Adaptive Extraction and Representation of Geometric Structures from Unorganized 3D Point Sets

Dissertation

Eine dem Fachbereich Informatik der Technischen Universität Kaiserslautern
vorgelegte Arbeit zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

von Patric Keller

Kaiserslautern, 22. Juni 2009

Termin der wissenschaftlichen Aussprache: 20.11.2009

Dekan: Prof. Dr. Karsten Berns
Vorsitzender des Prüfungsausschusses: Prof. Dr. Karsten Berns
1. Berichterstatter: Prof. Dr. Hans Hagen
2. Berichterstatter: Prof. Dr. Bernd Hamann
3. Berichterstatter: Dr. habil. Martin Hering-Bertram

D 386

This work is devoted to
my parents
Regina and Hans Keller

© Copyright 2009
by Patric Keller

Acknowledgements

First of all I want to thank my supervisors Dr. Martin Hering-Bertram at the Fraunhofer Institute für Techno- und Wirtschaftsmathematik (ITWM) in Kaiserslautern, Prof. Hans Hagen at the University of Kaiserslautern and Prof. Bernd Hamann at the University of California, Davis for providing me with ideas and valuable advice. Especially, I want to thank Prof. Hagen for giving me the opportunity to perform all these studies. One special thank to Dr. Martin Hering-Bertram for his all-time support and guidance during this work. I also want to thank the members of the DFG's International Research Training Group (IRTG) and the Computer Graphics Research Group at Kaiserslautern for their advice and the interesting discussions. I thank Mady Bold and Inga Scheler for helping me managing all my foreign research stays and conference visits.

At the University of California, Davis I want to thank Louise H. Kellogg and Eric S. Cowgill of the Department of Geology and the other members of the KeckCAVES Research Group as well as the members of the LIDAR Research Group for providing me with valuable feedback and data sets. I also want to thank Oliver Kreylos, Marek Vanco for providing me with advice and data sets. I want to thank the scientific staff including Christoph Garth, Eduard Deines and Mario Hlawitschka at the Institute of Data Analysis and Visualization (IDAV) at the University of California, Davis for their support.

Abstract

The primary emphasis of this thesis concerns the extraction and representation of intrinsic properties of three-dimensional (3D) unorganized point clouds. The points establishing a point cloud as it mainly emerges from LiDaR (Light Detection and Ranging) scan devices or by reconstruction from two-dimensional (2D) image series represent discrete samples of real world objects. Depending on the type of scenery the data is generated from the resulting point cloud may exhibit a variety of different structures. Especially, in the case of environmental LiDaR scans the complexity of the corresponding point clouds is relatively high. Hence, finding new techniques allowing the efficient extraction and representation of the underlying structural entities becomes an important research issue of recent interest.

This thesis introduces new methods regarding the extraction and visualization of structural features like surfaces and curves (e.g. ridge-lines, creases) from 3D (environmental) point clouds. One main part concerns the extraction of curve-like features from environmental point data sets. It provides a new method supporting a stable feature extraction by incorporating a probability-based point classification scheme that characterizes individual points regarding their affiliation to surface-, curve- and volume-like structures. Another part is concerned with the surface reconstruction from (environmental) point clouds exhibiting objects that are more or less complex. A new method providing multi-resolutional surface representations from regular point clouds is discussed. Following the applied principles of this approach a volumetric surface reconstruction method based on the proposed classification scheme is introduced. It allows the reconstruction of surfaces from highly unstructured and noisy point data sets. Furthermore, contributions in the field of reconstructing 3D point clouds from 2D image series are provided. In addition, a discussion concerning the most important properties of (environmental) point clouds with respect to feature extraction is presented.

Zusammenfassung

Diese wissenschaftliche Arbeit handelt von der Extraktion und der Darstellung intrinsischer Eigenschaften von dreidimensionalen (3D) Punktwolken. Die zugehörigen Punkte in Form von 3D Ko- odinaten, wie sie z.B. von Laser-/LiDaR (Light Detection and Ranging) Scannern generiert, oder mithilfe von speziellen Algorithmen aus einer Serie von zweidimensionalen (2D) Bildern rekonstruiert werden, repräsentieren Objekte aus der realen Welt. Abhängig von der fokussierten Szenerie weisen die resultierenden Punktewolken eine Vielfalt unterschiedlichster Strukturen auf. Speziell im Fall von Landschafts- und Umgebungsscans besitzen diese eine hohe Komplexität. Die Entwicklung neuer und effizienter Methoden, um solche strukturellen Informationen zu filtern und darzustellen, ist ein wichtiger Forschungszweig in der Computer Grafik.

Diese Arbeit leistet einen Beitrag hinsichtlich der Extraktion und der Visualisierung besonderer Merkmale wie z.B. flächen- oder kurvenähnlicher Strukturen (ridge-lines, creases) aus 3D Punktewolken. Ein Teil beschäftigt sich dabei mit der Extraktion von kurvenähnlichen Strukturen aus Landschafts- und Umgebungscans. Das vorgestellte Verfahren verwendet ein neuartiges wahrscheinlichkeitsbasiertes Punkte-Klassifizierungsschema, welches die einzelnen Punkte entsprechend ihrer Zugehörigkeit zu flächen-, kurven- und volumenartigen Strukturen analysiert. Ein weiterer Teil dieser Arbeit betrifft die Rekonstruktion von Flächen aus 3D Punktewolken, welche eine moderate bis hohe Komplexität aufweisen. Dabei wird eine neue Methode vorgestellt, welche die Rekonstruktion von Flächen in multiplen Auflösungen erlaubt. Auf diesem Ansatz beruhend wird eine volumen-basierte, implizite Rekonstruktionsmethode eingeführt, die die Rekonstruktion von Flächen aus hochgradig komplexen und verrauschten Punktewolken, unter Verwendung des vorgestellten Punkte-Klassifizierungsschemas, gestattet. Zudem werden Beiträge hinsichtlich der Rekonstruktion von 3D Punktewolken aus einer Serie von 2D Bilder präsentiert, sowie einige der wichtigsten Eigenschaften von 3D Punktewolken, wie sie hauptsächlich von Umgebungsscans generiert werden, diskutiert. Diese Diskussion ist speziell auf die Extraktion von flächen- und kurvenähnlichen Strukturen hin ausgerichtet.

Contents

List of Figures vii

List of Tables ix

1 Introduction 1

1.1 Unorganized/Unstructured Data 1
1.2 Problems with Unstructured Point Data 3
1.3 Contribution of Dissertation 5
1.4 Motivating Applications 6
1.4.1 Applications in Geoscience 6
1.4.2 Applications in Reverse Engineering 7
1.4.3 Applications in Creating and Populating Virtual Environments . . 7
1.5 Overview of Dissertation 8

2 Point Sets 9

2.1 Overview 9
2.2 Data Acquisition 10
2.2.1 LiDaR Scanners 11
2.2.2 Radar Scanners 12
2.2.3 CT/MRI Scanners 13
2.3 Reconstruction of Point Clouds from 2D Images 14
2.3.1 Overview 14
2.3.2 The Projection Model 15
2.3.3 Epipolar Geometry 18
2.3.4 Related Work 21
2.3.5 Point Reconstruction 22
2.3.6 Resolving Image Correspondences 22

2.3.7 Maximum-Likelihood Based Parameter Reconstruction 27
2.3.8 Point Cloud Reconstruction 32
2.3.9 Results 33
2.3.10 Conclusions 38
2.4 Properties of Point Clouds 39
2.4.1 Noise 39
2.4.2 Object Occlusion 40
2.4.3 Sampling Density 41
2.4.4 Overlapping Point Samples 42
2.5 Data structures for Organizing Point Clouds 43
2.5.1 Binary Space Partitioning Trees 43
2.5.2 KD-Trees 44
2.5.3 Octrees 45
2.6 Examples 46

3 Feature-Based Analysis of Point Data 49
3.1 Overview 49
3.1.1 Previous Work 51
3.2 Principle Point Cloud Structures 53
3.2.1 Surfaces 53
3.2.2 Point Set Surfaces 54
3.2.3 Feature Lines 54
3.2.4 Feature Curves 56
3.2.5 Structural Ambiguities and Discontinuities 56
3.2.6 Terminology and Notation 56
3.3 Data Management 57
3.4 Identification of Feature Lines 57
3.4.1 Local Feature Size 58
3.4.2 Moving Least Squares Surface Fitting 58
3.4.3 Detecting Feature Lines 60
3.5 Point Classification 61
3.6 Probability-Based Point Classification 63
3.6.1 Examples 65
3.7 Extraction of Feature Graphs 67

3.7.1 Selecting Seed Nodes . . . 68
3.7.2 Edge Propagation . . . 70
3.7.3 Connecting Critical Nodes . . . 72
3.7.4 Graph Simplification . . . 73
3.7.5 Results . . . 74
3.8 Conclusions and Possible Future Work . . . 78

4 Surface Reconstruction 81
4.1 Overview . . . 81
4.2 Basics . . . 82
4.2.1 Surfaces . . . 82
4.2.2 Parametric Surfaces . . . 83
4.2.3 Implicit Surfaces . . . 86
4.2.4 Meshes . . . 87
4.2.5 Hybrid Meshes . . . 88
4.2.6 Surface Reconstruction . . . 89
4.2.7 Least Squares Surface Approximation . . . 89
4.2.8 Marching Cubes . . . 90
4.3 Related Work . . . 92
4.3.1 Parametric Surface Reconstruction . . . 92
4.3.2 Implicit Volumetric Surface Reconstruction . . . 93
4.3.3 Geometric Constrictive Approaches . . . 94
4.4 Multi-Resolution Surface Reconstruction . . . 95
4.4.1 Overview . . . 97
4.4.2 Hierarchical Spatial Partitioning . . . 98
4.4.3 Hybrid Mesh Wrapping . . . 99
4.4.4 Regular Mesh Refinement . . . 99
4.4.5 Irregular Mesh Refinement . . . 100
4.4.6 Vertex-Mapping . . . 103
4.4.7 Results . . . 104
4.5 Feature-Based Implicit Surface Reconstruction . . . 109
4.5.1 Overview . . . 110
4.5.2 Pre-Processing . . . 111
4.5.3 Scattered Data Approximation . . . 112

4.5.4 Initial Surface Extraction 116
4.5.5 Approximation of Local Maxima 117
4.5.6 Results 121
4.6 Comparison 125
4.7 Conclusions 125

5 Summary **129**

A Determining the Gauss-Weingarten Matrix **131**

B Least Squares Fitting **133**

References **135**

Curriculum Vitae **151**

List of Figures

1.1 Examples of different point data types. 2

2.1 Workflow diagram for data acquisition 9
2.2 Illustration of the principles of laser range scanning. 11
2.3 Examples images of CT/MRI scans. 13
2.4 Principle camera/projection model. 15
2.5 Epipolar geometry - an illustrative example. 19
2.6 Chessboard example image 33
2.7 Example demonstrating the reconstruction of point clouds using intensity based correspondence matching. 35
2.8 Example demonstrating the reconstruction using intensity based correspondence matching. 36
2.9 Example demonstrating the reconstruction using feature-based correspondence matching. 37
2.10 Noisy point clouds, reasons and examples. 39
2.11 Examples illustrating the occlusion problem at the point cloud acquisition. 40
2.12 Examples demonstrating the over- and undersampling of point clouds. . 42
2.13 Point cloud acquisition: Scan overlappings. 43
2.14 Examples illustrating the binary space partitioning scheme. 44
2.15 Examples illustrating the KD-tree based space partitioning scheme. . . . 45
2.16 Examples illustrating the octree based space partitioning scheme. 46
2.17 Examples of an environmental point cloud emerged from a terrestrial LiDaR scan. 47
2.18 Examples of point clouds emerged from stationary laser range scanning devices. 48
2.19 Point cloud examples of 3D reconstruction from 2D images. 48

3.1 Visualization of probability values for an water tower data set 66
3.2 Determination of feature curve related seed nodes and corresponding critical nodes from the test cube data set. 68
3.3 Determination of surface-related feature line seed nodes and the corresponding critical nodes from the curved test sheet data set. 69
3.4 Establishing edges between seed nodes 71
3.5 Incorporating critical nodes to the feature graph 72
3.6 Examples regarding feature curve extraction from relatively complex point structures. 75
3.7 Resulting feature graphs extracted from noisy test data sets. 76
3.8 Point classification and feature extraction performed on a geological data set acquired by airborne LiDaR scan. 79

4.1 Parametric B-spline surface patch. 83
4.2 Examples of an 2D simplicial complex and a triangular mesh. 87
4.3 Illustrative examples concerning regular and irregular hybrid mesh operations. 88
4.4 Principles of surface extraction using the marching cubes algorithm. . . 91
4.5 Illustration of the principle steps in reverse engineering. 95
4.6 Multi-resolution surface reconstruction: The principal steps. 97
4.7 Examples illustrating the principles of the applied regular and irregular mesh refinement of the HMW procedure. 99
4.8 Hybrid-Mesh Wrapping: Face-Propagation Rules 100
4.9 Examples of illustrating the principle approach for avoiding non-manifold mesh structures at the irregular mesh refinement. 102
4.10 Surface reconstruction: Vertex projection principle. 103
4.11 Example concerning mesh relaxation. 104
4.12 Limits of the multi-resolution surface reconstruction. 106
4.13 Multi-resolution surface reconstruction examples of the Stanford dragon point data set. 107
4.14 Multi-resolution surface reconstruction example turbine blade. 108
4.15 Multi-resolution surface reconstruction example concerning the skeleton hand. 108

4.16 2D example illustrating the scattered data approximation scheme used for constructing the desired feature field function. 114
4.17 2D example illustrating the extraction of the initial base mesh from the provided feature field. 117
4.18 Principle approach for approximating the meta-surface associated with the local maxima of the constructed feature field. 117
4.19 Test cube example demonstrating the topological changes of the extracted base mesh when varying the iso-value. 119
4.20 Example demonstrating the feature-based surface reconstruction from an environmental point cloud. 122
4.21 Example illustrating the impact of changing the neighborhood size used for point classification. 123
4.22 Example concerning the feature-based surface reconstruction from a modified Stanford bunny point data set. 127
4.23 Examples of the application of the multi-resolution reconstruction approach to the noisy bunny point data set. 128

List of Tables

2.1 Quality measurements regarding the reconstruction approaches. 34

4.1 Multi-resolution surface reconstruction: Computation times. 105
4.2 Multi-resolution surface reconstruction: Computation times for individual processing steps. 105
4.3 Multi-resolution surface reconstruction: Quality measures. 106
4.4 Computation times feature-based surface reconstruction approach. . . . 124

1 Introduction

1.1 Unorganized/Unstructured Data

The extraction and visualization of structures from unorganized point data sets is a challenging problem of current interest. By unorganized data in general we refer to computerized data or information that is unstructured in a sense that it does not possesses any underlying data model usable by computer programs or supporting its interpretation. Hence, handling or exploring such data without previous processing steps in most cases is difficult and hard to perform, especially, for large data sets of high dimensionality. The same holds for unstructured point data as it emerges in the field of computer graphics or scientific visualization. By unorganized or unstructured point data we hereby define discrete data samples of arbitrary dimensionality not exhibiting any additional information applicable for describing relationships between individual points. Considering this field of application processing the raw data in order to identify or extract useful structures and relations is an important problem frequently addressed by many scientists. The complexity of this problem becomes more clear when analyzing the principle nature of this kind of data. Depending on the field of application the data emerges we can distinguish between several types of unstructured point data. The most important ones are:

- *Statistical point data:*

  This type of data generally emerges through observations and surveys or is gathered by scientific experiments. Its dimensionality easily exceeds the "natural" limit of three dimensions. The process of extracting information or uncovering hidden patterns from unstructured data samples is called data mining. Amongst others, this includes the clustering - classification of data samples into similar groups - as well as the subsequent representation and visualization of the out-filtered information. The basic principle behind those techniques is to reduce redundancy

(a) (b)

Figure 1.1: (a) example of a simple 2D statistical point data set, (b) point cloud emerged from environmental LiDaR scan showing part of the campus at the University of California, Davis.

subject to certain conditions and to visually represent groups of similar samples in order to obtain a more specific insight. Figure 1.1a shows a simple example regarding statistical data.

- *Scientific point data:*

  This is the amount of data captured by scientific scan devices for example like light detection and ranging (LiDaR), and computer tomography (CT) / magnetic resonance imaging (MRI) scan devices or is produced by physically based simulations (e.g., fluid-, gas simulations). Principally, it models a real-world process or condition. The data samples mainly combine information like position, time, and physical quantities (e.g, temperature, mass, density etc.). In some cases this data is organized e.g., by grids (by grid-based simulations) whereas in others it is not. Especially, grid-less particle simulations produce large amounts of unstructured (time-varying) point data inheriting a great variety of relevant properties. Another source of high-grade unstructured scientific point data is provided by laser scan devices. The output forms unstructured point clouds representing samples of the scanned objects. This kind of data acquisition rather intends capturing geometric characteristics. Figure 1.1b shows a typical output of such scan devices in form of point clouds.

