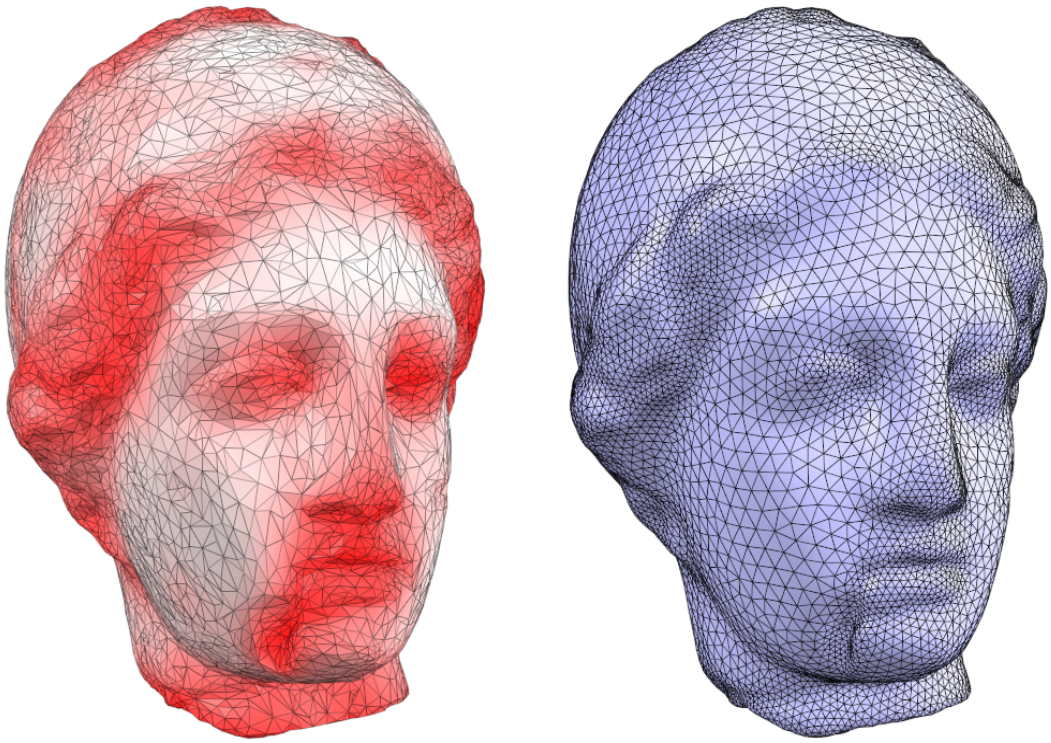

Figure 6.5: The Hygieia model with about 8k vertices and visualized density function (left) has been upsampled to 10k vertices (right).

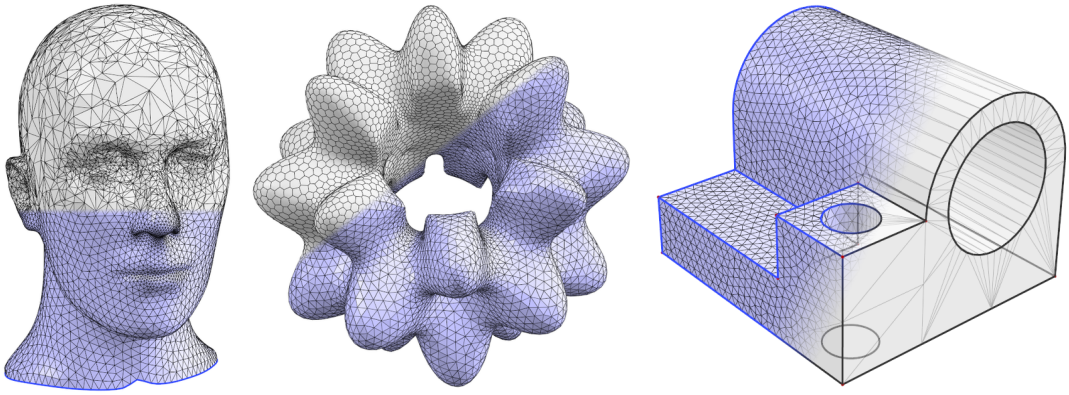

Figure 6.6: More remeshing results: The Mannequin model (left) with resampled subdivision surface (*top*) and the remeshed model (*bottom*), the remeshed Bumpy Torus model (middle) with Voronoi regions (*top*) and triangulation (*bottom*), and the Joint model (right). All meshes are remeshed with 5k vertices at almost interactive speed.

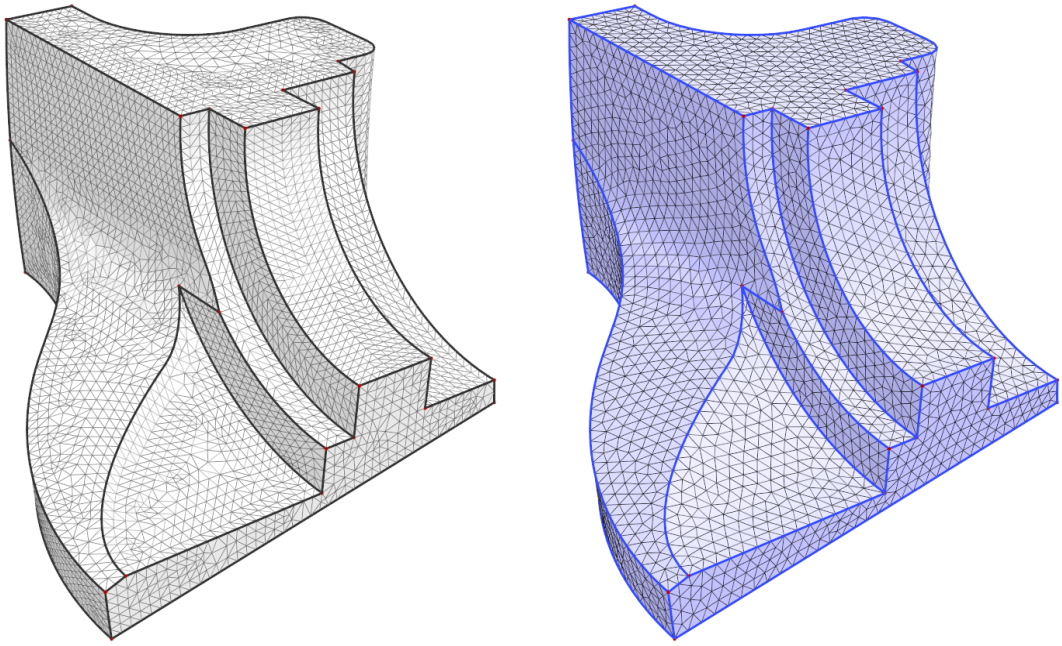

Figure 6.7: The Fandisk model with highlighted feature skeleton (left), remeshed with 4k vertices (right).

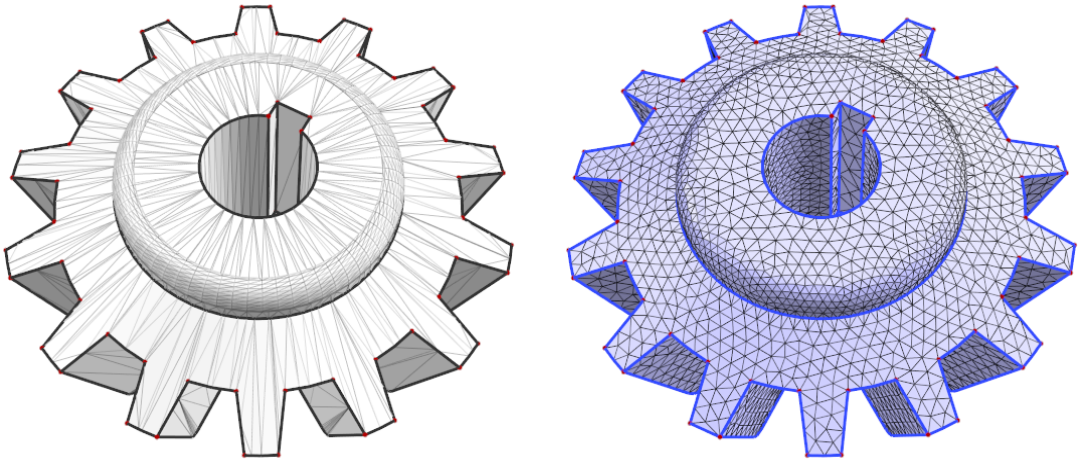

Figure 6.8: A typical CAD model with many degenerated triangles and feature skeleton (left) and the remeshed model with 5k vertices (right).

A model with typical CAD tessellation is presented in Figure 6.8. The degenerate triangles are a major issue for parametrization-based methods such as [Surazhsky and Gotsman, 2003], which is mainly caused by distortion in the patches and may lead to erroneous vertex relocations. We applied a simple mesh slicing procedure [Botsch and Kobbelt, 2001] with a regular grid to aid patch construction with more well-shaped triangles.

Table 6.1 presents a statistical analysis and comparison of the remeshing results.

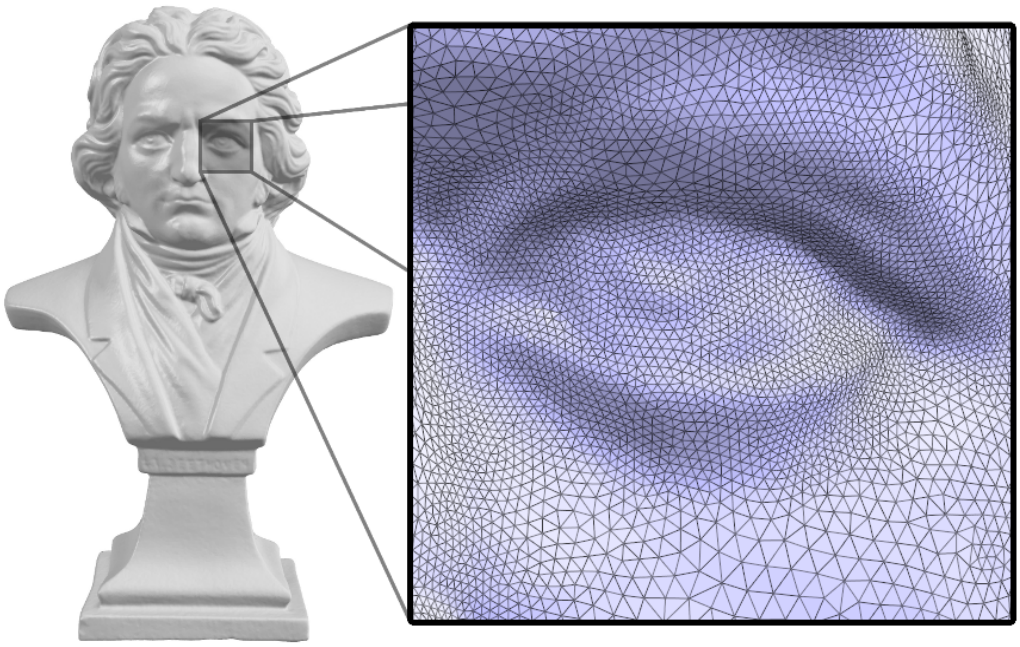

Figure 6.9: The Beethoven model remeshed from 1.5M vertices to 500k vertices. Resampling and Lloyd relaxation took less than 5 minutes with 8 threads.

The Avg$\angle$ and Min$\angle$ are the average of the minimum angle in each triangle and the smallest angle in the triangulation, respectively. The error is the Hausdorff distance w.r.t. the bounding box diagonal as calculated by Metro [Cignoni et al., 1998]. All comparisons are taken from the results of related work and have not been re-evaluated for normalization. Note that we achieve a low error while keeping an exact vertex budget.