- *Geometric point data:*

  This includes data samples that primarily describe the shape of objects. Mostly, the data is generated artificially using computer aided design (CAD) methods or is derived from other data sets. In this context we particularly refer the reconstruction of 3D point data from 2D image series. In general, this point data captures geometric characteristics like the shape of objects.

Many existing algorithms aim at the extraction and visualization of this "hidden" information. This dissertation addresses the extraction and visualization of structures mostly from scientific/geometric point data sets in form of unstructured point clouds as it typically emerges from LiDaR scan devices or as reconstructed from a series of 2D photographs.

In this context, a point is associated with coordinates in 3D / 2D space. Throughout this dissertation, the terms (3D) point and (3D) point cloud always refer to a point or a set of points located within 3D space.

1.2 Problems with Unstructured Point Data

Depending on the application there are several problems that arise concerning the size of the data sets, the visualization and the processing of unstructured point data sets.

The size of the data sets is an important aspect that has to be considered since point data generated in the context of scientific experiments and measurements often reaches a critical size for which the efficient handling and visualization becomes difficult. For example, when considering point clouds produced by LiDaR scan devices or obtained by reconstruction from 2D images the point cloud generated easily can have millions of point samples. This becomes more important especially in the area of geoscience where immense point data sets are produced. Hence, finding methods for efficient data management is essential.

The visualization of point data has the purpose to provide a visual basis allowing exploration in a natural way. However, this becomes a quite challenging task when addressing unstructured point data representing complex objects and/or large environments. The vast amount of points may cause the observer to lose orientation when

navigating trough the point data. The representation of multiple point layers simultaneously causing the human eye to lose the ability to distinguish between near and far and complicates the process of visual perception (e.g., differentiation between the front and back of objects). As a further consequence of this, boundaries between objects are hard to recognize. Also, helpful structures apparent on real world objects e.g., like ridges, creases are difficult to identify due to the missing continuity of the point samples. Missing depth perception induced by non-existing lighting information is another issue leading to a possible miss-interpretation of such data.

Processing point data sets becomes necessary when specific information inherited by the raw data needs to be collected. The processing either affects the data set partially or entirely depending on what type of information has to be extracted. Some possible operations requiring processing are:

- *Filtering.* These operations concerns the identification and the extraction of special parts of the point data matching predefined conditions.
- *Point-based rendering.* This technique aims at improving visual appearance by associating lighting information with the individual point samples. However, it requires the extraction of well-defined normal vectors.
- *Feature extraction.* This allows the extraction of special information capable to describe structural and topological aspects of the point data.
- *Surface reconstruction.* The goal is to generate geometric models representing a continuous description of the surfaces inherited by the considered point cloud.

In addition to the vast amount of point samples scientific and geometric data sets are composed of, several other issues complicate these operations. Point data sets obtained from terrestrial and airborne LiDaR scan devices typically contain diverse and highly complicated objects featuring topological ambiguities making e.g, surface reconstruction a sophisticating task. Since the measurements producing these data sets are taken from great distance (predominately in the case of environmental LiDaR scans) measurement errors are induced increasing the noise level. Occlusion, semi-transparent and reflecting objects, structures with small fractured and under-sampled surface components like trees, as well as scanning error due to motion of cars, humans, animals, etc. also cause

considerable noise. When performing the above operations the noise level has great impact on the quality of the results.

1.3 Contribution of Dissertation

The fundamental contributions of this dissertation are:

- It provides a new framework [64] regarding the acquisition of point data from 2D images sequences as produced by video camera systems (cf. Section 2.3). The reconstruction includes the matching of point correspondences between consecutive image pairs, the recovering of the unknown intrinsic and extrinsic camera parameters and the actual point reconstruction i.e., the computation of the 3D coordinates from corresponding image point pairs. The proposed framework relies on a maximum likelihood (ML) based camera calibration scheme. In this context an analysis of the reconstruction regarding intensity- and feature-based correspondence matching schemes is provided.

- The dissertation introduces an approach for characterising the points of given point clouds in a way that they facilitate extended visual exploration [83]. The proposed method filters and colors the point cloud based on a probability-like classification of local point neighborhoods into curve-like, surface-like, and solid-like shapes. In context with the applied data structures this approach allows to perform a detailed analysis of the point data at its full resolution. It also supports developing new feature extraction and surface reconstruction algorithms (see Section 3 and Section 4) overcoming the major problems associated with processing complex and noisy unstructured point clouds.

- Based on the introduced point classification scheme a new approach for probability-based feature extraction [83] (see section 3) is discussed. The feature values associated with the individual points are used to perform a stable and reliable extraction of feature lines matching user-specified conditions. The feature graphs generated in this context can be used to describe the structural composition of the input point cloud and can be visualized together with the original data to guide the user when exploring point data sets. Unlike other approaches this method is suitable

to be applied to complex, and noisy environmental point clouds [1] featuring high complexity.

- A novel method allowing adaptive surface reconstruction from point clouds [81] [80] having unknown topology is presented. The introduced algorithm is able to efficiently provide surface representations reflecting the topology of the input point cloud at multiple resolutions. The method, in particular, is qualified to perform surface reconstruction from noisy point clouds having moderate complexity.

- Following the idea of the former method a novel feature-based approach [82] concerning surface reconstruction from high complex and noisy (environmental) LiDaR point clouds is presented. The previously introduced point classification scheme is applied to construct a continuous implicit function quantifying whether the underlying point set locally represents a surface or not. A surface then is extracted with a variant of marching cubes (MC) which is transformed into a final non-manifold representation. The user can control the surface topology and thus can adapt the reconstruction to the conditions (e.g., density, noise, etc.) of the point samples. This reconstruction method is able to provide surface representations from high-grade noisy and unstructured point data.

1.4 Motivating Applications

There are several areas in science and engineering the methods introduced by this dissertation may find possible application.

1.4.1 Applications in Geoscience

In geology and seismology a combination of terrestrial/aircraft-based LiDaR scanners and GPS systems have evolved to an important tool for studying landscapes. The output of the technologies can produce accurate digital elevation models. The techniques produces new insights into geologic features by providing high-resolution 3D measurements of landscapes e.g., for studying slip on faults, erosion, and other landscape-altering processes. Each LiDaR scan produces a large number of individual points associated with

[1] Throughout this dissertation environmental point clouds always refer to point clouds describing environmental scenery's.

the reflection of the laser from natural and engineered surfaces. Currently, there are few methods that can analyze the data at its full resolution. For geologists, such methods are of special interest since they facilitate the detection and visualization of features like creases in structures caused by earthquakes, as well as the rapid and automatic identification of key features in environmental data such as ridge lines, stream beds, and edges of terraces. The existence of reliable methods would allow to perform (time-dependent) virtual-based measurements e.g., in order to track the movement of landmasses as caused by earthquakes and flooding.

1.4.2 Applications in Reverse Engineering

Reverse engineering is the process of reconstructing computer-based models from physical objects. Such models are indispensable in several fields of science and engineering. For example, the design and manufacturing of vehicles, such as cars and aircrafts, would not be possible without sophisticated CAD and simulation tools predicting the behavior of the product. On the other hand, designers often do not like working on virtual models, though sophisticated tools, like immersive VR-environments are available. Hence, they often produce physical prototypes made from materials that can be easily assembled and shaped like clay models. Surface reconstruction algorithms support the rapid prototyping by allowing the creation and remodeling of virtual copies of existing manufacturing parts in a very efficient and accurate way. Applications in medical areas also benefit from those methods since they enable generating precise models of the inside of the human body from point data produced by CT- and MRI-scans.

1.4.3 Applications in Creating and Populating Virtual Environments

In some application areas, having accurate models of real world objects and environments is very important. Especially, when considering fields like urban planning, computer-based models of landscapes, buildings and construction areas become more and more relevant. In addition, such models are demanded in the context of simulating environmental disaster and pollution scenery's helping to develop appropriate counteractions. The relevance also holds for other application areas requiring virtual copies of real world objects e.g., digital libraries like the Digital Michelangelo Project [1]. Another application area is the movie industry which has the permanent need to construct photo-realistic

models for producing the next blockbuster. Creating such virtual models using common CAD tools is a sophisticating and time consuming task. Scanning and rebuilding environments and objects e.g., buildings, landscapes or exhibits as virtual representations is a topic attracting recent attention.

1.5 Overview of Dissertation

The dissertation is structured as follows: In Section 2 we discuss the principles of the acquisition and organization of point clouds. The most commonly deployed scan systems (see Section 2.2) are introduced and a framework for the reconstruction of point clouds from 2D images (see Section 2.3) is presented. In this context we discuss some of the properties the produced point data sets typically exhibit and how they interfere with further processing (see Section 2.4). Section 3 is concerned with the method for analyzing the structural composition of point clouds. We discuss the approach for probability-based point classification and address the method for extracting curve-like features from environmental point data sets. Surface reconstruction is subject to Section 4. Section 4.4 discusses the multi-resolution surface reconstruction approach, whereas the volumetric feature-based reconstruction method is introduced in Section 4.5. The Summary (Section 5) concludes the dissertation and provides ideas concerning potential directions for future research.

2 Point Sets

2.1 Overview

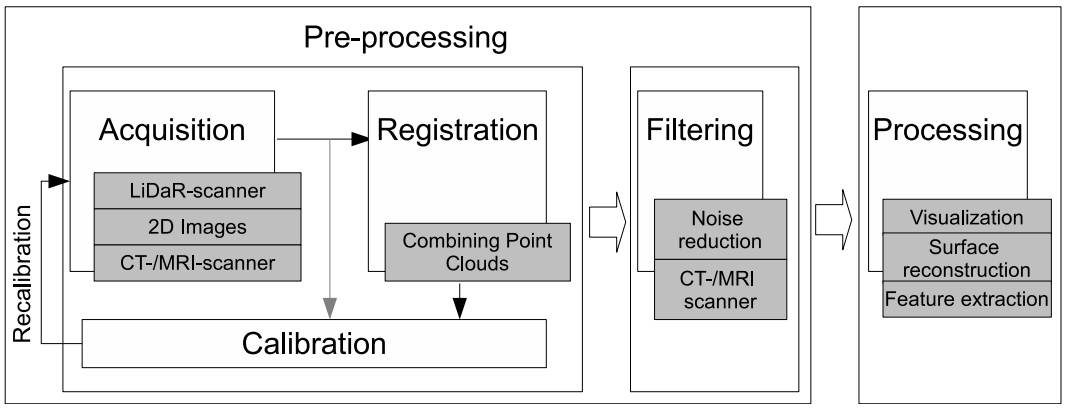


Figure 2.1: Diagram depicting the individual steps concerning the workflow from data acquisition up to the final data representation.

Point data sets are used in several areas of science and engineering to measure physical properties like the shape, the temperature or the density of objects. In most cases these properties represent or are associated with coordinates in 3D space, primarily measuring geometric characteristics of objects e.g., engine parts, environmental structures or parts of the human body. One of the goals of this dissertation is to recover "hidden" features like surfaces or curve-like structures from unorganized point data sets. This demands providing suitable methods for transforming structural information captured by the point cloud into according representations. The overall workflow from data acquisition toward the final representation includes several steps (see Figure 2.1):

- *Acquisition:* The capturing of the initial point data from physical objects and environments by using special scan devices, or from other data (e.g. volumetric flow simulation data, 2D stereo images) by using particular reconstruction methods.

Depending on the scan method several data sets may be generated representing different views of a scene.

- *Registration:* The definition of a reference coordinate system and the transformation of all corresponding point data into this system.
- *Calibration:* The application of methods for improving the quality of measurements (e.g., recalibration of scan devices and repetitive execution of the scan processes).
- *Filtering:* Concerns the extraction of points or groups of points matching certain conditions. For example, this includes the reduction of noise, outliers or redundant points.
- *Processing:* The application of methods for visualization and the generation of feasible data representations. Amongst others this includes algorithms for point-based rendering, surface reconstruction and feature extraction.

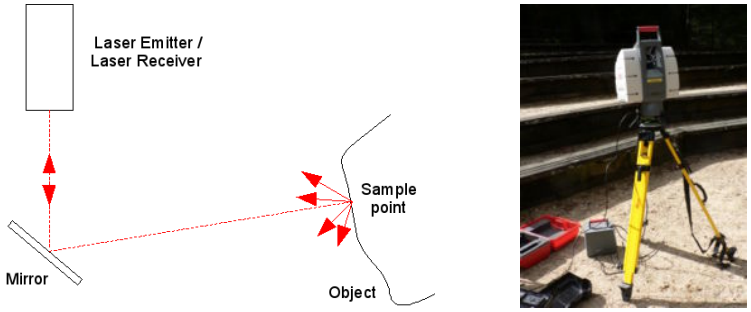
Discussing the methods developed for extracting structural features like surfaces and curves requires reviewing basic aspects of capturing and organizing point data sets. This chapter gives a general overview about the devices used to gain point clouds, predominantly LiDaR scan devices and CT-/MRI scanners. But it also introduces a method that allows the reconstruction of point data from 2D image series. As already noted in the first chapter handling those point clouds is challenging because of their principle nature. Thus, a brief discussion of some important properties of point clouds highlighting the principle difficulties associated with processing unorganized point data is performed. In order to accomplish efficient processing, concepts for organizing point data are required which are introduced in the last part of this chapter.

2.2 Data Acquisition

The term data acquisition refers to the processes responsible for sampling real world objects in order to generate virtual representations. Typically, this incorporates the acquisition of signals and impulse responses that can be transformed to 3D coordinates. The components of the acquisition devices include sensors that convert any measurement parameter to an electrical signal, which then is conditioning and acquired by special data acquisition hardware. There exist a variety of different acquisition systems including

laser-based, sonar-based and photometry devices. In the following we examine some of them.

2.2.1 LiDaR Scanners



(a) (b)

Figure 2.2: (a) Principle functionality of LiDaR scan devices, (b) image [1] of a mobile LiDaR scanner.

LiDaR (Light Detection and Ranging) scan devices represent an optical-based remote sensing technology capable of measuring the distance to objects by laser impulses. Sometimes this acquisition technique is also referred to as laser altimetry. Similar to the radar technology, which uses radio waves instead of light, the range to an object is determined by measuring the time delay between the transmission of a pulse and the detection of the reflected signal (see Figure 2.2a). This type of scanner typically consists of a laser-unit emitting the laser pulse and a CCD-sensor receiving it. The time the laser beam needs from the emitter to the receiver is measured and converted into a distance parameter. These types of scanners are either used stationary (e.g., in production lines for quality control), as mobile scan systems mounted on tripods for close and mid-range terrestrial scanning (e.g., to capture buildings, landscapes etc.), or as airborne system for performing large scale topographic scans.

Stationary 3D scanners are primarily used in cases where small objects have to be measured. These devices either are fix where the object is shifted, in most cases rotated, or are movable (hand-held) whereas the object is kept fix.

[1] Image source: `http://en.wikipedia.org/`

Terrestrial-, tripod-based scan systems are commonly used for scanning buildings, landscapes, etc. Mostly, these systems are micro-pulse LiDaR (low power) systems that allow environmental scans from multiple viewpoints. In general these systems produce huge amounts of data. They are important in the area of civil engineering, for example, to detect environmental changes or to guide the construction of buildings, roads, or bridges. Figure 2.2b shows an image of a tripod mounted LiDaR scanner.

Airborne laser altimetry uses high energy laser systems mounted on airplanes in combination with GPS devices to sense the topography of landscapes including trees as well as buildings. Despite of the great distances the resulting measurements are relatively precise. The possibility to combine airborne and terrestrial LiDaR data makes this technique indispensable in the areas of geoscience and 3D mapping. In combination with terrestrial scan systems it is possible to adapt the scan resolution to provide a better insight into areas of special interest. The produced point data sets easily exceed the limit of 100 million points.

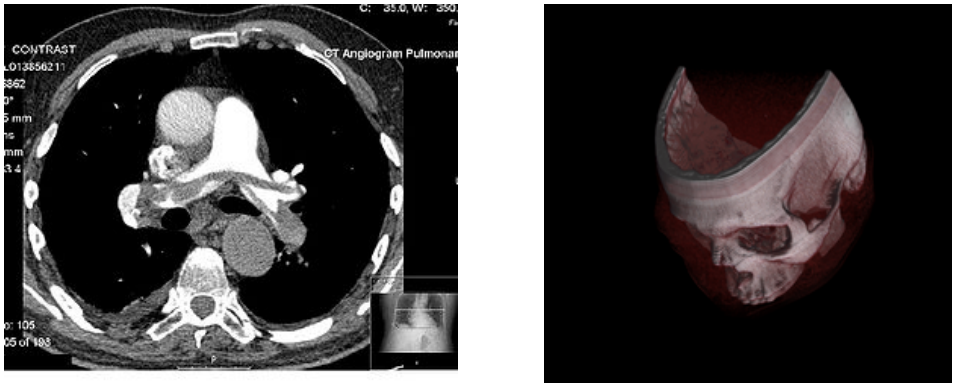
Satellite laser ranging (SLR) systems are mainly used to provide digital elevation maps and allow the exploration of unknown or non-accessible terrains or are used for measuring volumetric changes in continental ice-masses. The data sets produced, too, may be immense and critical to store.