For all timings we used an AMD Opteron multicore system with 2.7GHz per CPU. We limited the memory consumption for the patch cache to 1GB but this limit was never reached, not even for the Beethoven model (Figure 6.9), which used 630MB RAM for 84k patches.

6.7 Conclusion and Future Work

We presented an efficient remeshing framework for relaxation-based methods. In particular, we used Lloyd's algorithm to build a WCVT on the mesh to obtain an isotropic sample distribution. The key to performance is the initial sampling procedure which is itself efficient and simple, and provides the means for fast convergence of Lloyd's relaxation method. The mesh topology is preserved because both, the edge flip algorithm and hole re-triangulation do not introduce edges that are already present in the mesh. In theory, nothing prevents the algorithm from creating surface self-intersections, but we never observed this in practice. Our results compare favorably to state-of-the-art techniques in both processing time and mesh quality and fulfill the desired criteria listed in Section 6.1.

Model	Vertices	Time (sec)	∠ (deg) Avg	∠ (deg) Min	Error (10^{-3})
Hygieia (original)	8 268	–	34.7	0.25	–
Hygieia (our)	10 000	9 / 3.5	52.9	30.1	3.5
Hygieia [Surazhsky and Gotsman, 2003]	8 750	17	52.4	25.9	2.7
Hygieia [Fu and Zhou, 2009]	6 529	113	51.9	35.4	n/a
Horse (original)	48 485	–	37.1	1.27	–
Horse (our)	6 000	16 / 7	51.9	29.8	4.9
Horse [Surazhsky and Gotsman, 2003]	5 695	28	50.1	9.1	6.1
Horse [Fu and Zhou, 2009]	3 017	103	51.9	35.7	n/a
Fandisk (original)	6 475	–	43.5	17.0	–
Fandisk (our)	4 000	3 / 1.4	53.3	20.6	1.7
Fandisk [Surazhsky and Gotsman, 2003]	5 135	17	49.1	16.8	0.4
Beethoven (original)	1.5M	–	34.2	0.01	–
Beethoven (our)	500k	676 / 280	52.7	28.0	1.4

Table 6.1: Analysis and comparison of the remeshing results. We always applied 100 Lloyd iterations. The timings are for the full pipeline with 1 and 8 threads, respectively. Note that total times are not normalized across publications. [Surazhsky and Gotsman, 2003] used P4 with 2.4GHz; [Fu and Zhou, 2009] used P4 with 2.8GHz, we used AMD Opteron with 2.7GHz.

In the future, we would like to investigate different data structures for patch caching that may be more suitable for parallelization and will probably reduce the drop in performance for a larger number of threads. We would also like to improve the parametrization strategy to handle complicated regions of the mesh in a consistent way. For example non-manifold connectivity is problematic and the system fails to create patches in these regions. Meshes with highly degenerate triangles prevent the parametrization strategy from creating roundish patches, which causes large distortions and harms the accuracy of vertex relocations.

Chapter 7

Surface Reconstruction Evaluation

Abstract

In this chapter we compare and discuss the results of the surface reconstruction approaches presented in Chapter 4, *DMFusion* [Fuhrmann and Goesele, 2011] and Chapter 5, *FSSR* [Fuhrmann and Goesele, 2014]. We are interested in the performance on several types of datasets.

- Scanner data: We investigate how both algorithms perform with relatively clean scans obtained with 3D scanners.
- Multi-View Stereo data: The impact of noise and outliers in MVS data on the reconstruction algorithms will be examined.
- Multi-scale MVS data: A visual comparison on how these algorithms perform in the presence of multi-scale data will be analyzed.

Although both approaches have strengths and weaknesses, *FSSR* turns out to be superior in quality while consuming fewer resources. In particular, the more efficient surface extraction and the capability to close small holes are useful in both controlled scanning and difficult MVS scenarios. We also present remeshing results on some reconstructions.

Contents

7.1 Scanner Data

The data obtained from 3D scanners is usually much cleaner in terms of noise and outliers than the geometry from stereo-vision algorithms. On the other hand, scanner data is less redundant and thus mis-alignments of the individual scans can have an adverse impact on the geometry. In this section we perform qualitative comparisons on a range of datasets from small, compact objects to larger, detailed scans. Table 7.1 in Section 7.4 shows quantitative reconstruction statistics for all datasets.

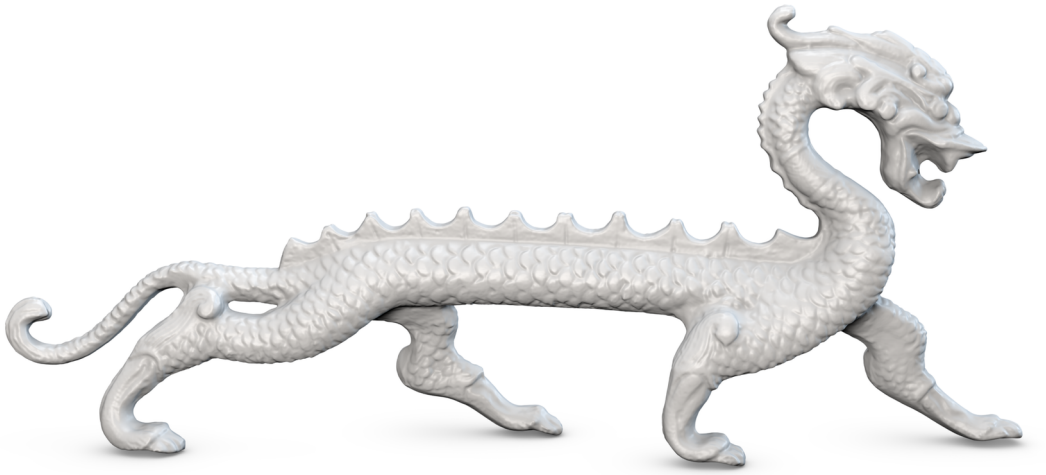

Figure 7.1: The FSSR reconstruction of the *Polymetric Dragon* dataset.

Dragon Dataset

This dataset of a small dragon object is courtesy of *Polymetric GmbH*. It consists of 32 range images with a total of 3 091 786 samples. Overall, the meshes reconstructed by the two approaches look visually very similar. The reconstruction using FSSR is shown in Figure 7.1. On close inspection, the surface computed with *DMFusion* shows some small artifacts (see Figure 7.2). We believe that this is due to fusion of information at multiple scales, caused by the discretization of the scale space into octaves. These artifacts disappear when performing the reconstruction on a uniform scale. This makes the approach conceptually equivalent to the *VRIP* method [Curless and Levoy, 1996]. The FSSR approach performs well on this dataset, producing a smooth yet detailed surface.

The running time and peak memory usage for *FSSR*, *DMFusion* and *DMFusion* on a single scale is listed in Table 7.1. It can be seen that the multi-scale *DMFusion* reconstruction is by far the most expensive approach. The data structure used by the global Delaunay tetrahedralization for isosurface extraction is the main bottleneck with a peak memory usage of 4601 MB and 216 seconds running time alone. The single scale reconstruction is much more efficient because the surface can be extracted using the Marching Cubes algorithm [Lorensen and Cline, 1987].

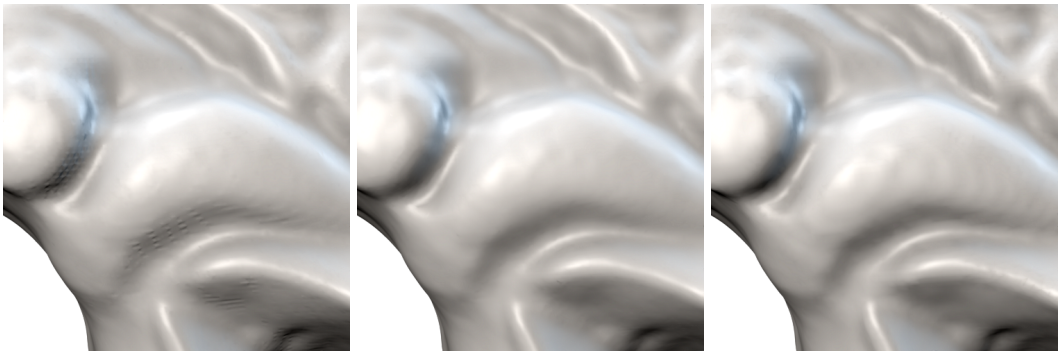

Figure 7.2: *Left:* The reconstruction of the *Polymetric Dragon* with *DMFusion* shows a few small artifacts. *Middle:* Performing the *DMFusion* reconstruction on a single scale removes these artifacts. *Right:* The reconstruction using *FSSR*. Note: All images are contrast enhanced.

CNR Laurana Dataset

The *Laurana* dataset has been reconstructed from 27 range images yielding 3 034 883 surface samples. The *DMFusion* result is slightly smoother mainly because of the chosen reconstruction resolution. *FSSR* produces a cleaner result with slightly more details, especially in the region around the ear, see Figure 7.3. Both approaches leave a small hole inside the ear due to insufficient data coverage and the lack of hole filling capabilities.

Polymetric Pumpkin Dataset

The *Pumpkin* dataset consists of 18 colored, high-resolution scans with a total of 10 955 857 samples. See Figure 7.4 for the reconstruction result. Similar to the other datasets, reconstruction with *DMFusion* at a single level yields good results. However, in regions with sparse sampling the reconstruction shows a few holes. In contrast, the *FSSR* reconstruction yields a watertight result. Due to the clean and round structures of the Pumpkin, some very subtle ringing artifacts become visible at certain camera angles. This issue of the *FSSR* approach is further discussed in the conclusion of this chapter.

Figure 7.3: The *Laurana* dataset reconstructed with *FSSR*, courtesy of *CNR ISTI*. *Bottom left:* Reconstruction with *DMFusion*. *Bottom right:* Reconstruction with *FSSR*, which is slightly less smooth but more detailed.

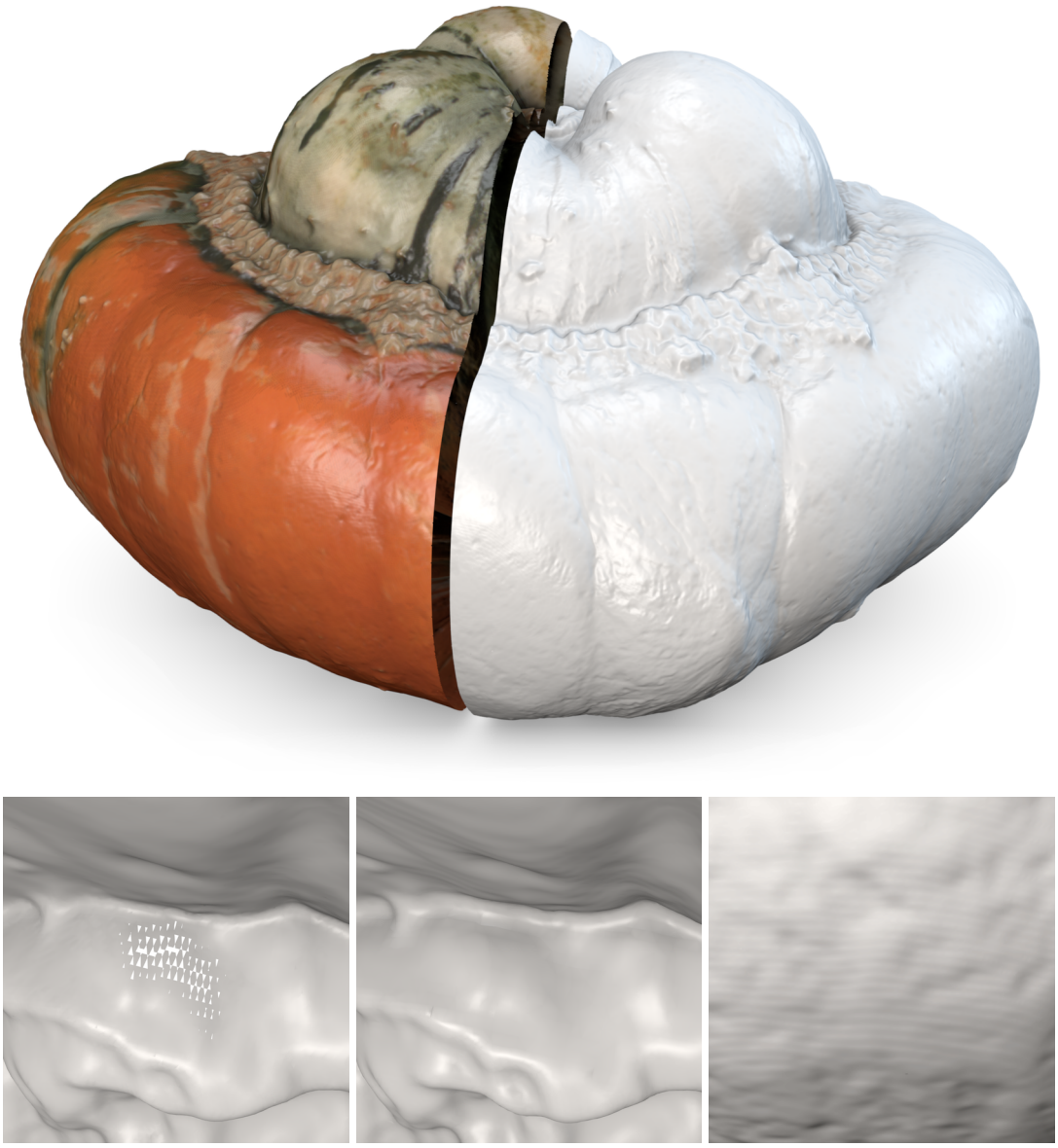

Figure 7.4: The *Pumpkin* dataset courtesy of *Polymetric GmbH*, reconstructed with *FSSR* [Fuhrmann and Goesele, 2014], rendered with color (left) and shading (right). The bottom row shows occasional holes with *DMFusion*, while *FSSR* produces a watertight result (middle). The ringing artifacts produced by *FSSR* (right) are rare and barely visible (the image is contrast enhanced).

7.2 MVS Data

We are interested in the performance of the reconstruction algorithms on Multi-View Stereo data. Controlled datasets with a dense camera coverage are the *Middlebury Dino* and *Middlebury Temple* dataset. We will further present *Die Badende* with less controlled cameras but only moderate variations in scale. Note that, as with all Multi-View Stereo datasets, the result of the surface reconstruction algorithm heavily depends on the output geometry of the specific MVS algorithm.

Figure 7.5: Reconstruction of the *Middlebury Temple* with *DMFusion* (left) and *FSSR* (right). Although the *DMFusion* reconstruction produces slightly sharper edges in some regions, the *FSSR* reconstruction is very clean without spurious geometry.

Middlebury Temple

The *Temple* dataset contains 312 images at a resolution of 640×480. Individual depth maps are reconstructed using the MVS approach by Goesele et al. [2007]. The 312 depth maps (22 849 326 samples) are then combined using both techniques, *DMFusion* and *FSSR*. A rendering of the resulting surfaces is available in Figure 7.5. A quantitative evaluation of the *FSSR* result is available on the Middlebury benchmark[1] under the name *Fuhrmann-SG14* (ranked 3rd with an accuracy of 0.39mm, and ranked 9th with a completeness of 99.4%, retrieved March 2015).

[1]Middlebury Multi-View Stereo evaluation: http://vision.middlebury.edu/mview/eval/

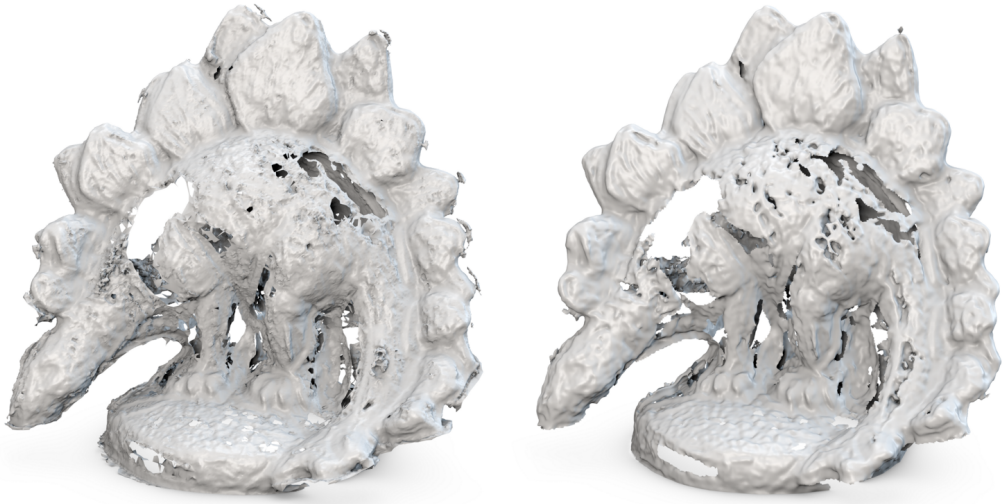

Figure 7.6: Reconstruction of the *Middlebury Dino* with *DMFusion* (left) and *FSSR* (right). The *FSSR* reconstruction is smoother, more detailed and produces less clutter. Both results are incomplete and noisy due to the MVS algorithm.

Middlebury Dino

Similar to the *Temple*, the *Dino* dataset contains 312 images at a resolution of 640×480 and yields a total of 29 872 435 samples. Although captured with a very controlled setup, the absence of texture on the object makes this dataset very challenging and causes extremely noisy depth maps. We argue that the MVS algorithm at hand by Goesele et al. [2007] is not suited for this dataset due to the lack of explicit regularization in weakly textured regions. The differences between the two approaches become more obvious on this challenging dataset, see Figure 7.6 for a visual comparison. Here, *DMFusion* is used in multi-scale mode. A cleaner reconstruction can be achieved by choosing a fixed resolution for *DMFusion*, but was avoided to make the comparison fair.

Die Badende

This sculpture called *Badende* is a replica of Bernhard Hoetger's original work from 1911. The 343 input photos in this dataset have been taken at various positions around the statue with different distances to the geometry, which results in moderate variations in scale. The full reconstruction starting from images to final surface has been performed with the *MVE* software presented in Chapter 3. A total of 52 970 450 samples have been obtained for surface reconstruction. Both surfaces show some holes in occluded regions, such as under the arm and chin, and a few erroneous surface parts are reconstructed due to MVS mismatches. Renderings of the resulting meshes are presented in Figure 7.7, where *FSSR* produces more details and better reconstructions especially in concavities.

Figure 7.7: Reconstruction comparison of *Die Badende*, an uncontrolled MVS dataset with moderate scale differences. The bottom row shows 5 of the 343 input photos. The *FSSR* reconstruction (right) shows more detail and more complete geometry compared to the *DMFusion* reconstruction (left).

7.3 Multi-Scale MVS Data

In this section we present a direct comparison of the two approaches on multi-scale datasets. These datasets, namely the *Citywall* and the *Goethe Fountain*, are known from Chapter 4 and 5.

Figure 7.8: The colored *Citywall* dataset. Most of the brick work is represented at lower resolution while the fountain and the miniature city (not visible here) are reconstructed at much higher resolution.

Citywall

The dataset contains a total of 564 photos from an historic wall in Darmstadt, Germany (see Figure 7.8). Most of the wall is brick work, but a few interesting details are part of the scene: A fountain with two small, detailed lion heads and a miniature city model with challenging geometry. Both algorithms perform well on this dataset. However, the *FSSR* reconstruction produces crisp and clean geometry with more details while better suppressing the noise in the individual depth maps. The *DMFusion* approach, on the other hand, shows small artifacts and less clean geometry. Figure 7.9 shows a comparison on a few details.

Goethe Fountain

The *Goethe Fountain* is located in Hochstädten near Bensheim, Germany. The whole site is excavated a few meters in the ground and consists mostly of brick work, but contains two interesting fountains. Because the fountains look similar to each other, we concentrated on one of the fountains by taking many detailed photos. The resulting reconstructions feature both coarse and very detailed geometry from the multi-scale input.

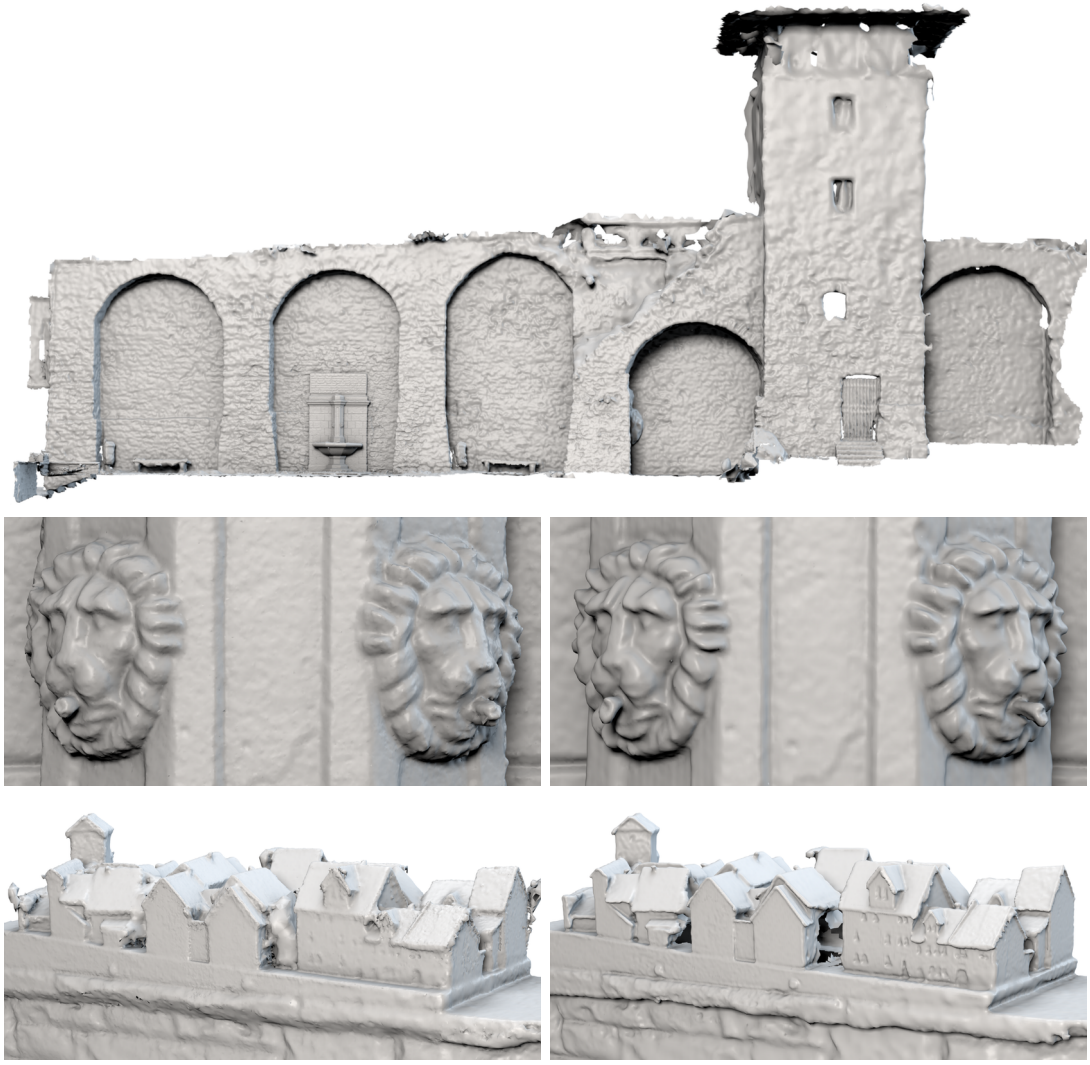

Figure 7.9: *Top row:* The full *Citywall* dataset without color reconstructed with *FSSR*. The fountain region is covered with more close-up input photos and a higher resolution mesh is extracted in this region. *Bottom rows:* Comparison of details reconstructed with *DMFusion* (left) and *FSSR* (right).