2.2.2 Radar Scanners

Radar scanners on the other hand follow a similar approach. Instead of using light waves, electro-magnetic waves are used to sense conditions of remote objects. In most cases radar antennas or dishes transmit short radio impulses which are reflected from the surface of the sensed object. Distance measurements, again, are performed by measuring the time the signal needs to return. Compared to light waves radio waves have a much shorter wavelength allowing the penetration of certain materials. This provides the possibility to measure properties like the density or the mass of objects. The class of these scan instruments comprise stationary as well as movable devices. The applications, for example, lie in the field of terrain surveying for generating digital elevation maps (DEM), where it is possible to provide additional information regarding the mineralogical composition of terrains. These systems are also applied in the field of

meteorology to perform observations on cloud density and rainfall to provide weather forecasts. Combining the principles of LiDaR and Radar in form of laser radar scanners enables to improve measurements and applicability.

2.2.3 CT/MRI Scanners



(a) (b)

Figure 2.3: (a) image slice [1] obtained from a CT-scan of the human body, (b) example of a virtual skull representation [1] reconstructed from a CT-scan.

Computer tomography (CT) and *magnet resonance imaging* (MRI) devices are scan instruments allowing to generate images e.g., from the inside of the human body. The scans mainly produce image slices showing structures in form of density values (see Figure 2.3a). Assembling these data images yields a volumetric representation which when using special visualization methods, e.g. methods for volume rendering (including volume ray casting, volume ray tracing), provide a keen insight of interior of objects. Figure 2.3b shows an example of such an assembled, volumetric data set. However, in order to draw further advantage from these data sets, for example, extracting structures for modeling purposes, this kind of representation is unfavorable. By filtering the data regarding a predefined target density value it is possible to extract certain structures, e.g. in form of point clouds or connected surfaces (iso-surface extraction). The obtained structures then can be used for further processing.

[1] Image source: `http://en.wikipedia.org/`

2.3 Reconstruction of Point Clouds from 2D Images

Another possibility to obtain point data in form of point clouds is provided by the reconstruction from 2D stereo image pairs. In the following we review the basic concepts and discuss a new framework [64] which enables the reconstruction of point clouds from a series of corresponding images e.g., as provided by video camera systems.

2.3.1 Overview

Recovering the geometry of a scene from multiple images taken from different viewpoints is one of the oldest problems in computer graphics. The goal is to reconstruct point clouds of objects from a collection of images taken from different viewpoints. Over the last few years a great number of algorithms have been developed, and the state of art is improving rapidly.

Concerning the reconstruction process we distinguish between the case in which camera parameters are known and the case the camera parameters are unknown. In this section we primarily concentrate on the second case in which the camera system is uncalibrated. To be able to reconstruct geometric information we first have to recover the camera parameter like focal length, position and orientation. This requires to detect corresponding images points between the individual image pairs. The accuracy of the obtained parameters in particular depend on the applied correspondence matching scheme. Once the parameters are known we are able to project image points back to its corresponding positions in 3D space. Summarized, the following contributions are provided:

- A framework for reconstructing point clouds from a sequence of corresponding image pairs based on a *maximum likelihood* (ML) camera calibration scheme is introduced. The proposed method relies on intensity and feature-based correspondence matching schemes.

- To be able to perform the reconstruction of point clouds one has to identify correspondences between image points of subsequent images. Several methods already exist to accomplish this task. Within this section we review some fundamental methods and perform a discussion concerning their suitability with respect to point reconstruction.

The section is structured as follows: Section 2.3.2 and Section 2.3.3 discuss some fundamental concepts concerning the basic projection model and the geometry of stereo projection also known as *epipolar geometry*. In Section 2.3.4 we review some related work. Section 2.3.5 presents the basic stages regarding the reconstruction of point clouds from uncalibrated image pairs. In Section 2.3.6 we review some of the applied approaches regarding the correspondence matching process followed by the introduction of the ML based parameter identification, see Section 2.3.7. Section 2.3.8 subjects the actual reconstruction of the 3D coordinates from image points. The last part 2.3.9 provides and discusses the results followed by the conclusions.

2.3.2 The Projection Model

This section introduces the basic projection model used to describe the processes within a camera. All subsequent considerations are based on the so called *pinhole camera model* which generalizes the physical projection process and allows to define a mathematical model.

Static Projection

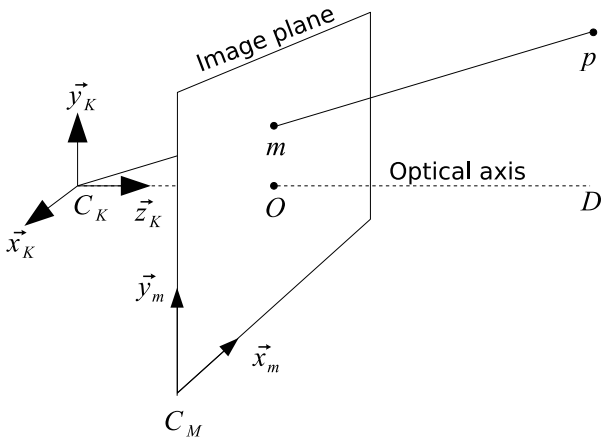


Figure 2.4: Simplified camera model illustrating the projection of a point p onto an image plane M.

Informally, the projection occurs as follows: Light rays are reflected from objects in front of the camera and are projected through the focal projection center located at the

camera aperture onto an image plane at the inside of the camera; Instead of positioning the image plane behind the camera aperture we move the plane in front of the projection center. This does not violate the projection model but neutralizes the mirror effect causing the reverse projection. Figure 2.4 illustrates the simplified model. Here C_K represents the focal projection center that now simultaneously represents the origin of the camera coordinate system. The optical axis defined by the $\vec{z}_C$-axis, intersects the image plane M at the point O. The origin of M is located at C_M. The projection of a point p onto M delivers the image point m. The image coordinates of a point $p = (x, y, z)^T$ projected onto M are computed by

$$m = (\Delta u + f\frac{x}{z}, \Delta v + f\frac{y}{z}), \tag{2.1}$$

where $(\Delta u, \Delta v)^T$ represent the coordinates of O in image space. In matrix notation the projection of a point p can be written as

$$m = K\ p, \tag{2.2}$$

where $m = (u, v, 1)^T$ now denotes the image coordinates of the projection of point p in homogeneous coordinates and

$$K = \begin{pmatrix} fk_u & -fk_u \cot(\Theta) & \Delta u \\ 0 & \frac{fk_v}{\sin(\Theta)} & \Delta v \\ 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

represents the standard projection matrix. Here, f denotes the focal length, k_u and k_v define the number of pixel per millimeter in the direction of the image axis $\vec{x}_M$ and $\vec{y}_M$. The parameter Δu and Δu represent the offset of the center point O from the origin of the image plane C_M. The skew angle Θ, describing a possible shear of the image plane, mostly equals 90 degrees. Thus, the calibration matrix simplifies to

$$K = \begin{pmatrix} \alpha_u & 0 & \Delta u \\ 0 & \alpha_v & \Delta v \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.4}$$

where α_u and α_v substitutes fk_u and fk_v. In this context, the four camera parameter focal length f, aspect ratio $\frac{\alpha_x}{\alpha_y}$ as well as Δu and Δv are the so called *intrinsic camera parameters.*

Moving the Camera

The previous considerations assume that the camera coordinate system is tied to a given reference coordinate system i.e., the projection center of the camera and the origin of the reference coordinate system as well as the principal coordinate axis of both systems are equal. However, in general the camera needs to be arbitrarily positioned and oriented in order to capture the scenery from different viewpoints. The entire projection procedure is defined by incorporating additional matrices R and vectors t responsible for rotating and translating the scenery. This is equivalent with the rotation and translation of the camera system by R^{-1} and t^{-1}. Assuming that the camera coordinate system is tied to a reference coordinate system, the transformation of a point $p_r = (x, y, z, 1)^T$ (in homogeneous coordinates) is realized by

$$p = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} p_r, \tag{2.5}$$

where R represents the 3×3 rotation matrix and $t = (t_x, t_y, t_z)^T$ denotes the translation vector. The resulting point in camera coordinates is given by p. Here, the camera parameters used to set up the rotational matrix R defined by the three principle Euler angles and the coordinates of the translation vector t are called *extrinsic camera parameters.*

Combining the projection (2.2) and transformation (2.5) leads to the following linear system

$$m = K\,[I|0] \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} p_r, \tag{2.6}$$

where I denotes the identity matrix. The term 2.6 can be simplified to

$$m = K\,[R|t]\,p_r. \tag{2.7}$$

The general projection matrix for projecting a point p_r onto M, delivering m, then has the form

$$P = K\,[R|t]\,. \tag{2.8}$$

The reconstruction of 3D point clouds from 2D images aims at recovering depth information for each point in the image plane. This makes it necessary to estimate the missing intrinsic and extrinsic camera parameters. In the next section we review the principle geometric interrelationships between points in 3D reference space and a pair of corresponding image points located on virtual projection planes.

2.3.3 Epipolar Geometry

Instead of considering the geometry of a single camera system we now consider the geometry of a pair of cameras at once. The relations between two distinct camera systems A and B located at different positions within a reference coordinate system define the so called *epipolar geometry*. Let M and M' be the two corresponding virtual image planes whose projection centers are located at C_K and C'_K (cf. Figure 2.5). Given a point p_r within the reference system that is projected onto both image planes, along with the corresponding image points m and m' the projection implies a number of geometric constraints also called *epipolar constraints*:

The connection between both projection centers is called the *epipolar base line*, the

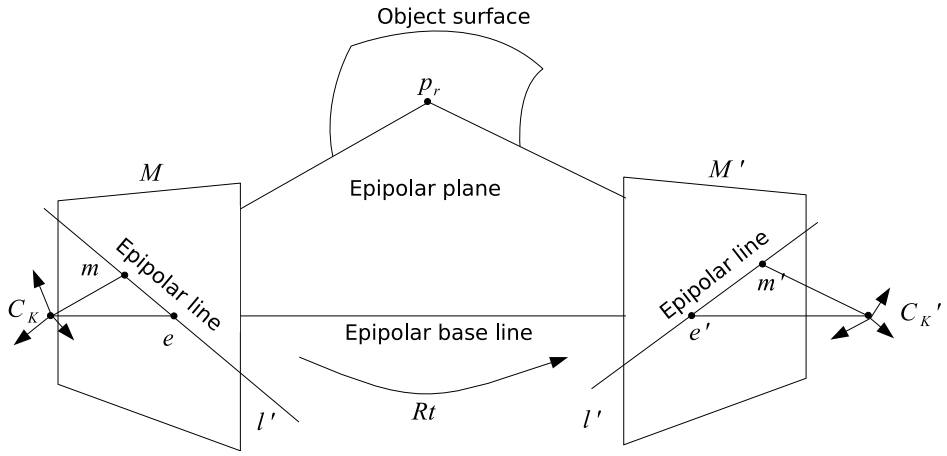


Figure 2.5: Illustration of the geometric constraints concerning the projection of a point onto two corresponding virtual image planes.

corresponding intersection points e and e' on the planes M and M' are the *epipoles*. The plane spanned by C_K, C'_K and p_r represents the *epipolar plane* on which the points p, m and m' are located on. The straight lines l and l' resulting from the intersection between the epipolar plane and the image planes M and M' are the so called *epipolar lines*. Each epipolar line intersects its corresponding epipole. Given an image point $m = (u, v, 1)^T$ (in homogeneous coordinates), located on l, the corresponding image point $m' = (u', v', 1)^T$ in M' is always located on the opposing epipolar line l', and vice versa. This constraint is expressed by the equation

$$m'^{\,T} F m = 0, \tag{2.9}$$

where F denotes the 3×3 *fundamental matrix*, m and m' are the corresponding image points in M and M'; The fundamental matrix encapsulates the intrinsic as well as the extrinsic camera parameters and hence describes the epipolar constraints between an uncalibrated stereo image pair. In principle, F maps the image points to their corresponding epipolar lines. In the context of reconstructing point clouds from a sequence of "uncalibrated" 2D images the fundamental matrix has to be identified first. The process of how the entries of F are recovered is discussed in the next Section 2.3.5.

Now, let us assume the two camera systems are calibrated i.e., the coordinates of the camera systems C_K coincides with the reference coordinate system and M and M' are

related by a rotation R and a translation t (cf. Figure 2.5). In this case the epipolar constraint is expressed by

$$p'^{\,T} E p = 0, \tag{2.10}$$

with E representing the 3×3 *essential matrix* of rank 2, and $p = (x, y, z)^T$ and $p' = (x', y', z')^T$ are the related points on the epipolar lines l and l' represented in coordinates of the reference system; The essential matrix is defined by

$$E = [t] \times R, \tag{2.11}$$

where $\times$ denotes the cross product operator. The essential matrix can also be rewritten as

$$E = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} R, \tag{2.12}$$

with $t = (t_x, t_y, t_z)^T$. In case the camera system is calibrated, the relationship between the fundamental and the essential matrix is expressed by

$$F = K^T E K, \tag{2.13}$$

with K representing the camera calibration matrix defined by (2.4). For a more detailed inside into the field of epipolar geometry we refer to the work of Hartley and Zisserman [66].

2.3.4 Related Work

The problem of reconstructing geometric shapes from multi-view stereo images has received a lot of attention over the last years, yielding a great variety of reconstruction algorithms. Hence, providing a complete review about the stereo reconstruction area is out of the scope of this work. This section rather restricts to provide a brief overview of some of the most important methods. However, instead of reconstructing point clouds most algorithms primarily aim at the immediate construction of continuous surface representations.

Volumetric based reconstruction approaches [130] [51] [42] [89] [88] [136] [25] in general assume that there is a known bounded area in which the objects of interest are located. One possibility to represent this volume is by using a regular tessellation of cubes, also called *voxels*, as it is provided by e.g., Octrees (see Section 2.5.3). In case, the input cameras are not calibrated and the fundamental matrices are known for the cameras pairs, a projective space representation can be defined by performing a so called projective space voxelization [100] [128]. These methods remove voxel from the scene if they are not photo-consistent. In contrast, ray space representations [87] [103] directly operate on the rays from the camera's optical centers, avoiding volumetric discretization issues. A survey of those methods is provided by [135] and [140].

Another approach concerns the reconstruction and merging of depth maps that allow the computation of surfaces representations or point clouds. Such methods typically optimize an image based smoothness term [139] [78] [85] [53] [111] that seeks to give neighboring pixels the same depth values. Other approaches [142] [44] [107] [101] perform the reconstruction by first identifying a set of feature points within consecutive images, most likely are part of underlying surfaces, and then extract the final surface representation based on the set of the reconstructed feature points.

In contrast to immediately reconstruct surface representations from uncalibrated multi-view image pairs, this part of the dissertation especially aims at recovering accurate point samples. The proposed method thereby relies on a ML based auto-calibration scheme allowing to perform a precise reconstruction. Instead of restricting the reconstruction to an assorted set of feature points, probably defining a surface, the proposed framework is intended to provide a great amount of accurate point samples that can be used to perform an independent surface reconstruction e.g., by using the methods introduced in Chapter 4.

2.3.5 Point Reconstruction

The reconstruction of geometric information in form of 3D coordinates from a series of corresponding 2D images is a challenging task. Especially, since essential information like the exact location and orientation of the camera at the time at which the corresponding images were taken is unknown. In principle, the overall problem of reconstructing points from a series of consecutive images can be reduced to the case of reconstructing points from stereo images. To accomplish this task the following steps have to be performed:

1. First of all we have to detect correspondences between the consecutive image pairs in form of corresponding image points. This is accomplished by detecting a so called *displacement vector field* which warps the first image into the subsequent one. Concerning the quality of the final reconstruction this is one of the most crucial steps of the entire process. In Section 2.3.6 we review some of the most common methods applied to solve this task.

2. Provided that, reliable point correspondences have been identified we proceed recovering the unknown camera parameters for each consecutive image in form of projection matrices. We discuss the principle course by using a maximum likelihood based parameter estimation scheme (see Section 2.3.7).

3. The last step consists of the actual point reconstruction. Assumed the projection matrices are recovered successfully we need to perform a correspondence matching concerning those image points for which the 3D coordinates have to be determined. The 2D image points are then reprojected to their corresponding locations in 3D space (see Section 2.3.8).