The two approaches perform similar and both produce a good reconstruction, see Figure 7.10. We believe that the large amount of redundancy is mainly responsible for the clean geometry. However, the *DMFusion* approach delivers a result with many small holes, and the Marching Tetrahedra algorithm creates many more unnecessary triangles. *FSSR* produces a smoother geometry with less triangles and less geometric artifacts while preserving more details.

Figure 7.10: Comparison of the *Goethe Fountain* dataset. *Top:* The full reconstruction using *FSSR*. *Middle Row:* Details reconstructed with *DMFusion*, *Bottom row:* Details reconstructed with *FSSR*.

7.4 Reconstruction Statistics

The following table lists the reconstruction statistics for all datasets presented in this chapter. The individual timings for each algorithm are split into computation of the implicit function and surface extraction. The timings have been obtained on a Intel Xeon Quad-Core computer with 8×2.6 GHz per CPU and exclude the time required for file I/O.

Dataset	Algorithm	Memory	Runtime	Vertices
Polymetric Dragon	FSSR	611 MB	116 + 7 sec	941 143
3 091 786 samples	DMF-MS	4 601 MB	71 + 239 sec	1 541 663
32 range images	DMF-SS-11	569 MB	47 + 16 sec	768 738
Polymetric Pumpkin	FSSR	4 062 MB	1161 + 59 sec	5 263 482
10 955 857 samples	DMF-MS	25 251 MB	321 + 1420 sec	8 578 361
18 range images	DMF-SS-11	5 950 MB	365 + 189 sec	7 886 029
CNR Laurana	FSSR	1 471 MB	375 + 20 sec	1 935 528
3 034 883 samples	DMF-MS	10 478 MB	91 + 599 sec	3 670 876
27 range images	DMF-SS-11	4 052 MB	175 + 137 sec	5 541 438
Middlebury Temple	FSSR	2 085 MB	366 + 4 sec	266 465
22 849 326 samples	DMF-MS	2 863 MB	548 + 162 sec	1 080 837
312 depth maps	DMF-SS-10	1 748 MB	566 + 13 sec	824 659
Middlebury Dino	FSSR	2 579 MB	726 + 10 sec	384 985
29 872 435 samples	DMF-MS	6 073 MB	932 + 362 sec	2 410 996
312 depth maps	DMF-SS-10	5 137 MB	1255 + 42 sec	2 817 702
Die Badende	FSSR	5 203 MB	1678 + 34 sec	2 683 086
52 970 450 samples	DMF-MS*	6 760 MB	544 + 255 sec	2 676 170
343 depth maps	DMF-SS-11	4 801 MB	431 + 45 sec	3 542 891
Citywall	FSSR	7 638 MB	2737 + 35 sec	2 624 613
77 086 356 samples	DMF-MS*	7 823 MB	698 + 328 sec	2 500 367
564 depth maps				
Goethe Fountain	FSSR	16 559 MB	5863 + 114 sec	9 605 213
161 842 791 samples	DMF-MS*	32 274 MB	1858 + 1284 sec	8 267 119
249 depth maps				

Table 7.1: Statistics for all reconstructions presented in this chapter. FSSR refers to the *Floating Scale Surface Reconstruction* presented in Chapter 5, DMF refers to *Fusion of Depth Maps with Multiple Scales* presented in Chapter 4 using either multiple scales (MS) or a single scale (SS) on a fixed octree level. For the DMF reconstructions marked with a star (*), the depth maps have been sampled at half the normal sampling rate, resulting in 1/8th of the octree voxels, to keep the memory and runtime in reasonable bounds. This should be considered when comparing with FSSR.

7.5 Remeshing Results

In this section we show results of the remeshing technique presented in Chapter 6. In particular, we demonstrate how the amount of triangles can be significantly reduced from the raw reconstruction output by either employing a new uniform or adaptive sample distribution. While the adaptive sampling is suited for meshes with both, uniform and highly varying curvature, the uniform vertex distribution is mainly applicable to compact, simple meshes. Remeshing of large multi-scale meshes requires a suitable definition of the curvature field and a precise relationship between surface curvature, triangle size and approximation error. Such a definition is out of scope of this thesis, and large multi-scale datasets are not considered for remeshing here. Figure 7.11 shows a few input images of multi-view stereo datasets that we use for remeshing in this section.

Figure 7.11: From top to bottom: The *Scared Owl* dataset, the *Snow Owl* dataset, the *Squirrel* dataset, and the *Middlebury Temple* dataset.

Scared Owl Dataset

The *Scared Owl* mesh in Figure 7.12 is an image-based reconstruction from 221 input images. Although this object is compact and quite small, the large number of input images leads to a very clean and noise-free reconstruction result. The *FSSR* reconstruction resulted in 284 211 vertices and has been remeshed to 10 000 vertices. Here, the object details are uniformly distributed over the surface, and a uniform vertex distribution is suitable. Because the final mesh contains less than 5% of the original number of vertices, a certain loss of detail is inevitable at this level of reduction.

The running time of the remeshing procedure can be considered insignificant compared to the time required for surface reconstruction. For the *Scared Owl* dataset, the resampling of the original mesh took about 5 seconds, and performing 100 iterations of Lloyd relaxation to obtain a precise isotropic vertex placement took about 10 seconds. The runtime for other models in this section are very comparable with the exception of the slightly bigger *CNR Laurana* dataset, for which we report runtime individually.

Snow Owl Dataset

Like the *Scary Owl*, the *Snow Owl* in Figure 7.13 is an image-based reconstruction from 61 input images. Surface reconstruction with *FSSR* resulted in 280 727 vertices. This mesh has many details and requires a denser sampling. In contrast to the *Scary Owl*, some regions contain less detail, e.g., around the bird's eyes. Here, an curvature-adapted vertex distribution with 25 000 vertices has been employed, which creates larger triangles in regions with less curvature.

The precise isotropic vertex distribution is achieved using Lloyd's relaxation procedure by iteratively moving the vertices to the centroids of their Voronoi cells. To obtain an adaptive vertex distribution with respect to a density function defined over the mesh, vertices are instead moved to the centers of mass of their Voronoi cells.

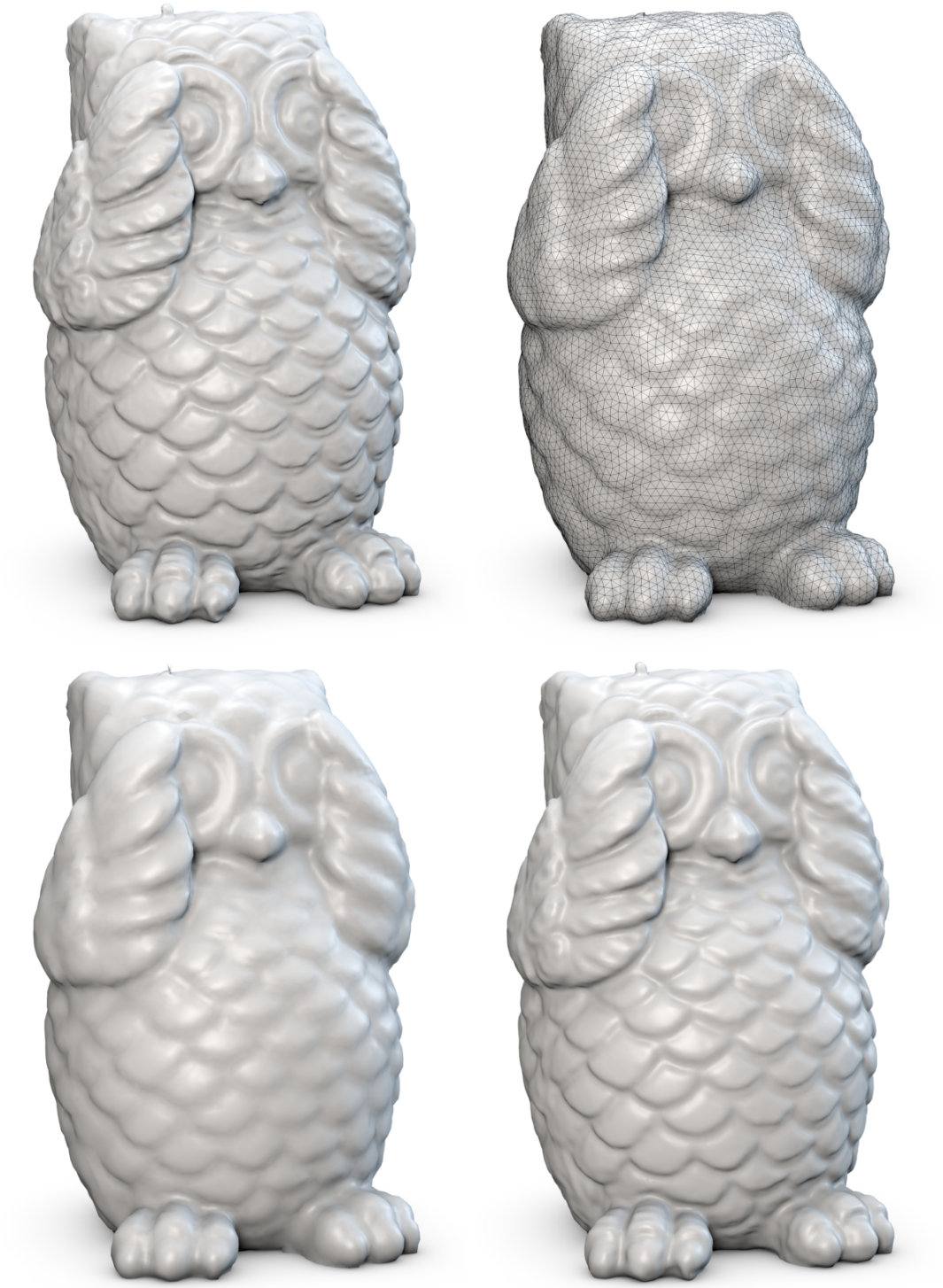

Figure 7.12: The *Scared Owl* reconstruction with 284 211 vertices (top left, triangles not shown) is remeshed to 10 000 vertices with a uniform vertex distribution (top right). The drastic vertex reduction causes a loss of shading details (bottom left), while a remesh with 50 000 vertices shows much more details (bottom right).

Figure 7.13: The *Snow Owl* reconstruction with 280 727 vertices (left, density field shown) is remeshed to 25 000 vertices with an adaptive vertex distribution. Regions with low curvature receive fewer triangles, such as the face region. Given a predetermined vertex budget, this leaves more triangles for regions of higher curvature to increase fidelity to the original surface.

CNR Laurana

The *CNR Laurana* reconstruction using *FSSR* with 1 935 208 vertices has been remeshed to 50 000 vertices, which is less than 2.5% of the original number of vertices. Here the vertex distribution has been chosen adaptively to better capture the details, while maintaining smooth transitions between triangle sizes. Figure 7.14 shows the remeshed model and a closeup view. (The original model is shown in Figure 7.3.) The remeshing operation required about 40 seconds for resampling of the original mesh, and 45 seconds for 50 Lloyd iterations.