In the following sections we discuss the above steps in detail.

2.3.6 Resolving Image Correspondences

Given a sequence of 2D images, the goal of recovering image correspondences between two consecutive images M and M' consists of identifying a displacement vector field V, also termed *flow field*, which transforms M into M'. The correspondences are represented in form of image point pairs (m, m'), with $m' = m + d$, with $m \in M$ and $m' \in M'$, where $d \in V$ is the displacement vector of m.

A great variety of methods have been proposed to solve this problem [75] [98] [27] [113] [148] [157]. However, not each method is applicable to each type of image sequence. Depending on the way the focused scene is composed of and the pixel intensity of a image is distributed, the right method has to be chosen. The following part is intended to provide the reader with a brief overview of the methods applied in this work. Primarily, we address the fundamental intensity- and feature-based approaches. For more detailed information concerning the individual methods we refer to the according works.

Intensity-Based Correspondence Matching

Most of the introduced intensity-based methods rely on the general assumption that the intensity of corresponding points do not vary (considerably) between subsequent images. Let us assume an image sequence continuously defined over time, where $M(m,t)$ denotes the intensity of the point $m = (u,v)^T$ of the image M at time t, then the above consideration is expressed by

$$\frac{\partial M}{\partial u}\frac{du}{dt} + \frac{\partial M}{\partial v}\frac{dv}{dt} + \frac{\partial M}{\partial t} = 0, \tag{2.14}$$

where $\frac{du}{dt} = d_u$ and $\frac{dv}{dt} = d_v$ define the displacement vector $d = (d_u, d_v)^T$. Equation (2.14) can be rewritten as

$$M_u d_u + M_v d_v + M_t = 0, \tag{2.15}$$

where M_u, M_v and M_t denote the partial derivatives of $M(m,t)$ according u, v and t. To compute the displacement vector d at coordinates m one has to minimize the following squared residual regarding d_u and d_v

$$\varepsilon^2 = (M_u d_u + M_v d_v + M_t)^2, \tag{2.16}$$

The coordinates of the corresponding image point in the subsequent image then is defined by $m' = m + d$.

The intensity-based method of Horn and Schunk [75] attempts to determine the optical

flow between two images in form of a smooth vector field by incorporating additional smoothness terms

$$\varphi^2 = (\nabla d_u)^2 + (\nabla d_v)^2. \tag{2.17}$$

The overall minimization problem now evolves to

$$\omega = \varepsilon^2 + \alpha\varphi^2 \tag{2.18}$$

where α represents a parameter controlling the smoothness of the resulting displacement vector field.

Another method proposed by Lucas and Kanade [98] is based on the assumption that proximate image points correspond to the same underlying object and thus experience the same optical flow. To detect the optical flow for a point m all points lying within a predefined neighborhood are incorporated for solving equation (2.15). The overdetermined system has the form

$$\begin{pmatrix} M_u(m_0) & M_v(m_0) \\ \vdots & \vdots \\ M_u(m_n) & M_v(m_n) \end{pmatrix} \begin{pmatrix} d_u \\ d_v \end{pmatrix} = \begin{pmatrix} M_t(m_0) \\ \vdots \\ M_t(m_n) \end{pmatrix}, \tag{2.19}$$

where m_i represent the n neighbor points (including point m) of image M at time t. The linear system is solved by using least squares minimization (cf. Appendix B). The method is confined to be applicable to points whose environment corresponds to image areas having non-constant intensity. Depending on the scene focused, this does not always hold. To measure the quality of point correspondences we simply compare the intensity of the corresponding image points. This way we are able to discard "bad" correspondences whose quality fall below a given threshold.

The above discussed approaches in general provide good results. However, in cases the point intensity varies considerably or is constant over a wide region the correspondence detection can fail. In that case, the following methods may be capable of producing better results.

Feature-Based Correspondence Matching

In contrast to intensity-based approaches feature-based correspondence detection, in general, does not provide a dense displacement vector field. Instead, displacement vectors are computed only for image points associated with prominent features. This includes points or regions that differ considerably from their immediate neighborhood. At best, the identified points are invariant according image transformations like rotation and translation.

A well-known approach for detecting prominent features is proposed by Harris and Stephens [65]. The method concentrates on identifying edges and corners within 2D images. In this context, edges and corners are associated with image points at which the intensity varies significantly in either one or two principal directions. Given a point $m = (u, v)^T$ within image M, the applied auto-correlation function measures local variations between image patches shifted in different directions by $(\delta u, \delta v)$. The corresponding equation has the form

$$S(u, v) = \sum_{W} \omega(u_i, v_i) \left(M(u_i, v_i) - M(u_i + \delta u, v_i + \delta v) \right)^2 , \tag{2.20}$$

where ω represents a (Gaussian-based) weighting function, u_i and v_i represent the coordinates of the image points within the pre-specified integration neighborhood W centered at (u, v). This expression can be rewritten as

$$S(u, v) \approx (\delta u, \delta v)\, A(u, v) \begin{pmatrix} \delta u \\ \delta v \end{pmatrix} , \tag{2.21}$$

where

$$A(u, v) = \sum_{W} \omega(u_i, v_i) \begin{pmatrix} M_u^2 & M_u M_v \\ M_u M_v & M_v^2 \end{pmatrix} , \tag{2.22}$$

with M_u and M_v defining the derivations at position (u, v) along the principal image directions. The matrix A captures the intensity structure of the local neighborhood. A point m is then characterized by comparing the eigenvalues λ_1 and λ_2 of A. This

allows to distinguish between three cases: (a) both λ_1 and λ_2 are relatively small. In this case the local neighborhood has constant intensity along the principle directions and thus does not exhibit prominent features. (b) either λ_1 or λ_2 is high, whereas the other eigenvalue is low. This configuration indicates the presence of an edge. (c) both eigenvalues are high. The auto-correlation function is sharply peaked which indicates an existing corner. This method has experienced some improvements over the recent years e.g., see [133].

Once the prominent features within two given images M and M' are identified, one can track the features by detecting correspondences based on the comparison of intensity and distance properties between points of consecutive images. However, in some cases the found correspondences are not unique. A common strategy to improve the matching is provided by applying a combined forward-/backward matching scheme. The idea is the following: Given an image point m in the first image, the task is to identify the corresponding point m' in the subsequent one. Backwards, given the point m' the matching should now identify m as correspondence. If this is not the case, m and m' should not be linked as corresponding point pair.

Though, the regular tracking in general provides a reliable number of matchings it still may occur that due to inaccuracies more or less correspondences are not detected. Another interesting method for detecting the optical flow for a set of identified feature points was presented by Shi and Tomasi [134]. They propose to extend the regular approach of determining a displacement vector of a point by incorporating an transformation that modifies the points within a local predefined integration environment. This allows to more accurately extract the displacement of an initial set of feature points over a greater number of consecutive images. Given an identified feature point $m = (u, v)^T$, the final displacement r of m is defined by

$$r = Dm + d, \tag{2.23}$$

where d denotes the regular displacement vector and

$$D = \begin{pmatrix} d_u d_u & d_u d_v \\ d_u d_v & d_v d_v \end{pmatrix} \tag{2.24}$$

represents the deformation matrix which manipulates the displacement d, with $d_u = \frac{\partial d}{\partial u}$ and $d_v = \frac{\partial d}{\partial v}$. The solution for r is now obtained by solving the following minimization problem according the entries of D and d, defined by

$$\varepsilon = \int_W \omega(u,v) \left(M((I+D)m+d) - M(u,v)\right)^2, \tag{2.25}$$

where W denotes the chosen integration environment, I is the identity matrix and $\omega(u,v)$ represents a Gaussian weighting function. To obtain reliable results a properly sized integration domain W needs to be chosen.

A principle problem associated with all of the above introduced approaches is induced due to the varying conditions at the image boundaries. The camera movement in a sense effects the image points to march across the image boundaries. As a consequence, we lose track of some of those points close to the boundary domain, simultaneously, new image points are introduced elsewhere. In general, this causes the correspondence matching to fail at the affected regions. One way to deal with this fundamental problem is to confine the matching process to a restricted region within the images.

The introduced methods are well established in the field of image-based correspondence matching and implementations are provided by several libraries. In this work the OpenCV library [2] was used to perform the extraction of the flow fields from a series of 2D images.

2.3.7 Maximum-Likelihood Based Parameter Reconstruction

Once the correspondences between image pairs are identified we proceed determining the unknown camera parameters. This process is also known as *auto-calibration*. Given an unknown point $p_r = (x,y,z,1)^T$ in 3D reference space (represented in homogeneous coordinates) and the two corresponding points $m = (u,v,1)^T$ and $m' = (u',v',1)^T$ (also in homogeneous coordinates) within the images M and M'. By assuming the first camera system is tied to the reference coordinate system, m and m' are related by two unknown projection matrices P_1 and P_2:

$$m = [K_1|0]\, p_r = P_1 p_r, \tag{2.26}$$

$$m' = [K_2 R| - K_2 Rt]\, p_r = P_2 p_r, \tag{2.27}$$

where R represents the rotational matrix and $t = (t_x, t_y, t_z)^T$ the translation vector describing the relation between the first and the second camera system. K_1 and K_2 are the two intrinsic camera calibration matrices. Here, $[K_1|0]$ and $[K_2R| - K_2Rt]$ are 3×4 matrices divided into a 3×3 block and a 3×1 column. Now, the goal is to recover the matrices R, K_1 and K_2 as well as vector t by exploiting coherence's stated by the epipolar geometry. As discussed in Section 2.3.3 the image point m' in M' corresponding to m in M must lie on a certain epipolar line determined by two points: The first point is defined by the center of the first camera system projected onto image plane M' - the corresponding epipole e' - defined by $[K_2Rt]\,(0,0,0,1)^T$. The second point represents the projection of the point at infinity

$$\begin{pmatrix} K_1^{-1} m \\ 0 \end{pmatrix}$$

onto M'; The corresponding epipolar line, intersecting m' then is defined by

$$K_2 Rt \times K_2 R K_1^{-1} p_r. \tag{2.28}$$

As shown in [68], term (2.28) is used to derive the corresponding fundamental matrix

$$F = (K_2^{-1})^t R S K_1^{-1}, \tag{2.29}$$

with

$$S = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix}, \tag{2.30}$$

By assuming the image offset $(\Delta u, \Delta v)$ is zero and the focal length is known it can be shown (see [68]) that the fundamental matrix F, consisting of 9 matrix entries, has 7 degrees of freedom (the aspect ratio, the translational coordinates, and the three principal rotational angles).

Least Squares Based Solution

To solve F the linear constraint (2.9) can be rewritten as

$$\begin{pmatrix} u'_0u_0 & v'_0u_0 & u'_0 & v'_0u_0 & v'_0v_0 & v'_0 & u_0 & v_0 & 1 \\ u'_1u_1 & v'_1u_1 & u'_1 & v'_1u_1 & v'_1v_1 & v'_1 & u_1 & v_1 & 2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u'_Nu_N & v'_Nu_N & u'_N & v'_Nu_N & v'_Nv_N & v'_N & u_N & v_N & N \end{pmatrix} \begin{pmatrix} f_{00} \\ f_{01} \\ f_{02} \\ \vdots \\ f_{22} \end{pmatrix} = 0. \tag{2.31}$$

Here, $m_i = (u_i, v_i)^T$ and $m'_i = (u'_i, v'_i)^T$ denote the n identified corresponding point pairs of the images M and M'. We abbreviate equation (2.31) by $A\ vec(F) = 0$. Since F only can be determined up to a scale factor, a number of 8 corresponding point pairs and one additional scaling constraint are necessary to solve the above equation. However, since the point correspondences may be afflicted with errors, using a greater number of correspondences probably lead to more accurate results. In this case, the system is overdetermined. Then, applying least squares minimization to AA^T delivers the best approximation for F.

In case the intrinsic camera parameters are known i.e., the corresponding camera system is calibrated, the problem of recovering the missing parameters only consists in determining the entries of R and S of the term (2.29). A possible approach for solving this problem is presented by Longuet-Higgins [96]. He aims at factorizing F into R and S with known matrices K_1 and K_2. However, since we are considering image sequences taken from multiple locations within an uncalibrated environment the intrinsic camera parameters are unknown. Thus, the problem of decomposing the fundamental matrix F becomes more challenging. One possibility is provided by Hartley [68], he proposes a method based on a *singular value decomposition* (SVD) of F to provide solutions for R,S,K_1 and K_2.

Maximum-Likelihood Based Solution

Instead of applying a linear least squares based approach, for reconstructing and solving F for the individual calibration matrices, we apply a Maximum Likelihood (ML) based approach. Let $Avec(F) = 0$ be the system (2.31), where the entries f_{ij} of F depend on the intrinsic and extrinsic camera calibration parameters that define K_1, K_2, R and t. The matrices K_1 and K_2 are defined by the respective focal length f and the aspect ratio $\frac{\alpha_0}{\alpha_1}$ (cf. term (2.4)). Here, the image offset (u_x, u_y) is set to zero. The rotation matrix R is defined by the three principle Euler angles α, β, γ specifying the rotation of the image plane about the three principal axes. The remaining three parameters $t = (t_x, t_y, t_z)$ control the displacement of the camera system. For the following let us represent the linear system (2.31) by the function $g(v)$, where v denotes the camera parameter vector. Then, the calibration problem can be reformulated as follows: Given the function

$$z = h(v, \omega) + \varepsilon, \tag{2.32}$$

measuring the quality z of a given parameter vector v, with $h(x, w)$ representing g, by incorporating a certain system error ω, and measurement error ε. The parameters v, ω and ε are unknown, but assumed to have a Gaussian distribution. The overall optimization problem is defined by

$$O(v, \omega, \varepsilon) = \exp\left(-\frac{1}{2} \begin{pmatrix} v \\ \omega \\ \varepsilon \end{pmatrix}^T \begin{pmatrix} C_v^{-1} & 0 & 0 \\ 0 & C_\omega^{-1} & 0 \\ 0 & 0 & C_\varepsilon^{-1} \end{pmatrix} \begin{pmatrix} v \\ \omega \\ \varepsilon \end{pmatrix} \right), \tag{2.33}$$

with $C_\omega^{-1} = \sigma^2 I$ and $C_\varepsilon^{-1} = \varrho^2 I$ denoting the covariance matrices of σ and ϱ, where σ and ϱ term the standard deviation. Since v is arbitrarily selectable we assume an infinite standard deviation, inducing $C_v^{-1} = 0$. An optimal parameter vector v is found, if v maximizes $O(v, \omega, \varepsilon)$.

To solve this optimization problem we linearize the problem by introducing the parameter adjustment vector u, with $\hat{v} = \hat{v} + u$. Be $\hat{v}$ the current parameter estimation and $\hat{z}$ the current quality measure. The linearization in $\hat{z}$ is given by

$$z = \hat{z} + Z_u u + Z_\omega \omega, \tag{2.34}$$

with

$$Z_u = \frac{\partial h}{\partial u}|_{v=\hat{v},\omega=0} \quad ; \quad Z_\omega = \frac{\partial h}{\partial \omega}|_{v=\hat{v},\omega=0}. \tag{2.35}$$

The linear system to solve now evolves to

$$BZ_u u = B(z - \hat{z}), \tag{2.36}$$

with

$$B = Z_\omega^T (C_\varepsilon + Z_\omega C_\omega Z_\omega^T)^{-1}. \tag{2.37}$$

After the parameter adjustment u is determined we update $\hat{v}$ and $\hat{z}$ by

$$\hat{v} = \hat{v} + u \tag{2.38}$$

$$\hat{z} = \hat{z} + h(\hat{v}, 0), \tag{2.39}$$

with $\omega = 0$ and $\varepsilon = 0$. The adjustment is repeated until the u becomes sufficient small. Due to reasons of numerically stability we substitute the Euler angles in v by the entries of the corresponding rotational matrix.
The initial estimation for v is defined by an approximation e.g., derived from equation (2.33) or provided by other cost-efficient methods like [96]. The parameters ω and ε initially are chosen to be zero.

Since we are working with sequences of corresponding images $\{M^0, \cdots, M^k\}$ we perform the parameter identification for each image pair (M^j, M^{j+1}) with $0 \leq j < k$. Thus, assuming the camera coordinate system of image M^j is tied to the reference coordinate system, and a sufficient number of point correspondences is extracted, the method de-

livers a specific projection matrix for each image pair (M^j, M^{j+1}). However, because of errors in the correspondence matching and numerical inaccuracies the recovered intrinsic camera parameter may vary. This has to be considered in the subsequent point reconstruction.

2.3.8 Point Cloud Reconstruction

Once, the intrinsic and extrinsic camera parameters are known, we can proceed computing the 3D coordinates of pre-selected image points. This is accomplished in a sense by projecting the image points back to the reference system. Next, we discuss the principles for determining the point $p_r = (x, y, z)^T$ in the reference space that projects onto $m = (u, v, 1)^T$ and $m' = (u', v', 1)^T$ of the two corresponding image planes M and M'. The equations of two rays originating at the two projection centers of the cameras passing through the two image points m and m' are defined by

$$wm = P_1 p_r, \tag{2.40}$$

$$w'm' = P_2 p_r, \tag{2.41}$$

where w and w' denote the involved scaling parameter. The projection matrices P_1 and P_2 as well as the coordinates of both image points $m = (u, v)^T$ and $m = (u', v')^T$ are known. Thus, we have 6 equations and 5 unknowns allowing the computation of the coordinates of the unknown point p_r by least square minimization of the corresponding residual.

Since we are focusing on point reconstruction from multiple images we need to somehow combine the relative coordinates of the 3D points recovered from each image pair into global coordinates. To accomplish this task, we obey the strategy proposed by [67]. Let $P_1, P_2, \cdots, P_n$ be the recovered projection matrices of the n cameras representing the individual images. Let $\{p_i = (x_i, y_i, z_i, 1)^T\}$ be a set of control points (given in homogeneous coordinates). The control points are defined by those points that are reconstructed from the image points $m_i = (u_i, v_i, 1)^T$ associated with different images (at best the x_i represent points corresponding to those image points for which accurate correspondences have been identified). Furthermore, let $P_{\sigma(i)}$ be the corresponding projection matrix. It is assumed that there exists a 4×4 non-singular matrix H that transforms each $P_{\sigma(i)}$ to

its real global placement. This consideration leads to the following set of equations

$$w_i m_i = P_{\sigma(i)} H p_i, \tag{2.42}$$

where the unknowns are the entries of H and w_i are scaling parameters. To compute the entries of H a minimum number of 8 control points are necessary. The solution can be determined up to a scale factor, corresponding to the fact that H itself is determined up to a scale factor. Once the matrix H is known the real 3D points can be reconstructed by applying the transformation H^{-1} to the earlier reconstructed points.

2.3.9 Results

In this section we review some of the results obtained from applying the proposed method to a set of assorted example image sequences. The results were obtained in collaboration with Kai Hammann. Further Results are provided by his Diploma Dissertation [64].

First, it is shown that the proposed ML based parameter estimation produces reliable results by comparing it to the standard least squares based approach.

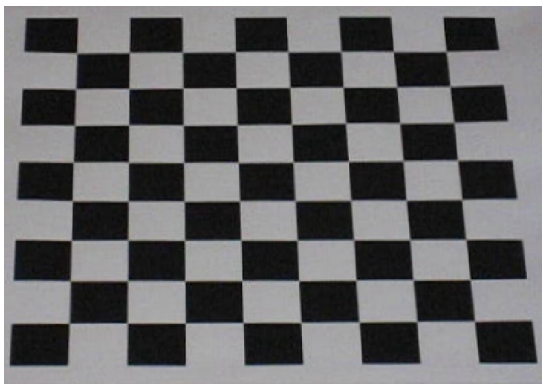


Figure 2.6: Chessboard used to perform quality comparison between least square based and the ML based reconstruction method.

To evaluate the method we compare the results of the ML-based and the least squares based calibration approach. This turned out to be rather challenging because the parameters produced by the two methods in general do not vary significantly. To proof this observation we have applied both methods to a test sequence of images showing a chessboard from multiple viewpoints (see Figure 2.6). In this example, the location

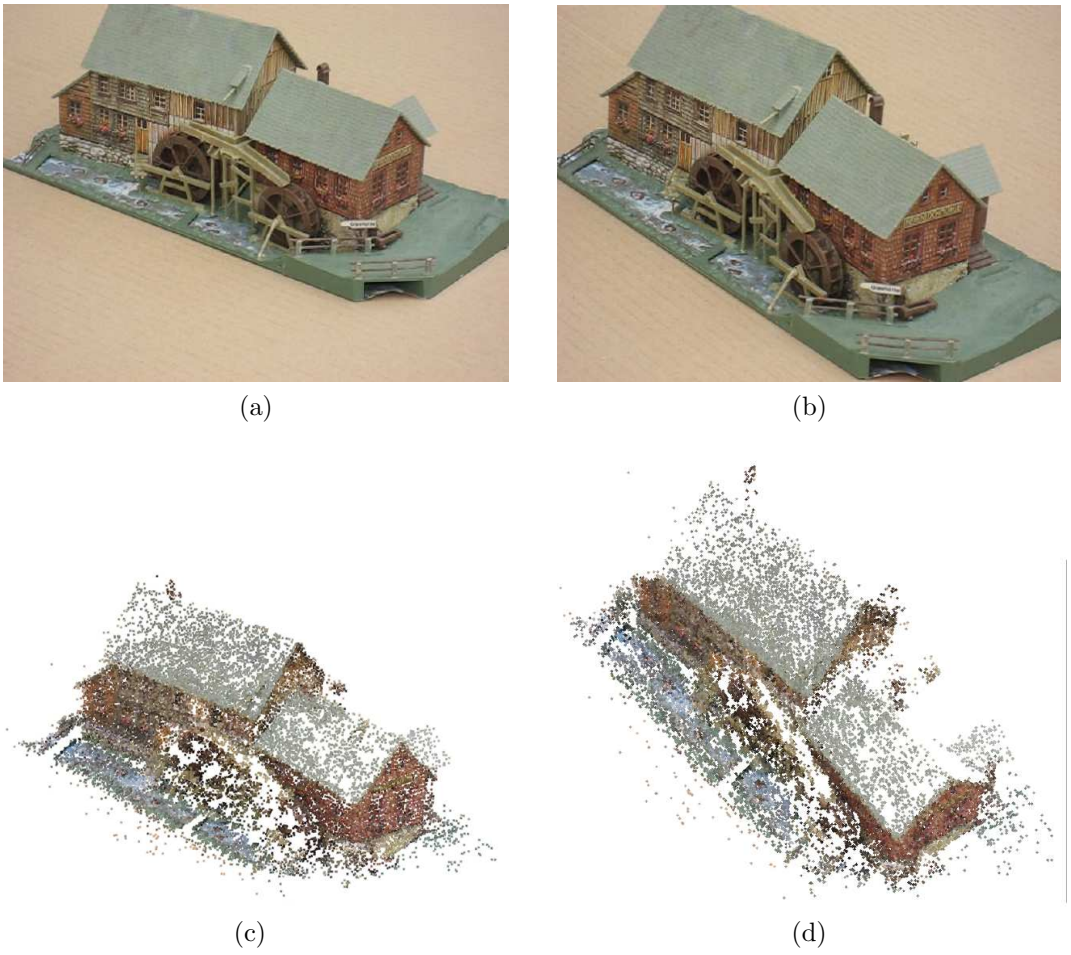
and the orientation of the camera were known in advance and the point correspondences have been set manually to agree with the corners of the individual chess fields between subsequent images. After performing the auto-calibration and the reconstruction we have compared the average deviation of the reconstructed corner points of both methods from whose real world coordinates. Table 2.1 presents the results depicting the average deviation in x-, y- and z-direction in millimeters.

	Least Squares	ML
δ_x	0.1585mm	0.1593mm
δ_y	1.4576mm	1.4608mm
δ_z	1.3459mm	1.3372mm

Table 2.1: Average deviation between the reconstructed and the real world coordinates depicted for the least squares based and the proposed ML based approach.

Next, we discuss some results regarding the reconstruction of point clouds from a set of example image sequences using different correspondence matching methods. Since in most examples the camera parameters are unknown and no reference point data exists allowing to determine the quality of the produced results, the analysis mostly is performed empirically based on the visual exploration of the generated point data. The data sets chosen are well suited to demonstrate the strengths and the shortcomings of the method and support an objective analysis. In all following examples the feature-based correspondence matching method proposed by Harris for recovering the unknown camera parameter has been applied. Compared to other correspondence matching methods, this approach provided best results.

The first example concerns the reconstruction of point clouds from images showing a miniature house (cf. Figure 2.7a and Figure 2.7b). The correspondence matching method chosen for the reconstruction part in this case was the intensity based approach by Kanade [98]. As shown in Figure 2.7c and Figure 2.7d the generated point cloud, colored according the corresponding image points to provide a better visual basis for exploration, mostly captures the prominent geometric properties. The object possesses a variety of interesting features that are almost completely recovered. Only some areas e.g., associated with the roof of the miniature house are partially not recovered. The main reason for this are missing or lost point correspondences. Although the roof area apparently does not exhibit prominent features, the correspondence matching provides a relatively high number of point matches suitable for reconstruction.



(a) (b)

(c) (d)

Figure 2.7: (a) and (b), miniature house images of an example image sequence used for reconstruction. (c) and (d), reconstructed point clouds at different perspectives.

Another interesting example is given by the stuffed animal image sequence, see Figure 2.8a and Figure 2.8b. The matching of correspondences again using the intensity based approach of Kanade delivers the best reconstruction. The generated point cloud presented by Figure 2.8c is almost completely recovered. In this case the detection of correspondences is reliable as long as the incidence of light causes shadows leading to an increasing contrast.

The last example demonstrates the problem regarding the reconstruction using feature-based correspondence matching. It shows a situation in which a rather incomplete point cloud is produced. Here, the results concerning the reconstruction from an image sequence showing the part of a chessboard including some chessmen is provided. Figure

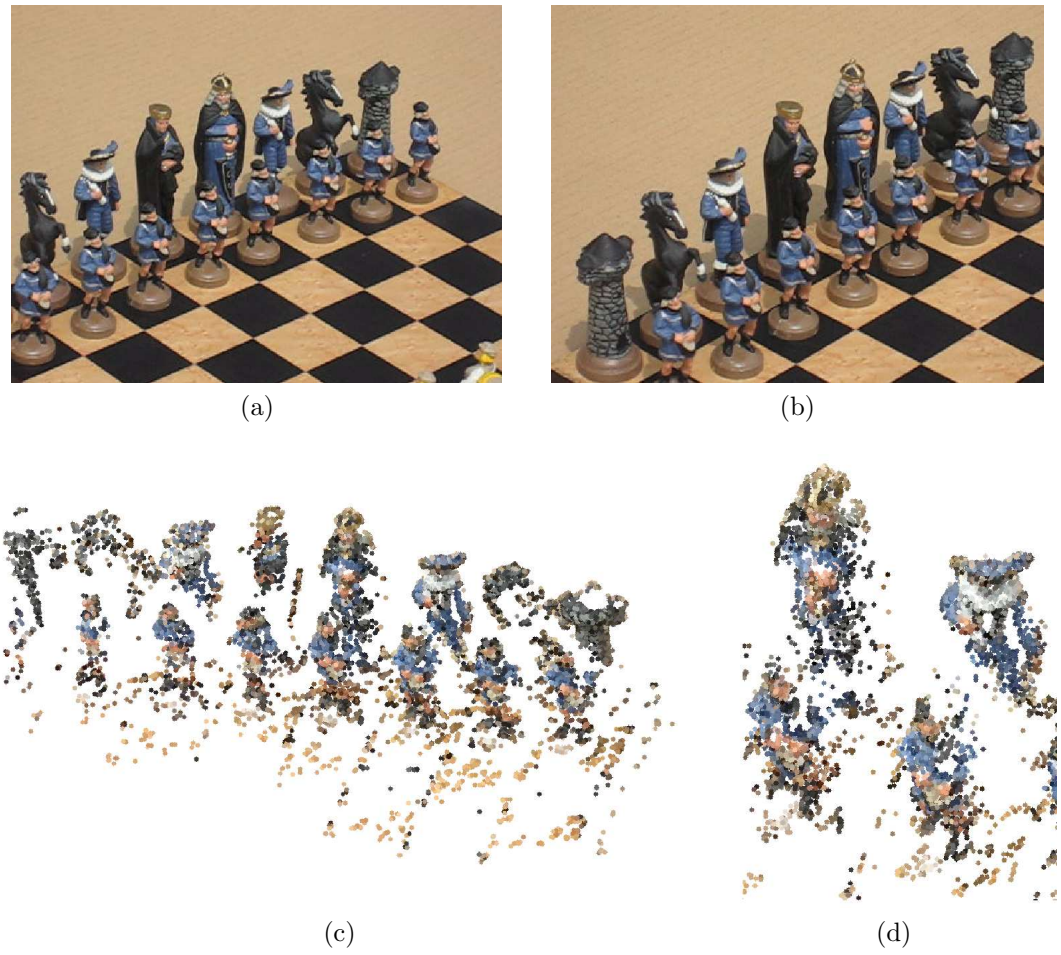
(a) (b) (c)

Figure 2.8: (a) and (b), images of the stuffed animal reconstruction example. (c) reconstructed point cloud.

2.9a and Figure 2.9b present some cutouts. The image points chosen for the reconstruction are provided by those image points for which reliable correspondence matches have been identified throughout the image sequence. The applied correspondence matching algorithm in this case is the feature-based approach introduced by Harris (see Section 2.3.6). From Figure 2.9c it is observable that the reconstruction of points associated with some image areas fail. Primarily, the reason can be traced back to inaccuracies and failures in the correspondence matching process. On the other hand this is due to missing image information e.g, caused by occlusion. Here, the importance of the correspondence matching becomes clear. Though, the image was taken from a certain distance, the auto-calibration and thus the points reconstructed are relatively accurate. As showed by image 2.9d, some of the details at the individual objects are accurately captured.

Also, the correspondence matching approach based on the method of Horn and Schunk [75] has been applied to the example images above. Since the method smooths the displacement vector field the reconstructed point cloud does not captures prominent features and rather delivers inaccurate results. One advantage is that it provides a high number of point correspondences.

As the examples demonstrate a reliable correspondence matching strongly depends on the actual lighting conditions. The existence of noise e.g., caused by reflections and transparency captured by the images is an important problem and may be reasons



(a) (b)

(c) (d)

Figure 2.9: (a) and (b) are images from the image sequence focusing the reconstruction of chessmen distributed on a chessboard. (c) and (d) show the reconstructed point cloud colored according the corresponding image values at different detail levels.

causing the reconstruction to fail.

All images used for reconstruction have been taken from a standard video camera having a resolution of 640×480 pixels. In general, the average number of images used for reconstruction was about 10, where the individual images have been assorted manually out of the image series produced by the video camera.

The generated examples above proof that the proposed reconstruction method in most cases provides accurate results even if the initial parameter estimates for ML auto-calibration are chosen arbitrarily. However, One remaining problem is that no explicit information about the variance of the measurement errors and the system errors is available in advance. For the considered examples the identity matrix was chosen to

initialize the covariance matrices. This in some cases does not lead to good results, hence estimating appropriate parameter scopes is essential for obtaining reliable calibrations.

2.3.10 Conclusions

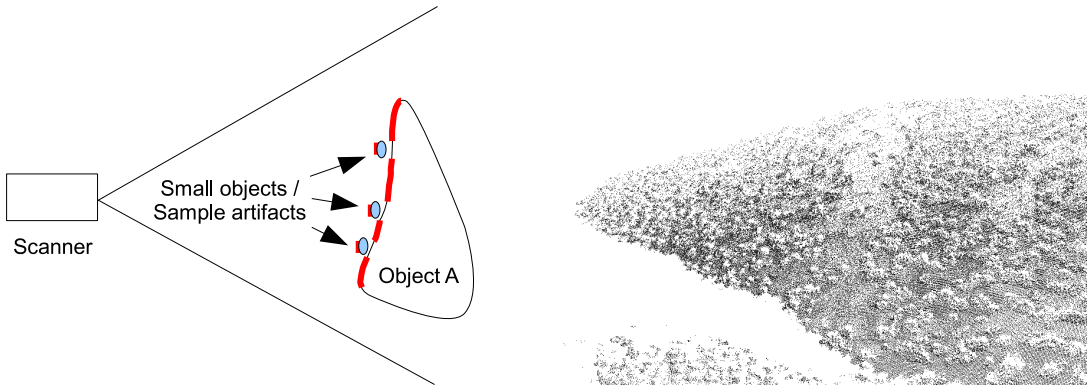
In this section an approach describing the reconstructing of point clouds from a set of corresponding image sequences is introduced. The proposed framework relies on a ML based method for estimating the intrinsic and extrinsic camera parameter like position and orientation using a feature-based image point correspondence matching. The parameter estimation was evaluated by comparing the ML approach to a similar least squares approach. In addition, [64] demonstrates that the quality of the reconstructed point clouds using the ML parameter estimation is comparable to those obtained by the standard least squares approach.

The biggest problem concerning the final point cloud reconstruction is represented by choosing the right correspondence matching method. Our analysis demonstrates the differences between the individual methods and highlights the advantages as well as the disadvantages associated with intensity- and feature-based approaches. The actual reconstruction step is less crucial, presumed reliable correspondences are found. The most important reasons for a possible failure of the reconstruction lies at the failure in the correspondence matching, primarily caused by varying light conditions. Another important influence factor complicating the identification of correspondences depends on the type of the scene the reconstruction concentrates on. This means, in case the scene is composed of a variety of different objects inducing a highly varying intensity distribution the matching process most likely delivers unreliable results. Hence, future work amongst others has to concentrate on the improvement and the incorporation of new correspondence matching methods.