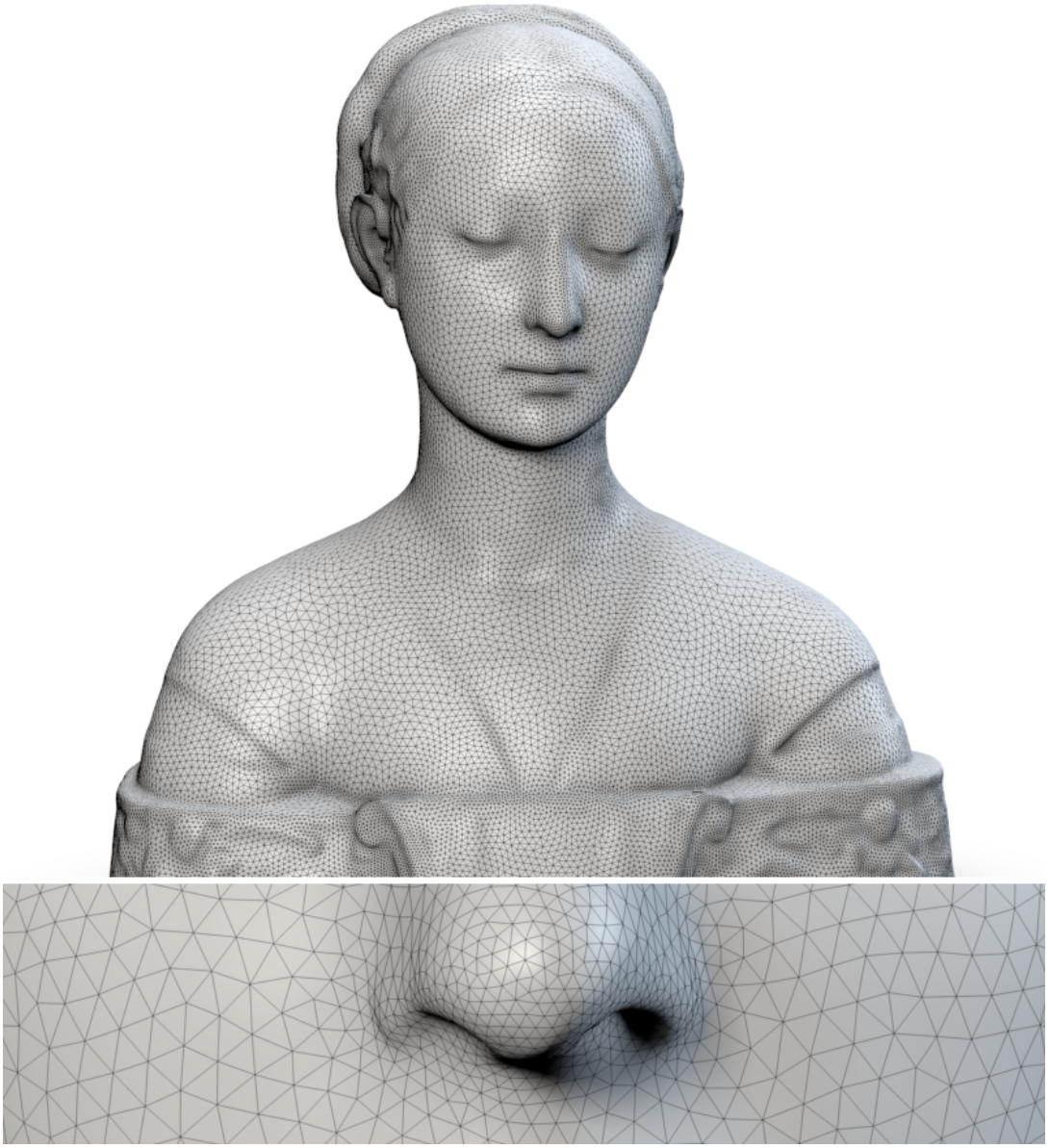

Figure 7.14: The *CNR Laurana* dataset adaptively remeshed with 50 000 vertices. The bottom shows the nose region where the adaptive sampling is clearly visible.

Squirrel Dataset

The squirrel dataset has been reconstructed from 92 images and yields a total of 327 140 mesh vertices after surface reconstruction with *FSSR*. Because the surface has many smooth parts with little curvature, not many triangles are required to reproduce most of the details. In Figure 7.15 the original mesh has been adaptively remeshed to 10 000 vertices.

Middlebury Temple Dataset

The reconstruction of the *Middlebury Temple* dataset using our pipeline described in Chapter 3 is one of the top-performing results on the Middlebury Multi-View Stereo benchmark. Here, the original reconstruction using *FSSR* with 233 674 vertices has been uniformly remeshed to 50 000 vertices and adaptively remeshed to 25 000 vertices, see Figure 7.16. The uniform remesh requires relatively many samples to avoid considerable loss of detail.

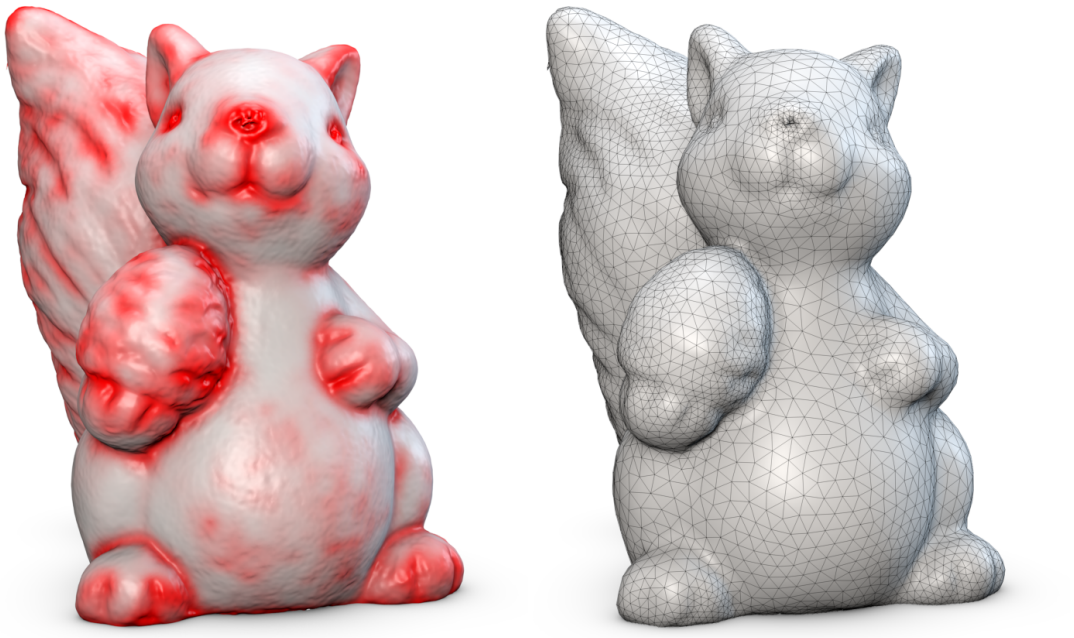

Figure 7.15: Remeshing of the *Squirrel* mesh from 327 140 vertices to 10 000 vertices. The vertex distribution is controlled by the density field visualized on the left.

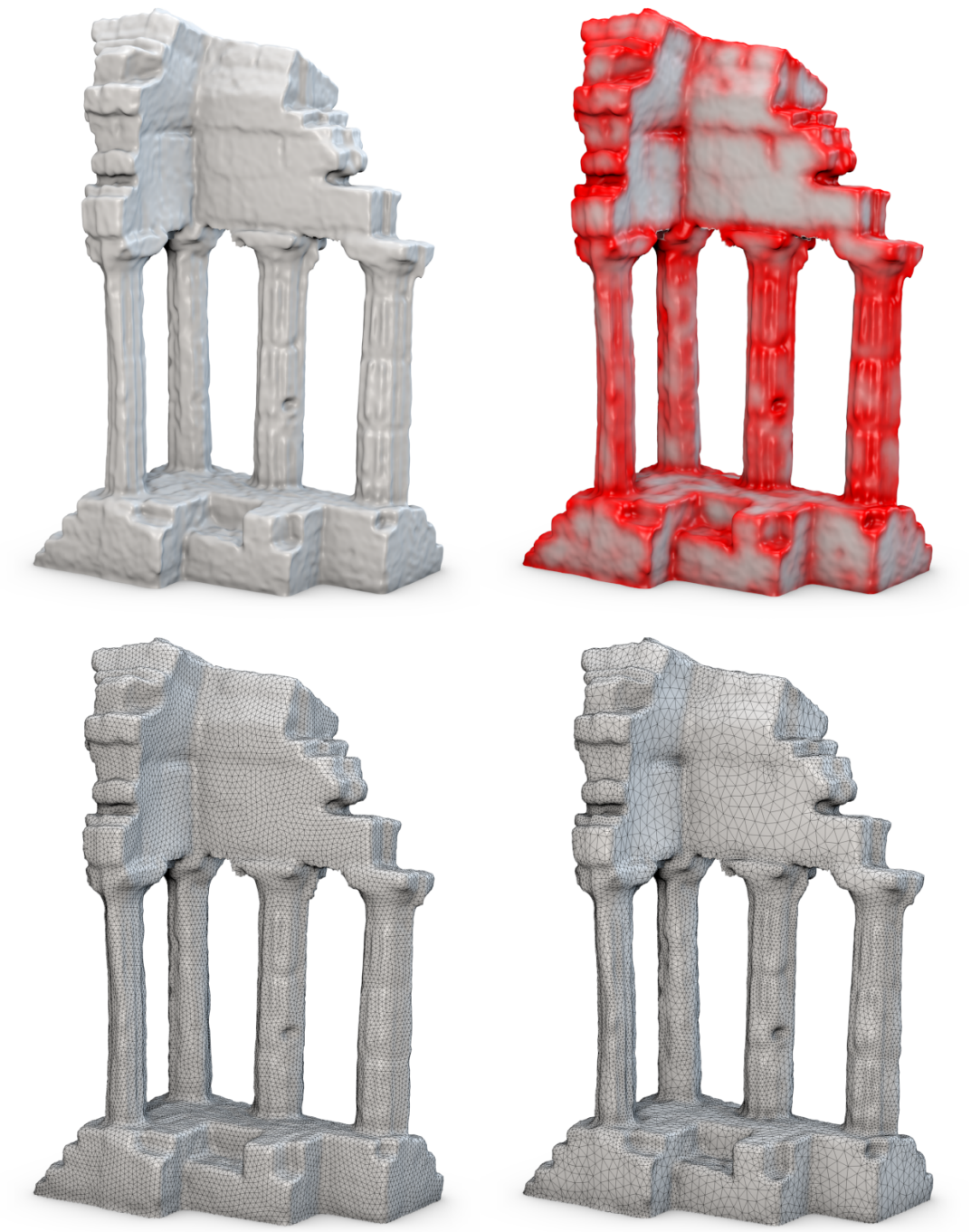

Figure 7.16: Remeshing results on the *Middlebury Temple*. The original mesh (top left, 233 663 vertices, triangles not shown) has been uniformly remeshed (bottom left, 50 000 vertices) and adaptively remeshed (bottom right, 25 000 vertices) using the density field (top right) computed according to mesh curvature.