2.4 Properties of Point Clouds

In the ideal case the points of point clouds are uniformly distributed, not afflicted with noise, free from scanning artifacts and do not exhibit any defects. However, because of measuring inaccuracies associated with the individual acquisition methods and adverse scan conditions that often appear, generating optimal point clouds mostly is not realizable. Developing methods, though, providing reliable results, even when processing affected point clouds, some qualitative aspects have to be considered. In the following we discuss some of the most important reasons (predominately in the context of physical-based acquisition) affecting the quality of point clouds.

2.4.1 Noise



(a) (b)

Figure 2.10: (a) small objects producing scan artifacts which due to undersampling are interpreted as noise. (b) point cloud showing an environmental scan concerning a wide velt area which features noise/scanning artifacts primarily caused by bushes.

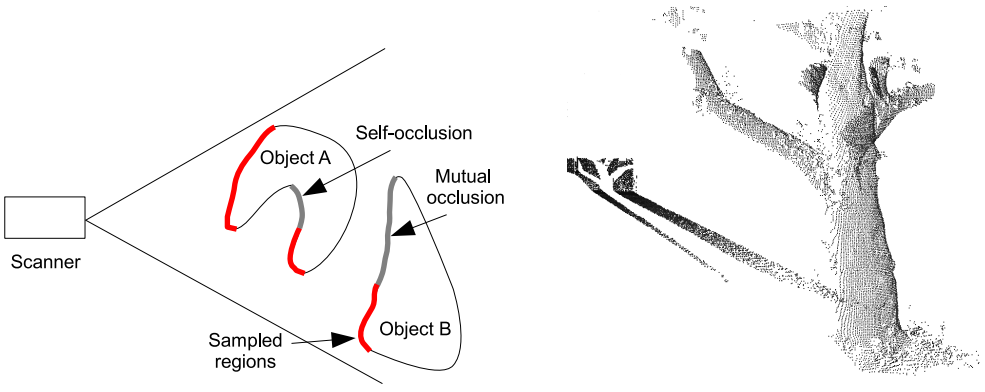
One possible reason for noisy point samples may be inaccuracies associated with the corresponding acquisition method. For example, scans performed from great distances e.g. by airborne, satellite scanning systems are afflicted with higher noise than scans performed with stationary devices. Another possible source for an increased noise level are scanning artifacts caused by the undersampling of detailed scenes. For example, when assuming environmental point clouds containing trees and bushes small artifacts (e.g. leafs, grass, etc.) may be scanned and interpreted as noise. Adapting the scan resolution does not always solve this type of problem, especially, when considering scenes

consisting of heterogeneous objects having different size. Furthermore, we have to take into account the possible movement of objects (e.g., humans, cars, trees, etc.), also significantly affecting the quality of the final point cloud. Theoretically, noise can be measured by the so called *signal to noise ratio* (SNR): Given a set of n sample points $X = \{x_i, \cdots, x_n\}$ with $X \subseteq \mathbb{R}^3$ afflicted with noise $x_i = y_i + e_i$, where $y_i \in Y$ is the original point without noise and $e_i \in \mathbb{R}^3$ is the so called noise vector, then the SNR is determined by

$$SNR(X) = \log \left(\frac{\frac{1}{n} \sum_{i=0}^{n} \|x_i\|}{\sqrt{\frac{1}{n} \sum_{i=0}^{n} \|x_i - y_i\|^2}} \right); \tag{2.43}$$

Furthermore, the point samples X are set to be δ-noisy, if the condition $\|e_i\| \leq \delta$ for all $0 \leq i \leq n$ holds. If one of these parameters becomes too large, reliable data processing becomes difficult. However, deriving those indicator values is challenging, especially when considering point clouds obtained from LiDaR scan devices, since no reference data is available.

2.4.2 Object Occlusion



(a) (b)

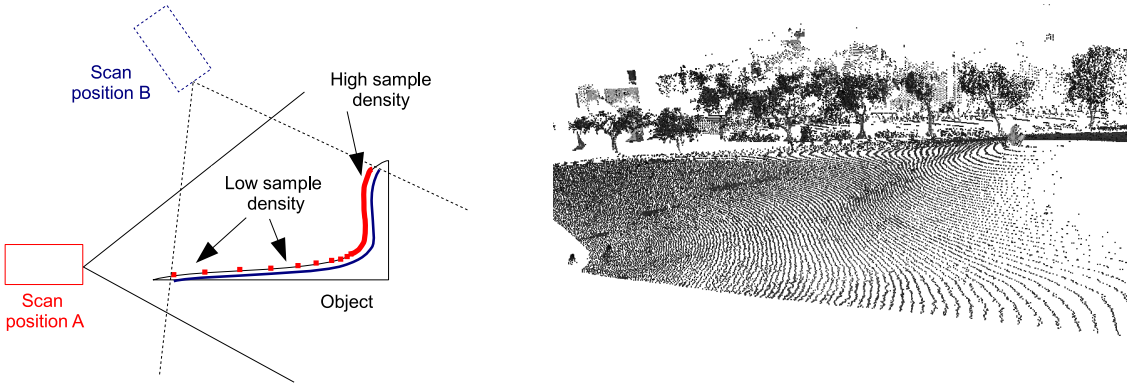
Figure 2.11: (a) principles of occlusion at the point cloud acquisition by using a 3D lasers scanner. Object A prevents surface parts of Object B to be sampled, simultaneously self-occlusion avoids that the front of object A is completely acquired. (b) point cloud in which the front object interferes with the acquisition of the surface structure in the back.

Often, point cloud acquisition concentrates on scenes including several objects having unknown [1] topology. In this case samples of objects or parts of objects may occlude other point regions featuring relevant structural information. In this context, we distinguish between the mutual object occlusion and the self-occlusion. In the first case objects are located in front of others avoiding the latter to be fully sampled (cf. Figure 2.11b). Self-occlusion occurs, if one or more parts of an object avoid the acquisition of other parts of the same object. Figure 2.11a illustrates the principles with a simplified example. In order to overcome this problem a possible solution is provided by performing scans from multiple positions and combining the resulting point clouds. However, not every object, due to the way they are conditioned, allow the acquisition from multiple viewpoints. Another problem that comes up with this approach is the problem of overlapping point clouds which we address in the following. Occlusion is also an important problem in the context of reconstructing point clouds from objects in 2D images. Since most methods exploit object-based correspondences between 2D images, occlusion leads to the loss of these correspondences. This is critical since it significantly complicates a precise reconstruction.

2.4.3 Sampling Density

Choosing proper sampling density has impact on the quality of the acquired point clouds. This requires a certain knowledge of the properties of the focused structures in particular the size, the consistency as well as the conditions under which the sampling is performed. Given a set of n sample points $X = \{x_i, \cdots, x_n\}$ with $X \subseteq \mathbb{R}^3$, a sample x_i is said to be ρ-dense, if ρ is the radius of the largest sphere that only contains x_i. From this notation we can derive a parameter measuring the average density $\rho_{avg} = \frac{1}{|X|} \sum_{x_i \in X} \rho_{x_i}$, where ρ_{x_i} denotes the radius ρ of point x_i. In general, if the average sampling density ρ_{avg} is too low with respect to the size of the objects to be acquired the risk of losing details is high. On the other hand, oversampling may cause capturing to much information also inducing problems in subsequent processing. Hence, adjusting the acquisition properly is essential. When applying 3D scan devices another problem comes up concerning alternating sampling density. Depending on the positioning and the orientation of the scanner with respect to the scene, the generated point clouds my be over-/undersampled

[1] The word unknown in this context refers the fact that no explicit information is given describing the topology or the structural composition of an object.



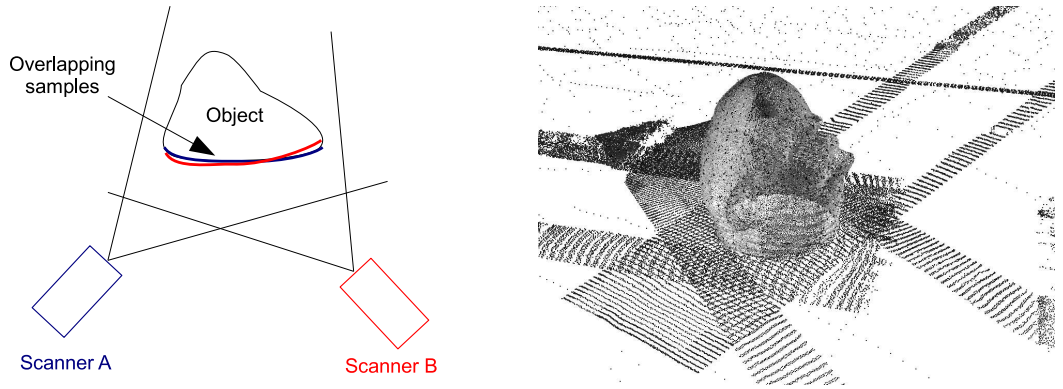
(a) (b)

Figure 2.12: (a) reason for alternating sample density within regions of point clouds. The scanner at position A causes the Object to be under-sampled at its frontal section. The scan performed at the alternative position B generates samples of constant density. (b) acquired point cloud in which regions are undersampled.

in some areas. In particular, this happens if the reflected laser beam establishes a wide angle with the corresponding surface normal. Figure 2.12a illustrates this effect with a simple example, where the quality of the samples generated at scan position B is better than the quality of the samples produced at scan position A. As Figure 2.12b demonstrates, this becomes a big problem when considering the scanning of wide areas. In some cases this effect can be avoided or reduced by adjusting scan parameter (e.g., location, resolution) or performing multiple scans, whereas in other cases this is not compensable and has to be considered when applying further processing.

2.4.4 Overlapping Point Samples

Overlapping occurs in regions where several data samples of performed acquisitions merge. This happens, especially, when combining 3D scans from multiple viewpoints (see Figure 2.13a). Overlapping causes oversampling and could drastically decrease the quality of the point cloud within the affected regions. Another problem concerns the accuracy concerning the extraction of features like surfaces or curves, since the different point samples may be shifted against each other (see Figure 2.13b). In some cases the overlapping of different point samples may be avoided or reduced by choosing proper acquisition parameters or applying additional filter methods. As shown in section 2.3.9 overlappings in 2D images is an issue when considering the acquisition of point clouds



(a) (b)

Figure 2.13: (a) showing two laser scan devices with overlapping scan focus, (b) overlapping point samples due to scanning of an object from multiple directions.

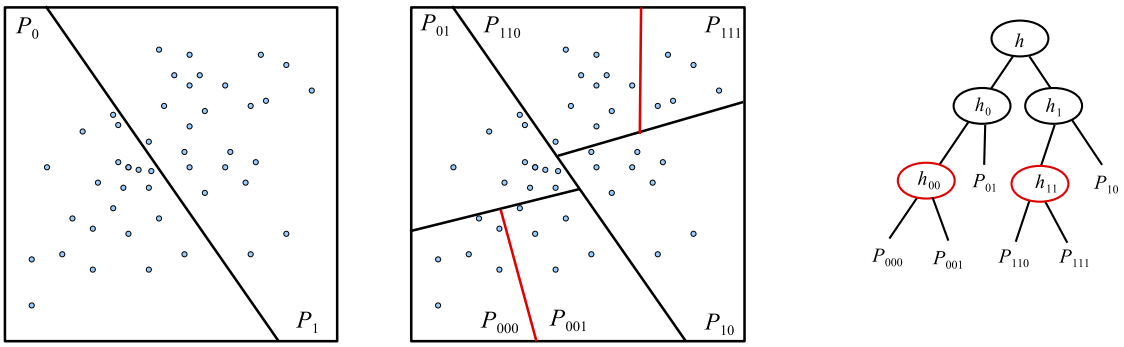
from 2D images since it causes the point correspondence matching to fail.

2.5 Data structures for Organizing Point Clouds

In order to efficiently access and process large point clouds it is necessary to apply appropriate data organization schemes. Let us assume P to be a set of unorganized points with elements $p \in P \subseteq \mathbb{R}^n$ within n-dimensional space. The goal is to partition P into a finite number of disjoint subsets $P = \bigcup_i P_i$ according certain conditions (e.g., distance, density, etc.) in a way that the partitioning allows efficient processing and visualization. In the following, we introduce some of the most important concepts regarding efficient organization of point clouds.

2.5.1 Binary Space Partitioning Trees

A *binary space partitioning tree* is a data structure obtained by applying a recursive space partitioning scheme, known as *binary space partitioning* (BSP). It allows to hierarchically cluster pairwise disjoint objects (e.g., points) in n-dimensional space by partitioning the space along a given hyperplane h into two half-spaces. This is performed in a way that each half-space contains a fraction of the original point set (see Figure 2.14a and Figure 2.14b). A hyperplane h in n-dimensional space thereby represents a (n-1)-dimensional planar surface, for example, a hyperplane in 2D space is represented by a line. The partitioning scheme can be applied, recursively, to further subdivide the obtained half-



(a) (b) (c)

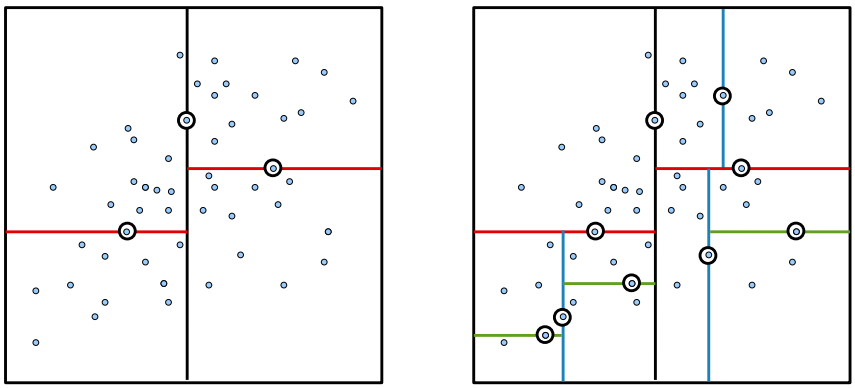
Figure 2.14: (a)-(b) examples illustrating the 2D BSP space partitioning. (a) partitioning after one refinement step, (b) partitioning after three refinement steps. (c) corresponding BSP tree.

spaces. The emerging data structure naturally corresponds to a *binary tree* (see Figure 2.14c) where the root represents the original object space enclosing the entire point cloud. Each partitioning step then creates two new leaf nodes corresponding to the introduced half-spaces. The remaining inner nodes of the tree represent the splitting hyperplane. The way the hyperplane partitions the object space has major impact on the quality of final BSP tree. Using the most common convention: A BSP-tree is optimal (well-balanced), if the number of points contained by the leaf nodes is (approximately) equal. One possible approach for obtaining a well-balanced BSP-tree is to define the orientation and anchor of h with subject to the principle orientation and the centroid of the considered point set.

BSP-trees find major application in the field of computer graphics where they are used e.g., for collision detection [13] or surface simplification [131, 117]. When considering unstructured point clouds BSP-trees are well-suited to speed up point operations like k-nearest neighbor searches.

2.5.2 KD-Trees

A KD-tree [19] is a specialization of a BSP-tree where each node represents a domain in k-dimensional space. Similar to the BSP-tree the KD-tree is a binary tree where each non-leaf node represents a hyperplane splitting the space into two halfspaces. The resulting



(a) (b)

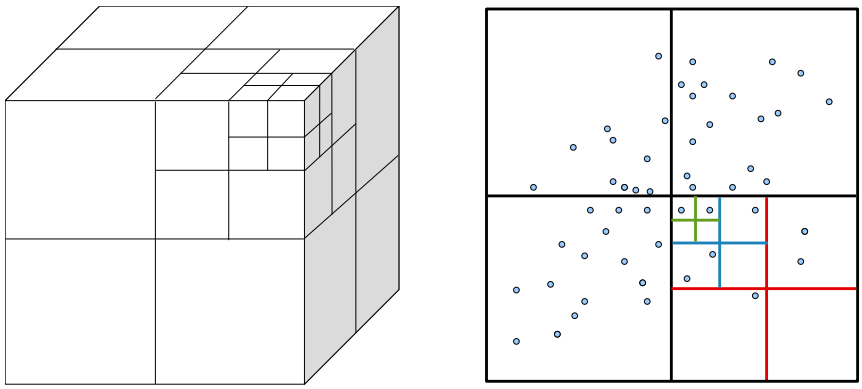
Figure 2.15: (a)-(b) examples illustrating the two-dimensional KD-tree space partitioning scheme applied to a set of arbitrarily distributed 2D points. (a) partitioning of the point cloud after two subdivision steps, (b) after four subdivision steps.

leaf nodes represent the final subspaces. In contrast to BSP-trees, the normal direction of the hyperplane is chosen with subject to the orientation of one of the k principal vectors spanning the k-dimensional space. Given a set of n points in k-dimensional space the generation of the KD-tree is performed recursively according the following rules: First, the median point is determined for which the hyperplane is oriented perpendicular to the ith principal vector. The resulting hyperplane divides the point cloud into two subsets, each having about the same number of points. This is recursively performed for each introduced subset by substituting i with the $(i + 1)$ re-orienting the splitting hyperplane. In the two-dimensional case the hyperplane would be alternately oriented to be perpendicular to the principal x- and y-axis. Figure 2.15a and Figure 2.15b illustrate the principles of creating a KD-tree in 2D.

KD-trees are essential data structures used for accelerating several operations like k-nearest neighbor (KNN) or approximate k-nearest neighbor (ANN) searches. Some important applications of KD-trees are ray tracing [49] [76] and point based rendering.

2.5.3 Octrees

An *octree* is another type of hierarchical data structure, quite similar to the BSP- and KD-trees, where each node has up to eight subnodes. Often, this type of data structure is used to recursively partition 3D space into eight uniform subspaces. These subspaces, or octants, are often referred to as *voxels*. One important advantage of the octree concept



(a) (b)

Figure 2.16: (a) example illustrating the the principle octree-based space partitioning, (b) 2D example regarding the partitioning of point clouds by quadtrees.

is that it can be adapted to efficiently organize point clouds (see Figure 2.16a and Figure 2.16b). For this purpose, the root of the octree representing the original space partition containing the whole point cloud is subdivided, recursively, until the desired subdivision level is reached or a predefined stop condition is fulfilled (e.g., max. number of points contained by a voxel). Commonly, empty subvoxels are deleted. The remaining leaf nodes represent subspaces partitioning the original point cloud according the criteria of point density. This means, the partitioning is detailed at regions the point cloud offers a high point density. Again, the quality of an octree is stated as good if the generated subspaces predominately contain the same number of points. Octrees are the 3D analogon to the 2D *quadtrees*, where each node has a maximum of four subnodes.

Main application fields of octrees are ray tracing, volume rendering collision detection, geometric modeling and LOD-applications. One of the main advantages of this kind of data organization is its adaptivity as well as its efficient handling.

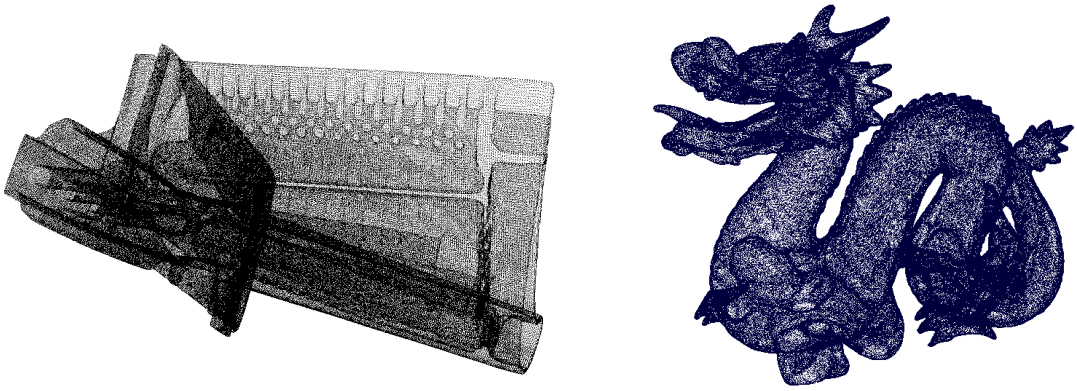
2.6 Examples

In this section we review some examples in order to demonstrate the differences between the point clouds obtained from the different acquisition techniques. Figure 2.17 shows a typical environmental point cloud (having about 4,8 million points) that consists of a great variety of heterogeneous objects. The scan focusing a part of the campus of UC Davis in CA, USA, was performed by a terrestrial LiDaR scanner. In this example noise

caused by scanning artifacts, density, as well as occlusion are fundamental issues. Figure 2.18 shows some point clouds obtained by stationary 3D laser scan devices. Figure 2.18a represents the scan of a turbine blade [1], Figure 2.18b shows the Stanford dragon point data set [2]. In contrast to point clouds obtained by e.g., tripod-based LiDaR scan devices, these point clouds are well defined i.e., the acquired point data mostly has constant sampling density and a low noise level. Figure 2.19 shows two examples obtained by reconstruction from 2D image sequences. Figure 2.19 shows the reconstructed point cloud of images of a vase, Figure 2.19 depicts the point cloud reconstructed from images showing a house. Since the resolution of the camera the images are taken with is confined, point clouds obtained from single stereo image pairs are composed of relative few points. Further problems concern the noise caused by inaccuracies in the reconstruction process that is due to varying light conditions and occlusion.

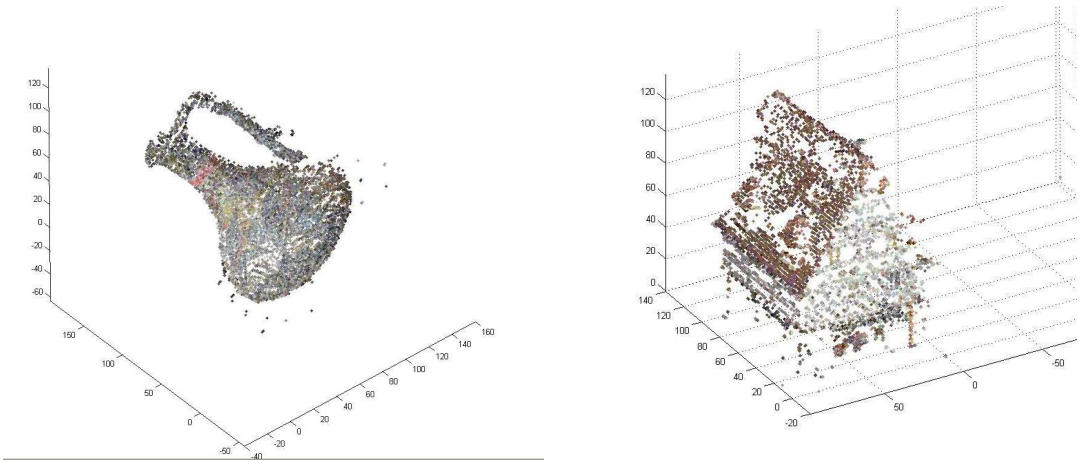
Figure 2.17: Environmental point cloud consisting of about 4.8 million points, emerged from a terrestrial LiDaR scan.

[1] Web address: `http://www.cc.gatech.edu/projects/large\_models/`
[2] Web address: `http://graphics.stanford.edu/data/3Dscanrep/`



(a) (b)

Figure 2.18: (a) point cloud representing a scanned turbine blade consisting of about 882,000 number of points, (b) Stanford dragon data set consisting of 437,645 points.



(a) (b)

Figure 2.19: (a) point cloud consisting of 8,567 points which was reconstructed from a set of corresponding 2D images showing a vase, (b) point cloud (8,429 points) representing a house.

3 Feature-Based Analysis of Point Data

3.1 Overview

The exploration of environmental point data sets is a challenging problem of recent interest. The LiDaR technology makes it possible to capture fast and accurate point data over large regions. As a consequence, LiDaR data sets are typically very large and contain diverse and highly complicated structures. In this chapter we focus on the analysis of point data including the detection, the extraction, the measurement and the visualization of relations between points or group of points (e.g. point distribution, point density, etc.) as well as the relations between points and other geometric structures (e.g., surfaces, lines, etc.). In this context, we distinguish between the structure-based and the object-based analysis. The former is concerned with analyzing the point data according its principle structural composition (e.g., surfaces, curves,etc.). The object-based analysis concentrates on filtering relations between points and predefined-defined objects having arbitrary complexity (e.g., identification and classification of buildings), often performed, with a preceding structural analysis.

In general, the raw point data does not exhibit any scientifically meaningful structural information facilitating its analysis. A possible way to still explore point data is based on surface reconstruction [12] [36] [37] (see section 4). However, Building consistent surface representations from such data is a difficult and often ambiguous task. Instead, the method proposed in this chapter, see [83], filters the point data according special criteria by using a probability-based classification of points into subsets having different structural characteristics. Depending on this classification and a user's choices of parameters the extraction of feature graphs capturing essential properties is performed. This way an explicit description of the structural composition of point data is provided

that can be used to support further processing (e.g., identification and classification of objects in environmental data) or to improve visual exploration, allowing the user to cope with large point sets without loosing orientation.

For geologists, this approach is of special interest since it makes possible the detection and visualization of features e.g., representing discontinuities in structures caused by earthquakes. It also facilitates the rapid automatic identification of key features in environmental data such as ridge lines, stream beds, and edges of terraces. The goal is to provide the user with effective tools for identification and interactive exploration of structural features in LiDaR point data. The contributions of this chapter are:

- The introduction of a new probability-based scheme for classifying points according their structural affiliation.
- New concepts for interactive multi-resolutional visualization of point cloud inherent structural information.
- A new method for probability-based extraction of feature graphs, especially suited for noisy point clouds having high complexity.

The chapter is organized as follows: In Section 3.1.1 we review some standard approaches regarding the extraction and the visualization of structures from point clouds obtained by LiDaR scans. Section 3.2 provides a discussion about the principle structures the point analysis is concerned with. In Section 3.5 the novel point-based classification scheme allowing a multi-scale feature-based structural analysis of 3D point clouds is introduced. This point classification characterizes points by comparing its neighborhood shape with respect to three principal geometric shape classes. Based on this classification Section 3.7 describes the method for generating feature graphs that allow to capture prominent point cloud inherent structures. Section 4.5.6 provides and discusses some results generated by the proposed method, followed by a short summarization and ideas for future work.

3.1.1 Previous Work

The visual understanding of 3D point data is driven by an user's effort to identify special features that describe the topology of objects. Visualizing structures within point clouds in a way that the exploration of certain intrinsic characteristics is supported, without applying excessive pre-processing (e.g., surface reconstruction) is an important task. Especially, if considering unorganized large-scale point clouds in 3D space.

There exist several approaches providing a variety of different solutions. In this context, point-based rendering techniques [71], [120], [54], [26] recently become quite important since they allow visualizing smooth surfaces defined by point sets [7]. Only few local geometric information like normal vectors [145] [146], principal curvatures and/or curvature directions need to be retrieved, but no mesh-like connectivity graph is required. Most of them support the level of detail concept and allow the multi-resolutional representation of point clouds either by using kd-trees, octrees or other types of hierarchical data structures like hierarchies of bounding spheres [126], or sequential point trees [33]. An overview concerning point-based rendering is provided by [127].

The potential of point-based methods has been demonstrated in different applications [8],[91]. However, algorithms to process point sets are still in their infancy when compared with algorithms for common mesh-based approaches. In computer-aided design (CAD), point sets may be used as underlying representation for surface editing [119] (like the Pointshop 3D framework proposed by Zwicker et al. [159]). Several authors based their segmentation and surface recognition algorithms directly on point sets, [17], [18], [147], [150], and [151]. Regarding the post-processing of laser range data, a variety of techniques improving the quality can be applied [106], [152]. Concerning surface reconstruction from laser-scanned point sets, much research has been done following the approach by Eck and Hoppe [40].

Feature lines produce important landmarks which besides of improving visual aspects are used for constructing meshes [46], [104] or for recognizing shapes like buildings [9]. In the context of complex geology-driven applications, besides the problem of correctly recovering surface topology, sharp feature lines like ridges, ravines or crest lines [70], [155], [138] need to be identified.

Concerning point sets, there have been several efforts for robust feature detection [60] [16], [34]. Neal [112] provided an instructive survey of such techniques. Some of these methods perform feature extraction based on the estimation of curvature values. These values are obtained from surfaces that are approximated via methods like moving least-squares (MLS) [94]. The principal curvatures and curvature directions carry important information about local surface behavior, but their estimation is numerically sensitive to noise present in the data, and many estimation methods have been proposed [39], [63], [55], [86], [124], [125], [143] [39], [63], [86]. Hamann [63] approximated principal curvature values based on considering a local, triangulated point neighborhood - the "platelet" - and determining a least-squares quadratic bivariate polynomial. Krsek et al. [86] provided an analysis and comparison of the estimation quality of three different methods: a) approximation by circles, b) Hamann method and c) a Dupin cyclic method. Buxton [39] approximated the neighborhood with an implicit polynomial of second order (a quadric surface). Rusinkiewicz [125] presented an efficient finite-differences approach for irregular triangle meshes that resembles the common method for estimating per-vertex normals on meshes. Goldfeather et al. [55] extended the approximation of the neighborhood with a bivariate function (generalization of Hamann method for neighborhoods) with estimated normal vectors. Razdan and Bae [124] proposed a Bézier patch-fitting method, computing the curvatures from biquadratic Bézier patches that are locally fit to the data.

Other methods directly work on point clouds without previous surface reconstruction. Gumhold et al. [60] provides a method for feature extraction based on defining a penalty function which assigns weights to points indicating the unlikelihood that a point is part of a feature. The approach of Pauly et al. [118] estimates the curvature at multiple-scales by using principal component analysis. Most of these approaches do not differentiate between the structural types a feature might belong to, which makes them inappropriate for data sets of arbitrary topology especially point clouds resulting from environmental scans.

3.2 Principle Point Cloud Structures

Point clouds acquired by 3D scans or by reconstruction from other data sets may represent complex objects assembled of structures of arbitrary complexity. In the following we discuss some principle structures important in the context of the subsequent point cloud analysis. These structures primarily are surfaces, curves and so called discontinuities.

3.2.1 Surfaces

In the fields of computer graphics and computer-aided design, surfaces are a natural way of representing objects. They are used to provide the user intuitively with information about the shape of objects.

There are several understandings of the term surface: In the field of mathematics, a surface is referred to as an orientable 2D topological manifold. A two-manifold surface S in 3D Euclidean space often is defined by parametric equations of the form

$$f : \mathbb{R}^2 \rightarrow \mathbb{R}^3. \tag{3.1}$$

Another way of defining surfaces is implicitly as level sets of 3D real-valued functions

$$\{(x, y, z) \in \mathbb{R}^3 | f(x, y, z) = c\}, \tag{3.2}$$

where $c \in \mathbb{R}^3$ is a constant. Such surface representations, also called *level surfaces* or *level set surfaces*, typically are extracted from (time-varying) volume data as it emerges from the field of medical scanning and computational fluid dynamics (CFD).

In computer graphics, surfaces are visually represented by using geometric structures like meshes (cf. Section 4.2.4). A mesh is a piecewise linear geometric structure consisting of connected elements like vertices, edges, and faces that are describing the shape of objects. Another possibility of representing surfaces is by using points sets. This form of representation in particular is interesting when working with large unstructured 3D point data sets.

3.2.2 Point Set Surfaces

Point Set surfaces are an important concept for representing surfaces. Given a set of points $P = \{p_i\}$, $p_i \in \mathbb{R}^3$, $i \in \{1, \cdots, N\}$ a *point set surface* S_P is implicitly defined by a projection operator $Q(p)$, projecting points onto S_P. Formally, the surface S_P is a set of points that project onto themselves. Thus, the surface itself is defined by both the projection operator and the points on the surface. In principle, the projection operator Q can be adapted in a way that the point set surface reflects the characteristics of any form of surface, for instance, be it parametric [5], implicit [6], or algebraic [58]. A more formal introduction to point set surfaces including the derivation of projection operators is given by [5] [7] [45]. Some of the most important properties of point set surfaces are:

- The surface is guaranteed to be locally two-manifold.
- Noise and redundant information are removed providing a smooth representation.
- The visual representation of S_P can be easily adapted by introducing/removing points to/from the surface.

In a sense, point set surfaces are omnipresent in practically every 3D point cloud, whether obtained by 3D scanning or the reconstructed from 2D images. However, identifying points corresponding to such "hidden" surfaces is a rather difficult task since point clouds generally contain a wide range of structural entities. In the subsequent sections, amongst others, we aim at identifying and measuring correspondences of points to such surfaces. The process of recovering and representing surfaces from identified point sets by suitable geometric and mathematical models is discussed more detailed in chapter 4.