7.6 Conclusion

From a surface quality point of view, we belief that a single scale approximation to the signed distance field as proposed by Curless and Levoy [1996], namely *VRIP*, produces excellent results with well studied quality guarantees on uniform datasets. This is reflected in the results obtained with *DMFusion* on a single octree level. However, most real-world datasets are not uniform in scale. Due to the scale space discretization performed by *DMFusion*, some visible artifacts can occur as demonstrated, e.g., in Figure 7.2. These artifacts become less visible with an increasing amount of redundancy. But nevertheless, this makes the *DMFusion* approach less practical for very clean and controlled data. Empirically, *FSSR* produces better results in all cases, with uniform as well as multi-scale data.

The evaluation identified an interesting issue of the *FSSR* approach. Because the approach employs a sum of non-linear basis functions, the implicit function is non-linear as well. This causes a potential issue with the particular isosurface extraction technique, namely a variant of the Marching Cubes algorithm [Kazhdan et al., 2007]. Because Marching Cubes performs a linear interpolation to find the precise surface position, the implicit function is assumed to be linear as well. If this is not the case, the interpolated surface position will be inaccurate. These inaccuracies are systematic and related to the surface orientation with respect to the chosen octree discretization. This can sometimes be observed as mild waving or ringing artifacts on the surface. Solving this issue is an interesting research direction and relevant for many other methods. As the sampling density of the implicit function

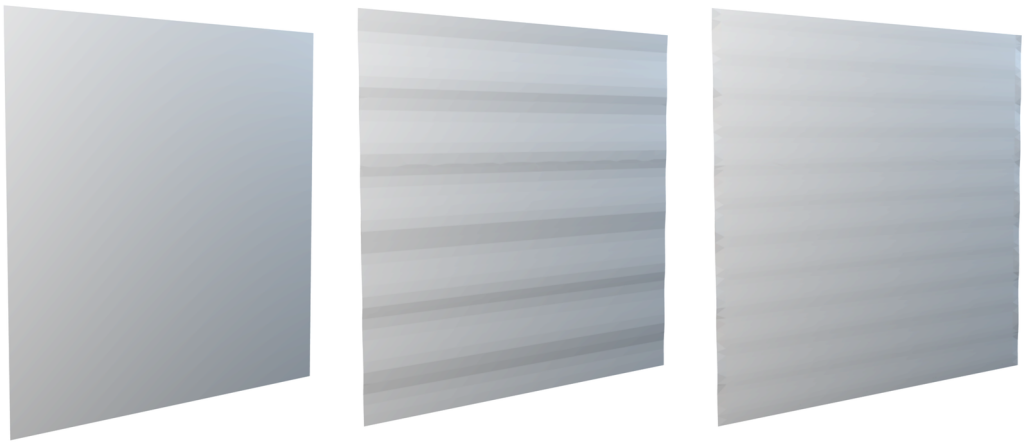

Figure 7.17: Synthetically generated samples of a planar surface are used for reconstruction. Depending on the orientation of the samples with respect to the global coordinate system, wavy reconstruction artifacts can occur because Marching Cubes' assumption of a linear basis function are not met. *Left:* The plane is aligned with the global coordinate system axis, no artifacts are visible. *Middle:* A slight rotation of the plane reveals mild artifacts. *Right:* A denser sampling of the implicit function reduces the artifacts. Note: All images are contrast enhanced.

approaches infinity, the implicit function becomes more linear within each octree cell. This implies that the artifacts can be considerably reduced with a denser sampling, which is demonstrated in Figure 7.17. One easy solution is to refine the octree hierarchy at the cost of increased resource consumption. Another possible solution is to make use of Hermite data, which allows for Hermite interpolation to improve surface localization.

It has been shown that both *DMFusion* and *FSSR* scale well to large datasets. Many other approaches require the solution of a global linear system of equations and are therefore not applicable to very large datasets. In contrast, the *FSSR* approach does not perform any global operations. The multi-scale reconstruction using *DMFusion* requires the construction of a global Delaunay tetrahedralization, which is clearly a bottleneck both in running time and memory consumption. The running time of *FSSR* is usually between the single-scale and multi-scale reconstruction performed with *DMFusion*. However, the single-scale reconstruction is only possible on uniform data and thus impractical for multi-scale datasets. Overall, *FSSR* is the more memory efficient choice.

Because both approaches are not designed to reconstruct and extend surfaces beyond the available data, holes can occur due to insufficient sampling. Although this is very useful in datasets with incomplete coverage such as outdoor scenes, it often leaves small holes on objects that should instead be watertight. If this is undesirable, hole-filling algorithms must be applied as a post-processing step.

The remeshing approach presented in Chapter 6 is capable of creating triangle meshes with improved triangle quality and drastically reduced triangle count if desired. This remeshing step is often necessary before texturing, transmission or even rendering of the reconstructed surfaces. Although the presented remeshing approach is not yet capable of handling large-scale meshes or drastic scale differences, it is clearly capable of improving the mesh quality on many scanner and MVS datasets. The use of a multi-scale density function and modifications to handle large-scale meshes are interesting future directions.

Chapter 8

Conclusion

Contents

8.1 Summary

In this thesis we presented an image-based reconstruction system that is able to cope with uncontrolled input images and enables non-expert users to quickly create new datasets. In contrast to controlled datasets, the images exhibit varying resolution, different lighting conditions, and drastic changes in sample scale, which leads to much more noise and outliers in the geometry. Figure 8.1 shows the *Trevi Fountain* reconstructed solely from 871 images downloaded from Flickr, a popular community photo collection.

The multi-scale capabilities of the system allow users to put an emphasis on certain details in the scene by providing more close-up photos of these regions. These capabilities are a result of explicitly modeling the scale of every sample point and utilizing this valuable information in the surface reconstruction step. As a result, the final surface represents every region at the corresponding scale available from the input data. Although both presented methods excel at handling multi-scale input, controlled datasets are handled with ease and the results compare favorably to state-of-the-art methods. Figure 8.2 shows some examples.

8.2 Discussion

The presented approaches shed some light on the aspect of scale in surface reconstruction that has mostly been neglected in the literature so far. In particular, we discuss sample scale, surface resolution and the relation to surface curvature.

Figure 8.1: The *Trevi Fountain* dataset reconstructed from 871 uncontrolled images downloaded from the community photo collection Flickr.

Figure 8.2: Several controlled datasets. *Left:* The *Snow Owl* dataset reconstructed from 61 images. *Middle:* The *Scared Owl* dataset reconstructed from 212 images. *Right:* The *Squirrel* dataset reconstructed from 92 images.

Scale, Resolution and Curvature

The aspect of sample scale, i.e., the physical extent of the sample, can be modelled as an additional scalar value per sample and is thus purely a property of the input data. This information can effectively be used to identify redundancy and avoid mixing samples at incompatible scale. We demonstrated that utilizing this information produces superior results compared to most approaches that neglect scale.

The resolution of the output surface, i.e., the decision at what scale the output surface should be represented, is not necessarily related to the input scale. The reconstruction approaches presented in Chapter 4 and 5 choose the highest resolution available to accurately represent any geometric detail in the input data. However, this often leads to over-tessellation in regions without geometric detail, and a simplification of these regions is reasonable. The remeshing approach presented in Chapter 6 is capable of producing a new tessellation that is more adapted to the actual detail in the geometry. To this end, the notion of surface curvature seems fitting and is inherently multi-scale. Intuitively, regions of higher curvature require a finer tessellation to represent detail. The precise connection between the estimated surface curvature, the vertex distribution and density, and the corresponding surface approximation error is, however, not well understood and left for future research.

The required resolution of the output surface is often determined from geometric properties of the surface. Although geometry itself can often be drastically simplified without a noticeable loss of accuracy, other per-vertex attributes (such as color) cannot be faithfully represented using the coarse tessellation. If the preservation of these attributes is desired, surface texturing algorithms can be used in order to delegate these attributes to dedicated textures.

Redundancy

The image-based reconstruction pipeline presented in Chapter 3 computes a depth map for every input image, and reconstructs the final surface from the union of all depth maps. In typical image-based scenarios, the input images are densely sampled around the surface, yielding many depth maps. The resulting redundancy can mostly be considered positive as it leads to a noise reduction by averaging many samples. On the negative side, a large amount of redundancy causes an increase in storage requirements, and much longer computation times for Multi-View Stereo and surface reconstruction. This becomes more problematic when working with video data with many small-baseline frames. Here, computing a depth map for every input frame and the resulting surface reconstruction task become prohibitively expensive. It seems natural to employ specialized Multi-View Stereo algorithms that exploit the special structure of video input, and we are interested in exploring these approaches in future work. A common solution is to reconstruct only a subset of the depth maps and use the visual redundancy to produce more accurate depth maps in the first place. But ultimately, a per-view representation seems unsuitable for this kind of input, and global representations such as meshes, volumes or point clouds might be more suitable.

Data Imperfections

One of the most difficult aspects of the reconstruction task is to deal with all sorts of imperfections in the data. The most obvious imperfections are noise and outliers. However, mis-alignments in the input data that exceed a certain level are very difficult to handle. Experience has shown that mis-alignments in multi-scale datasets are particularly challenging. Imprecise camera calibration is a common issue, and input photos that represent the surface at very different scales are not well aligned to one another. The resulting mis-alignments can lead to duplicate surfaces when merged. We argue that these problems must be addressed at the camera calibration or scan alignment stage. Brown and Rusinkiewicz [2007] attempt to solve these issues using a non-rigid alignment of 3D scans. Furukawa and Ponce [2009] address the problem of inaccurate camera calibration, but these attempts have not been evaluated in the presence of multi-scale data.

Another common imperfection is incomplete coverage due to occlusions, which often leads to small holes in the surface. Many global reconstruction approaches provide excellent hole-filling capabilities but often create extraneous surfaces way beyond the data support, which requires manual cleanup. We argue that small holes in the surface are undesirable and have to be closed in order to obtain a watertight surface. The incorporation of hole-filling capabilities in the presented surface reconstruction approaches has not been explored and is a relevant and interesting challenge. A hypothetical pre-processing solution is to introduce additional samples in regions with insufficient data, which can be difficult to realize especially in streaming reconstruction approaches. A simple post-processing solution is the application of a triangulation-based hole-filling algorithm once a surface is reconstructed. Holes with complex boundary shapes are difficult for triangulation-based methods. Volumetric diffusion approaches [Davis et al., 2002] have successfully been applied to compactly supported distance functions on uniform grids. An extension to multi-scale representations is an interesting future direction.

8.3 Future Work

In the previous section we identified aspects of the reconstruction pipeline that can benefit from further research. Handling mis-alignments in the camera calibration step, employing specialized stereo algorithms for video and more densely sampled datasets as well as incorporating hole-filling approaches seem interesting research problems that can improve the results.

At the heart of this thesis is *FSSR*, the Floating Scale Surface Reconstruction approach. Here, we wish to extent the algorithm to handle larger datasets. In its current implementation, all samples as well as the octree are completely contained in memory. One possible extension is a streaming implementation of *FSSR*, where samples are loaded in a streaming fashion while the octree is updated. To further reduce memory requirements, only parts of the octree may be contained in memory, resulting in an out-of-core reconstruction strategy. Such a steaming approach may

require careful sorting of the input samples to prevent excessive swapping of octree sub-hierarchies. We also want to investigate reducing the processing time by making use of massively parallel hardware. In particular, the evaluation of the basis functions associated with each sample is an independent process and can therefore be performed in parallel. Currently, only CPU parallelization is considered. Finally, performance may be improved by reducing the number of input samples, e.g., by aggregating redundant samples in a pre-processing step.

One potential issue with the surface extraction quality of *FSSR* has been identified in Chapter 7. Isosurface extraction algorithms usually employ linear interpolation to find the position of the surface with sub-voxel accuracy. As a consequence, if the underlying implicit function is not linear, surface contouring can lead to imprecise isosurface positions. In order to eliminate the resulting artifacts, Hermite data interpolation may be considered to obtain more accurate surface positions. Finally, the resolution of the resulting surface often contains many small triangles even in regions without geometric detail. Making the surface extraction adaptive to the actual geometry by directly simplifying the octree hierarchy is a promising idea.

The surface remeshing approach presented in Chapter 6 is able to better control the resolution of the final geometry in order to avoid over-tessellated regions without geometric detail. The presence of detail can be detected by employing measures of surface curvature. Since the curvature is inherently a multi-scale property, this idea seems fitting in the context of this thesis. However, the precise relationship between the curvature field and the required vertex density to obtain a scale-adaptive approximation threshold is not yet well understood and left for future work. The applicability of the remeshing algorithm to large, multi-resolution meshes is a relevant step towards higher quality meshes that render faster and allow more efficient post-processing, e.g., texturing of the mesh.

(Co-)Authored Publications

Jens Ackermann, Fabian Langguth, **Simon Fuhrmann**, Arjan Kuijper and Michael Goesele. Multi-View Photometric Stereo by Example. *Proceedings of the International Conference on 3D Vision*, Tokyo, Japan, 2014.

Simon Fuhrmann, Fabian Langguth and Michael Goesele. MVE – A Multi-View Reconstruction Environment. *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage*, Darmstadt, Germany, 2014.

Simon Fuhrmann and Michael Goesele. Floating Scale Surface Reconstruction. *ACM Transactions on Graphics*, Proceedings of ACM SIGGRAPH, Vancouver, Canada. 2014.

Jens Ackermann, **Simon Fuhrmann** and Michael Goesele. Geometric Point Light Source Calibration. *Proceedings of Vision, Modeling and Visualization*, Lugano, Switzerland. 2013.

Jens Ackermann, Fabian Langguth, **Simon Fuhrmann** and Michael Goesele. Photometric Stereo for Outdoor Webcams. *Proceedings of Computer Vision and Pattern Recognition*, Providence, USA. 2012.

Simon Fuhrmann and Michael Goesele. Fusion of Depth Maps with Multiple Scales. *ACM Transactions on Graphics*, Proceedings of ACM SIGGRAPH Asia, Hong Kong, China. 2011.

Meike Becker, Matthias Kirschner, **Simon Fuhrmann** and Stefan Wesarg. Automatic Construction of Statistical Shape Models for Vertebrae. *Medical Image Computing and Computer Assisted Intervention*, Toronto, Canada. 2011.

Simon Fuhrmann, Jens Ackermann, Thomas Kalbe and Michael Goesele. Direct Resampling for Isotropic Surface Remeshing. *Proceedings of Vision, Modeling and Visualization*, Siegen, Germany. 2010.

Michael Goesele, Jens Ackermann, **Simon Fuhrmann**, Carsten Haubold, Ronny Klowsky, Drew Steedly and Richard Szeliski. Ambient Point Clouds for View Interpolation. *ACM Transactions on Graphics*, Proceedings of ACM SIGGRAPH 2010, Los Angeles, USA. 2010.

Michael Goesele, Jens Ackermann, **Simon Fuhrmann**, Ronny Klowsky, Fabian Langguth, Patrick Muecke and Martin Ritz. Scene Reconstruction from Community Photo Collections. *IEEE Computer*, Issue June 2010, Invited Paper. 2010.

Thomas Kalbe, **Simon Fuhrmann**, Stefan Uhrig, Frank Zeilfelder and Arjan Kuijper. A New Projection Method for Point-Set Surfaces. *European Association for Computer Graphics*, Annex to the Eurographics Conference Proceedings. Munich, Germany. 2009.

Bibliography

Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building Rome in a Day. *IEEE 12th International Conference on Computer Vision*, pages 72–79, September 2009.

Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Point Set Surfaces. *IEEE Visualization*, pages 21–537, 2001.

Pierre Alliez, E. C. de Verdire, Olivier Devillers, and Martin Isenburg. Isotropic Surface Remeshing. *Shape Modeling International*, pages 49–58, 2003.

Pierre Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based Variational Reconstruction of Unoriented Point Sets. In *Eurographics Symposium on Geometry Processing*, pages 39–48, 2007.

Murat Arikan, Reinhold Preiner, Claus Scheiblauer, Stefan Jeschke, and Michael Wimmer. Large-Scale Point-Cloud Visualization through Localized Textured Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 20(9):1280–1292, September 2014.

Martin Armstrong, Andrew Zisserman, and Paul A. Beardsley. Euclidean Reconstruction from Uncalibrated Images. In *British Machine Vision Conference*, pages 509–518, 1994.

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Vangool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio T. Silva, and Gabriel Taubin. The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, October 1999.

James F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

Matthew Bolitho, Michael Kazhdan, Randal Burns, and Hugues Hoppe. Multilevel Streaming for Out-of-Core Surface Reconstruction. In *Eurographics Symposium on Geometry Processing*, 2007.

Mario Botsch and Leif P. Kobbelt. A Robust Procedure to Eliminate Degenerate Faces from Triangle Meshes. In *Vision, Modeling, and Visualization*, 2001.

Benedict J. Brown and Szymon Rusinkiewicz. Global Non-Rigid Alignment of 3-D Scans. In *ACM SIGGRAPH Computer Graphics*, page 21, 2007.

Faith Calakli and Gabriel Taubin. SSD: Smooth Signed Distance Surface Reconstruction. *Computer Graphics Forum*, 30(7):1993–2002, September 2011.

Jonathan C. Carr, Rick K. Beatson, J. B. Cherrie, T. J. Mitchell, W. Richard Fright, Bruce C. McCallum, and T. R. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, pages 67–76, New York, New York, USA, 2001. ACM Press.

Y. Chen and G. Medioni. Object Modeling by Registration of Multiple Range Images. In *IEEE International Conference on Robotics and Automation*, pages 2724–2729, 1991.

Paolo Cignoni, Claudio Montani, Claudio Rocchini, and Roberto Scopigno. A General Method for Preserving Attribute Values on Simplified Meshes. In *IEEE Visualization*, pages 59–66, 1998.

Brian Curless and Marc Levoy. A Volumetric Method for Building Complex Models from Range Images. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, pages 303–312, 1996.

J. Davis, S.R. Marschner, M. Garr, and M. Levoy. Filling Holes in Complex Surfaces using Volumetric Diffusion. In *3D Data Processing, Visualization and Transmission*, 2002.

Akio Doi and Akio Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE Transactions on Information and Systems*, 74(1): 214–224, 1991.

Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Review*, 41(4):637, 1999.

Nira Dyn, Kai Hormann, Sun-jeong Kim, and David Levin. Optimizing 3D Triangulations Using Discrete Curvature Analysis. In *Mathematical methods for curves and surfaces*, pages 135–146. 2001.

Herbert Edelsbrunner and Ernst P. Mücke. Three-Dimensional Alpha Shapes. *ACM Transactions on Graphics*, 13(1):43–72, January 1994.

Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating Textures. *Computer Graphics Forum*, 27(2):409–418, 2008.

Jakob Engel, Thomas Schoeps, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision*, pages 1–16, Cham, 2014. Springer International Publishing.

Carlos Estrada and Juan D. Tard. Hierarchical SLAM: Real-Time Accurate Mapping of Large Environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005.

Olivier Faugeras and Renaud Keriven. Variational principles, Surface Evolution, PDE's, Level Set Methods and the Stereo Problem. Technical report, Inria Sophia Antipolis, 2006.

Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

Michael Floater. Mean Value Coordinates. *Computer Aided Geometric Design*, 20(1): 19–27, March 2003.

Yan Fu and Bingfeng Zhou. Direct Sampling on Surfaces for High Quality Remeshing. *Computer Aided Geometric Design*, 26(6):711–723, August 2009.

Simon Fuhrmann. Curvature-adaptive and Feature-sensitive Isotropic Surface Remeshing. *TU Darmstadt*, 2009.

Simon Fuhrmann and Michael Goesele. Fusion of Depth Maps with Multiple Scales. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia)*, number December, page 1, New York, New York, USA, 2011. ACM Press.

Simon Fuhrmann and Michael Goesele. Floating Scale Surface Reconstruction. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, number July, 2014.

Simon Fuhrmann, Jens Ackermann, Thomas Kalbe, and Michael Goesele. Direct Resampling for Isotropic Surface Remeshing. In *Vision, Modeling, and Visualization*, 2010.

Simon Fuhrmann, Fabian Langguth, and Michael Goesele. MVE – A Multi-View Reconstruction Environment. In *Eurographics Workshop on Graphics and Cultural Heritage*, 2014.

Yasutaka Furukawa and Jean Ponce. Accurate Camera Calibration from Multi-View Stereo and Bundle Adjustment. *International Journal of Computer Vision*, 84(3): 257–268, April 2009.

Yasutaka Furukawa and Jean Ponce. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–76, August 2010.

Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Towards Internet-scale multi-view stereo. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1434–1441, June 2010.

David Gallup, Jan-Michael Frahm, Philippos Mordohai, Qingxiong Yang, and Marc Pollefeys. Real-Time Plane-Sweeping Stereo with Multiple Sweeping Directions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2007.

Michael Goesele, Brian Curless, and Steven M. Seitz. Multi-View Stereo Revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2402–2409. IEEE, 2006.

Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-View Stereo for Community Photo Collections. *IEEE 11th International Conference on Computer Vision*, pages 1–8, October 2007.

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The Lumigraph. In *Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 43–54, New York, New York, USA, 1996. ACM Press.

Gaël Guennebaud and Markus Gross. Algebraic Point Set Surfaces. *ACM Transactions on Graphics*, 26(3):23, July 2007.

Leonidas J. Guibas, Donald E. Knuth, and Micha Sharir. Randomized Incremental Construction of Delaunay and Voronoi Diagrams. *Algorithmica*, 7(1-6):381–413, 1992.

Robert M. Haralick, Chung-Nan Lee, Karsten Ottenberg, and Michael Nölle. Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem. *IEEE International Journal of Computer Vision*, 356:331–356, 1994.

Richard Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997.

Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. ISBN 0521540518.

Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation Averaging. *International Journal of Computer Vision*, 103(3):267–305, January 2013.

Richard I. Hartley. Euclidean Reconstruction from Uncalibrated Views. In *Applications of Invariance in Computer Vision*, pages 235–256, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

Wilfried Hartmann, Michal Havlena, and Konrad Schindler. Predicting Matchability. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9–16, June 2014.

Michal Havlena and Konrad Schindler. VocMatch: Efficient Multiview Correspondence for Structure from Motion. In *European Conference on Computer Vision*, 2014.

Carlos Hernandez and Francis Schmitt. Silhouette and Stereo Fusion for Object Modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004.

Kazunori Higuchi, Martial Herbert, and Katsushi Ikeuchi. Building 3D Models from Unregistered Range Images. In *IEEE Conference on Robotics and Automation*, pages 2248–2253, 1994.

A. Hilton and J. Illingworth. Multi-Resolution Geometric Fusion. In *International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 181–188. IEEE Comput. Soc. Press, 1997.

A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt. Reliable Surface Reconstruction from Multiple Range Images. *European Conference on Computer Vision*, pages 117–126, 1996.

Hugues Hoppe. New Quadric Metric for Simplifiying Meshes with Appearance Attributes. In *IEEE Visualization*, pages 59–66, 1999.

Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface Reconstruction from Unorganized Points. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, pages 71–78, July 1992.

Wolfgang Hormann, Josef Leydold, and Gerhard Derflinger. Automatic Nonuniform Random Variate Generation. *Statistics and Computing*, 2004.

Nianjuan Jiang, Zhaopeng Cui, and Ping Tan. A Global Linear Method for Camera Pose Registration. *IEEE 14th International Conference on Computer Vision*, pages 481–488, December 2013.

Xiangmin Jiao and Michael T. Heath. Feature Detection for Surface Meshes. In *International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 705–714, 2002.

Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual Contouring of Hermite Data. *ACM Transactions on Graphics*, 21(3), July 2002.

Michael Kazhdan. Reconstruction of Solid Models from Oriented Point Sets. In *Eurographics symposium on Geometry processing*, 2005.

Michael Kazhdan and Hugues Hoppe. Screened Poisson Surface Reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, June 2013.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In *Eurographics Symposium on Geometry Processing*, pages 61–70, New York, New York, USA, 2006. Eurographics Association.

Michael Kazhdan, Allison Klein, Ketan Dalal, and Hugues Hoppe. Unconstrained Isosurface Extraction on Arbitrary Octrees. In *Eurographics symposium on Geometry processing*, pages 125–133, 2007.

Ronny Klowsky, Arjan Kuijper, and Michael Goesele. Modulation Transfer Function of Patch-based Stereo Systems. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, June 2012.

Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation. In *Computer Vision and Pattern Recognition*, pages 2969–2976, 2011.

Kalin Kolev, Thomas Brox, and Daniel Cremers. Fast Joint Estimation of Silhouettes and Dense 3D Geometry from Multiple Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):493–505, March 2012.

Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. *IEEE International Conference on Robotics and Automation*, pages 3607–3613, May 2011.

Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts. *International Conference on Computer Vision*, pages 1–8, 2007.

Victor Lempitsky and Denis Ivanov. Seamless Mosaicing of Image-Based Texture Maps. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6, June 2007.

Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. *International Conference on Computer Vision*, pages 2548–2555, November 2011.

David Levin. The Approximation Power of Moving Least-Squares. *Mathematics of Computation*, 67(224):1517–1532, October 1998.

Marc Levoy, Jeremy Ginsberg, Jonathan Shade, Duane Fulk, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, and James Davis. The Digital Michelangelo Project. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, pages 131–144, New York, New York, USA, 2000. ACM Press.

Tony Lindeberg. Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision*, 30:79–116, 1998.

S. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

Charles T. Loop. *Smooth Subdivision Surfaces Based on Triangles*. M.s. thesis, University of Utah, 1987.

William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 163–169. ACM, 1987.

Manolis I. A. Lourakis and Antonis A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Transactions on Mathematical Software*, 36(1):1–30, March 2009.

David G. Lowe. Object Recognition from Local Scale-Invariant Features. *International Conference on Computer Vision*, pages 1150–1157, 1999.

David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

Q. Luong and O. D. Faugeras. The Fundamental matrix: Theory, Algorithms, and Stability Analysis. *International Journal of Computer Vision*, 17:43–75, 1995.

Don P. Mitchell. Generating Antialiased Images at Low Sampling Densities. In *Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 65–72, 1987.

Masahiro Mori, Karl F. MacDorman, and Norri Kageki. The Uncanny Valley. *IEEE Robotics and Automation Magazine*, 19:98–100, 2012.

Pierre Moulon, Pascal Monasse, and Renaud Marlet. Global Fusion of Relative Motions for Robust, Accurate and Scalable Structure from Motion. In *International Conference on Computer Vision*, 2013.

Patrick Mücke, Ronny Klowsky, and Michael Goesele. Surface Reconstruction From Multi-Resolution Sample Points. *Vision, Modeling, and Visualization*, 2011.

Marius Muja and David G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP'09)*, pages 331–340, 2009.

Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. DTAM: Dense Tracking and Mapping in Real-Time. In *International Conference on Computer Vision*, pages 2320–2327. IEEE, November 2011.

David Nistér. An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–77, June 2004.

David Nister and Henrik Stewenius. Scalable Recognition with a Vocabulary Tree. In *Conference on Computer Vision and Pattern Recognition*, pages 2161–2168. IEEE, 2006.

Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 22(3):463–470, July 2003.

Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Sparse Surface Reconstruction with Adaptive Partition of Unity and Radial Basis Functions. *Graphical Models*, 68(1):15–24, January 2006.

Gabriel Peyré and Laurent D. Cohen. Geodesic Remeshing using Front Propagation. *International Journal of Computer Vision*, 69(1):145–156, 2006.

R. Pito. Mesh Integration Based on Co-Measurements. In *IEEE International Conference on Image Processing*, pages 397–400. IEEE, 1996.

Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Intrinsic Camera Parameters. *International Journal of Computer Vision*, 1998.

Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Kurt Cornelis, and Jan Tops. 3D Recording for Archaeological Fieldwork. *IEEE Computer Graphics and Applications*, 23(3):20–27, 2003.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An Efficient Alternative to SIFT or SURF. *International Conference on Computer Vision*, pages 2564–2571, November 2011.

Pedro V. Sander, Diego Nehab, and Joshua Barczak. Fast Triangle Reordering for Vertex Locality and Reduced Overdraw. *ACM Transactions on Graphics*, 26(3):89, July 2007.

Scott Schaefer and Joe Warren. Dual Marching Cubes: Primal Contouring of Dual Grids. *Computer Graphics Forum*, 24(2):195–201, June 2005.

Daniel Scharstein and Richard Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 47(1):7–42, 2002.

William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *ACM SIGGRAPH Computer Graphics*, 26(2):65–70, July 1992.

W.J. Schroeder, B. Geveci, and M. Malaterre. Compatible Triangulations of Spatial Decompositions. *IEEE Visualization*, pages 211–217, 2004.

Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 519–528. IEEE, 2006.

Chen Shen, James F. O'Brien, and Jonathan R. Shewchuk. Interpolating and Approximating Implicit Surfaces from Polygon Soup. *ACM Transactions on Graphics*, 23 (3):896, August 2004.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, pages 835–846. ACM, 2006.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. Skeletal Graphs for Efficient Structure from Motion. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

Marc Soucy and Denis Laurendeau. Building a Surface Model of an Object using Multiple Range Views. In *Proceedings of SPIE*, page 85, 1992.

Marc Soucy and Denis Laurendeau. A General Surface Approach to the Integration of a Set of Range Views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):344–358, April 1995.

Vitaly Surazhsky and Craig Gotsman. Explicit Surface Remeshing. In *Eurographics Symposium on Geometry Processing*, pages 17–28, 2003.

Vitaly Surazhsky, Pierre Alliez, and Craig Gotsman. Isotropic Remeshing of Surfaces: a Local Parameterization Approach. In *12th International Meshing Roundtable*, 2003.

Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer International Publishing, 2010. URL http://szeliski.org/Book/.

Christian Theobalt, Naveed Ahmed, Hendrik Lensch, Marcus Magnor, and Hans-Peter Seidel. Seeing People in Different Light – Joint Shape, Motion, and Reflectance Capture. *IEEE Transactions on Visualization and Computer Graphics*, 13 (3):663–674, 2007.

Grit Thuermer and Charles A. Wuethrich. Computing Vertex Normals from Polygonal Facets. *Journal of Graphics Tools*, 3(1):43–46, 1998.

Engin Tola, Vincent Lepetit, and Pascal Fua. A Fast Local Descriptor for Dense Matching. In *Computer Vision and Pattern Recognition*, 2008.

Engin Tola, Vincent Lepetit, and Pascal Fua. DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, 2010.

James Tompkin, Ki Kim, Jan Kautz, and Christian Theobalt. Videoscapes: Exploring Sparse, Unstructured Video Collections. *ACM Transactions on Graphics ...*, 2012.

Greg Turk. Re-Tiling Polygonal Surfaces, 1992. ISSN 00978930.

Greg Turk and Marc Levoy. Zippered Polygon Meshes from Range Images. *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 311–318, 1994.

Greg Turk and James F. O'Brien. Variational Implicit Surfaces. Technical report, Georgia Institute of Technology, 1999.

Alex Vlachos, Jörg Peters, Chas Boyd, and Jason L. Mitchell. Curved PN triangles. In *Symposium on Interactive 3D Graphics*, pages 159–166, New York, New York, USA, 2001. ACM Press.

George Vogiatzis and Carlos Hernandez. Video-based, Real-Time Multi View Stereo. *Image and Vision Computing*, 2011.

Alexandre Vrubel, Olga R. P. Bellon, and Luciano Silva. A 3D Reconstruction Pipeline for Digital Preservation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2687–2694. IEEE, June 2009.

Hoang-Hiep Vu, Renaud Keriven, Patrick Labatut, and Jean-Philippe Pons. Towards High-Resolution Large-Scale Multi-View Stereo. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1430–1437, June 2009.

Hoang-Hiep Vu, Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. High Accuracy and Visibility-consistent dense Multiview Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):889–901, May 2012.

Michael Waechter, Nils Moehrle, and Michael Goesele. Let There Be Color! Large-Scale Texturing of 3D Reconstructions. In *European Conference on Computer Vision*, pages 836–850, Cham, 2014. Springer International Publishing.

R. Westermann, Leif P. Kobbelt, and T. Ertl. Real-time Exploration of Regular Volume Data by Adaptive Reconstruction of Iso-Surfaces. *The Visual Computer*, 15(2):100–111, 1999.

Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid, and Juan Tardós. A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 57(12):1188–1197, December 2009.

Kyle Wilson and Noah Snavely. Robust Global Translations with 1DSfM. In *European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 61–75, Cham, 2014. Springer International Publishing.

Changchang Wu. Towards Linear-Time Incremental Structure from Motion. *International Conference on 3D Vision*, pages 127–134, June 2013.

Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M. Seitz. Multicore Bundle Adjustment. *IEEE Conference on Computer Vision and Pattern Recognition*, (1):3057–3064, June 2011.

Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. Isotropic Remeshing with Fast and Exact Computation of Restricted Voronoi Diagram. *Computer Graphics Forum*, 28(5):1445–1454, July 2009.

Ruigang Yang and Marc Pollefeys. Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 211–217. IEEE, 2003.

Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Transactions on Graphics*, 32(1), January 2013.

Ramin Zabih and John Wood. Non-parametric Local Transforms for Computing Visual Correspondence. In *European Conference on Computer Vision*, number May, pages 151–158, 1994.

Christopher Zach, Thomas Pock, and Horst Bischof. A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration. In *IEEE 11th International Conference on Computer Vision (ICCV)*. Citeseer, 2007.

Wissenschaftlicher Werdegang des Autors[1]

2003 – 2009	Studium der Informatik Technische Universität Darmstadt
2007	Abschluss: Bachelor of Science Bachelor-Thesis: „Volume Data Generation from Triangle Meshes using the Signed Distance Function" Referenten: PD Dr. Frank Zeilfelder, Thomas Kalbe
2009	Abschluss: Master of Science Master-Thesis: „Feature-sensitive, Curvature-adaptive Isotropic Surface Remeshing" Referenten: Prof. Dr.-Ing. Michael Goesele, Thomas Kalbe
seit 2009	Wissenschaftlicher Mitarbeiter Graphics, Capture and Massively Parallel Computing (GCC) Technische Universität Darmstadt
2012 – 2013	Einjähriges Forschungspraktikum Google, Inc., Seattle, USA Entwicklung von "Lens Blur" für die Android Camera App

Ehrenwörtliche Erklärung[2]

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades „Doktor-Ingenieur" mit dem Titel *„Scene Reconstruction from Multi-Scale Input Data"* selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keinen Promotionsversuch unternommen.

Darmstadt, den 23. März 2015

Simon Fuhrmann

[1] Gemäß § 20 Abs. 3 der Promotionsordnung der Technischen Universität Darmstadt
[2] Gemäß § 9 Abs. 1 der Promotionsordnung der Technischen Universität Darmstadt