3.2.3 Feature Lines

Feature lines, intuitively spoken, are curves on surfaces visually representing most salient characteristics. Mathematically seen, feature lines are defined as local extrema of principal curvatures along corresponding principal directions. Given a smooth surface M in $\mathbb{R}^3$ and a Gauss map, which associates each point on $p \in M$ with an oriented unit normal vector $n(p)$, establishing a normal field N. The derivative $\nabla n(p)$ representing the Jacobian matrix of the Gauss map measures the variation of the normal vector near

a point p. The Jacobian expressed in terms of local surface coordinates defines the shape operator $S = -dN$, also known as *Gauss-Weingarten map* (see Appendix A), which describes a symmetric tensor field. The points where S is a multiple of the identity map are so called umbilic points. All non-umbilic points of the shape operator have two different eigenvalues which are called principal curvatures. The corresponding eigenspaces are the principal curvature vector fields. For the following let κ_{max} and κ_{min} be the principal curvatures values with $\kappa_{max} > \kappa_{min}$, and $\vec{k}_{max}$ and $\vec{k}_{min}$ be the corresponding principal curvature directions.

We consider the derivatives of the principal curvature values along their corresponding curvature directions $e_{max} = \langle \nabla \kappa_{max}, \vec{k}_{max} \rangle$ [1] and $e_{min} = \langle \nabla \kappa_{min}, \vec{k}_{min} \rangle$. The extrema of the principal curvatures along their curvature directions define the so called ridge- and valley lines, which are the zero crossings e_{max} and e_{min}. Ridge lines are characterized by

$$e_{max} = \langle \nabla \kappa_{max}, \vec{k}_{max} \rangle = 0, \quad \langle \nabla e_{max}, \vec{k}_{max} \rangle < 0, \quad \kappa_{max} > |\kappa_{min}|, \tag{3.3}$$

whereas valley lines are defined by

$$e_{min} = \langle \nabla \kappa_{min}, \vec{k}_{min} \rangle = 0, \quad \langle \nabla e_{min}, \vec{k}_{min} \rangle > 0, \quad \kappa_{min} < -|\kappa_{min}|. \tag{3.4}$$

In other words, ridge-, and valley lines are the set of points at which the normal curvature in the principal direction associated with the largest, positive curvature assumes a local maximum, minimum. The terms ridge- and valley line, originate from the field of geology where they describe more or less abrupt changes within topographic elevation maps. In computer graphics, both, ridge- and valley lines are often associated with sharp changes in surface intensity due to the locally rapid rate of change of the surface normal direction. These lines correspond to geometric features of surfaces, and as such they can be computed automatically from local measures of the surface's differential geometry. Another donation for feature lines especially used in association with smooth surfaces or meshes is *crease lines*. Since ridge- and valley lines exchange their roles, if the orientation of the surface flips, we only use the term ridge lines.

Though, feature lines are defined over surfaces, point clouds in some sense, still might

[1] $\langle a, b \rangle = a \cdot b$ denotes the inner product of two vectors.

exhibit such kind of structures. However, identifying the corresponding regions within a point cloud is challenging, foremost since no explicit surface representations are available. There are a bunch of methods dealing with this kind of problem e.g., [60] [16] [118]. The general approach is to determine a set of surfaces that locally approximate parts of the point cloud and then use these for identifying feature lines. In section 3.1.1 we give a detailed overview about the related techniques.

3.2.4 Feature Curves

In some cases, the acquired point structures e.g., due to reasons of undersampling may collapse to curve-like structures, though, the underlying original point structure represents a surface. The feature type associated with such point structures is referred to as *feature curve*. These feature curves in principle are related to feature lines but are not measurable the same way, and hence are hard to extract. In the following sections some methods capable of identifying and measuring the strength of such feature curve within 3D point clouds are introduced.

3.2.5 Structural Ambiguities and Discontinuities

Primarily, discontinuities represent regions within point clouds whose underlying local structure is not uniquely determinable. Examples for structural ambiguities are conjunctions of feature lines related to the so called umbilic points associated with surfaces e.g., points at sharp corners of objects, or as it occurs at intersections between point set surfaces e.g., in regions where several surfaces of objects meet and form non-manifolds.

In geometric modelling the intersection between different geometric objects like surfaces and/or curves are relatively easy to determine because of the presence of underlying mathematical representations. However, transferring the detection to unorganized 3D point clouds makes this task more complicated. To still allow identification several approaches incorporate structural information recovered by preceding point processing.

3.2.6 Terminology and Notation

In the remainder, the term surface, used in association with 3D point clouds, always refers the natural "hidden" surface the point samples are obtained from. In contrast,

point set surfaces or *point surfaces* denote the surfaces approximating the original surfaces. The term structural discontinuity or ambiguity always denote regions of the point cloud at which the underlying structure is not uniquely determinable or recognizable. These regions are also referred to as *critical regions*.

3.3 Data Management

To provide a LOD-representation of unstructured large-scale point data sets an octree-based approach is applied. The established octree is adaptively subdivided and only consists of non-empty nodes. Each non-leaf node has a number of subtrees whose leaf nodes represent parts of the original point cloud at its finest resolution. By performing downsampling each octree level is adapted to provide a certain point resolution. Informally, this is accomplished by iteratively removing points from dense regions of the point cloud. Starting with the point cloud at the leaf nodes the process proceeds until a predefined level-specific point density is reached. The modified point cloud then is reassigned to the corresponding non-leaf nodes at the next coarser octree level. This operation is performed recursively until the root node is reached.

Instead of keeping all data in main memory, the content of each node is stored on disc using a special file format. At any time, only the octree structure as well as the content of the nodes, needed to represent the focused regions of the point cloud, is present in the main memory. In order to refine or coarsen the point clouds resolution at certain regions the content of the respective octree nodes can be reloaded from disc. This approach allows an interactive LOD visualization of large unorganized 3D point clouds. It also significantly speeds up frequently used operations like nearest neighbor searches.

3.4 Identification of Feature Lines

This section introduces the principle processing steps for measuring and visualizing surface related feature lines. This is performed by applying methods directly operating on the specific regions of the point cloud.

3.4.1 Local Feature Size

The identification of feature lines depend on several parameters like feature size, density and noise. The *local feature size* (LFS) in some sense quantifies the local "level of detail" at a point p on a smooth curve. Amanta et. al [10] estimated the *local feature size* $LFS(p)$ of a point p on a surface S as the Euclidean distance from p to the *medial surface*. The set

$$B_{c,r} = \{p \in \mathbb{R}^3 | \|p - c\| < r\} \tag{3.5}$$

represents a ball with radius r and center c. Let $S \subseteq \mathbb{R}^3$ be a compact smooth surface without boundary then the medial surface M of S is the locus of the centers of the maximal balls whose interiors are empty of points of S. The *local feature size* at a point $p \in S$ is defined as $f(p) = d(p, M)$ where d denotes the Euclidean distance and f satisfies the Lipschitz property.

$$f(p) \leq f(q) + \|p - q\| \tag{3.6}$$

with $q \in S$. Obviously, the feature size can only be estimated if the medial surface can be approximated.

This concept of estimating the local feature size is only applicable for point samples representing closed surfaces [2] or curves. Generally, it is not transferable to unstructured point clouds in 3D space. Hence, we choose the local feature size to be specified, manually, by the user.

3.4.2 Moving Least Squares Surface Fitting

To identify feature lines within 3D point clouds, we first determine the corresponding point set surface S_p of a given point cloud $P \in \mathbb{R}^3$. This is a local procedure, accomplished by applying a method called *moving least squares* (MLS). The MLS method was introduced by Lancaster and Salkalukas [90] as an interpolation scheme for reconstructing continuous surfaces from a set of unorganized 3D point samples. The MLS

[2] By closed surfaces we refer to two-manifold surfaces having no boundaries.

approximation scheme starts with a weighted least squares problem description for an arbitrary point in $\mathbb{R}^3$ and then moves this point over the parameter domain, where a weighted least squares fit is computed for each location, individually.

To determine S_p we fit parametric surface patches to locally defined subsets of P. These subsets either are defined by the k-nearest neighbors of a point $p \in P$ or the neighbor points that lie within a predefined range of p. For the remainder, these subsets are denoted by $N(p)$.

The first step consists of finding a local reference plane $D = (n, c)$ for the points of $N(p)$. This plane is defined by

$$D = \{q \in \mathbb{R}^3 | \langle n, c - q \rangle = 0\}, \tag{3.7}$$

where n represents the plane's normal vector obtained using principal components analysis (PCA) and c denotes the centroid of $N(p)$. The alignment of the plane for a fixed c is determined by minimizing

$$\sum_{q \in N(p)} (\langle n, q - c \rangle)^2 \Theta(\|q - c\|). \tag{3.8}$$

according the vector n.

In the second step, the reference plane D is used to determine a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ that represents S_p. Let r denote the projection of the point $q \in N(p)$ onto D using local plane coordinates. Then, f is found by performing a least-squares fit to the height values $d = \langle n, p - r \rangle$ [3] at corresponding plane coordinates r. The residual to be minimized is given by

$$\sum_{q \in N(p)} (f(r) - d)^2 \Theta(\|p - r\|, r), \tag{3.9}$$

where

$$\Theta(d, h) = e^{-\frac{d^2}{h^2}} \ . \tag{3.10}$$

[3] The operator $d = \langle ., . \rangle$ denotes the scalar product.

The function Θ is a scaling function of Gaussian form, weakening the influence of distant points depending on h. In this definition h represents the anticipated spacing of points. The point set together with an adequately chosen value for h defines the surface S_p.

In case the S_p is a closed two-manifold sufficiently dense sampled, we can determine the LFS as described above and use it to derive a suitable value for h. However, considering unorganized point clouds, the computation of the LFS generally is not possible and thus h has to be determined indirectly. With respect to feature detection the value h depends on the type and the size of the features the user focuses and is chosen manually.

From these surface representations we are able to derive the information necessary to compute feature values measuring their strength as discussed in the following.

3.4.3 Detecting Feature Lines

Since feature lines are defined in terms of curvature measuring their strength by curvature is obvious. Hence, the robust approximation of the local curvature is essential. There are several approaches concerning the task of determining curvature estimates from point clouds [39], [63], [86]. The applied method is based on computing these values from biquadratic polynomials fitted via MLS to the point neighborhoods $\mathcal{N}(p)$ (see previous section 3.4.2). This approach delivers good estimates also in the presence of high noise since MLS naturally smooths out a certain amount of noise. Depending on the type of bivariate polynomials used, the quality of the curvature estimates varies.

There are different curvature measures that could be used to define the strength of surface related feature lines. Defined in terms of principal curvature values κ_{min} and κ_{max} the most important ones are:

- The maximum curvature, defined by

$$MAX = max(\kappa_{max}, \kappa_{min}). \tag{3.11}$$

- The Gaussian curvature, defined by

$$K = \kappa_{max} \cdot \kappa_{min}. \tag{3.12}$$

- The mean curvature, defined by

$$H = \frac{1}{2}(\kappa_{max} + \kappa_{min}). \tag{3.13}$$

- The root mean squared (RMS) curvature, defined by

$$RMS = \sqrt{\frac{1}{2}(\kappa_{max}^2 + \kappa_{min}^2)}. \tag{3.14}$$

One efficient way to determine the principal curvature values κ_{max} and κ_{min} is by computing it from the *Weingarten map* W [38], also known as shape operator S. The principle curvature values κ_{min} and κ_{max} and the principle curvature vectors are determined by the eigenvalues and eigenvectors of W. An example, for determining W, directly, from biquadratic polynomials is presented in appendix A.

To accomplish reliable feature detection the proposed method relies on a robust approximation of "feature strength", for this purpose the RMS curvature measure is used. Other curvature measures like mean, Gaussian or maximal curvature have been considered and tested, too. However, the RMS curvature was most appropriate and produced best results in the context of graph extraction since it provides a positive and continuous definition of feature strength allowing a better differentiation between features of the same type.

3.5 Point Classification

Performing feature extraction solely based on fitting surfaces and computing curvature values would not lead to reliable results since not every point structure can be approximated by a surface (cf. discontinuities, feature curves). To be able to detect features like feature lines, feature curves and critical regions, we have to distinguish between the different type of point structures associated with them.

The goal of the point classification is to decompose the point cloud P into subsets of points the different feature types can be associated with. We determine whether a point $p \in P$ corresponds to a feature curve, a surface structure or is part of a discontinuity and treated as a so-called critical point. Measuring the "shape" of the local point neighborhood allows us to distinguish between three principal structural types: spherical,

cylindrical and planar neighborhoods. Let $N(p)$ denote the point neighborhood e.g., specified by the number of points lying within a sphere centered at p and having radius r, or is defined by a predefined set of nearest neighbor points. The size of the neighborhood is specified by the user and depends on the size and the type of the features to be extracted. Determining the shape of $N(p)$ is accomplished by measuring the distribution of the points of $N(p)$ in each dimension by using the eigenvalues of the covariance matrix C of $N(p)$. Let $\lambda_0 \leq \lambda_1 \leq \lambda_2$ be the eigenvalues of C. It is a well-known fact that $N(p)$ has a spherical shape if $\lambda_0 \approx \lambda_1 \approx \lambda_2$, a planar (disc-like) if $\lambda_0 \ll \lambda_1 \wedge \lambda_1 \approx \lambda_2$ or a cylindrical shape if $\lambda_0 \approx \lambda_1 \wedge \lambda_1 \ll \lambda_2$. Using this information allows us to establish three point classes by defining the following sets:

$$\mathcal{P}_{cp} = \{p \in P | \lambda_0/\lambda_2 \geq \varepsilon\} \tag{3.15}$$

$$\mathcal{P}_{c} = \{p \in P | \lambda_1/\lambda_2 < \varepsilon\} \tag{3.16}$$

$$\mathcal{P}_{d} = \{p \in P | \lambda_0/\lambda_2 < \varepsilon\} \tag{3.17}$$

Here, $\mathcal{P}_{cp}, \mathcal{P}_c, \mathcal{P}_d$ define the points with spherical, cylindrical and planar (disc-like) neighborhood (with $\mathcal{P}_{cp} \cap \mathcal{P}_c = \emptyset$, $\mathcal{P}_{cp} \cap \mathcal{P}_d = \emptyset$ and $\mathcal{P}_c \subseteq \mathcal{P}_d$). The variable ε is a feature-specific value and determines when a point becomes e.g., a critical point. Naturally, ε is chosen to be $\frac{1}{2}$ but it can be adapted to meet a user's needs. For example, reducing ε increases the sensitivity of a point for being classified as a critical point, causing some points of $\mathcal{P}_c$ and $\mathcal{P}_d$ to migrate to $\mathcal{P}_{cp}$. This effect can be geometrically interpreted as extending the boundaries around a discontinuity.

This form of point differentiation implies that feature curves associated with cylindrically shaped point neighborhoods are part of surfaces point structures($\mathcal{P}_c \subseteq \mathcal{P}_d$). This is conform to the definition of feature curves. In some sense, feature curves represent feature lines, but due to noise and undersampling of the corresponding points, measuring their strength by curvature is hard and would not provide reliable results. Thus, the shape class $\mathcal{P}_c$ has been introduced which allows to apply methods for measuring the feature strength more reliably.

Points that lie within critical regions and thus are not associated to a surface or a feature curve are called *critical points*. This point type represent junctions of features lines (e.g., at corners) or intersections between surfaces and/or feature curves. In general they are not associated with curve-like point structures, but are used in the context of

graph extraction (sec. 3.7) to represent the underlying discontinuities.

The feature values measuring the strength of the different features are defined by

$$f_{cp} = \lambda_0\lambda_1/(\lambda_2)^2 \tag{3.18}$$

$$f_c = (\lambda_2 - \lambda_1)\lambda_0/(\lambda_2\lambda_1) \tag{3.19}$$

$$f_d = \kappa_{RMS} \tag{3.20}$$

where κ_{RMS} represents the RMS curvature. The terms $f_{cp}(p), f_c(p)$ and $f_d(p)$ denote the formulas for computing the feature values for the points of the classes $\mathcal{P}_{cp}, \mathcal{P}_c$ and $\mathcal{P}_d$. The values f_{cp}, f_c increase when the shape of $N(p)$ approaches the ideal form of a sphere, a line and decrease otherwise, whereas f_d increases as the curvature increases.

However, the proposed point classification scheme heavily depends on the size of the local neighborhood $N(p)$ and would not provide a reliable shape evaluation in presence of considerable noise. To enhance the stability a probability based point classification approach is applied.

3.6 Probability-Based Point Classification

The idea of the probability-based classification is to evaluate the shape of the neighborhood $N(p)$ of a given point p at multiple scales. We allow $N(p)$ to vary in size by a given percentage of its original magnitude. Then, the probability estimates are computed by measuring the number of times $N(p)$ can be associated with one of the classes defined by (3.15), (3.16), (3.17). For the following let $N^{\delta^+}(p)$ and $N^{\delta^-}(p)$ denote the considered maximum and minimum point neighborhoods of p with $N^{\delta}(p) = \{q \in P |\ \|q - p\| < \delta\}$. By evaluating

$$\mathcal{L}_{cp,c,d} = \frac{1}{\delta^+ - \delta^-} \int_{\delta^-}^{\delta^+} \varphi^t_{cp,c,d}(p)dt \tag{3.21}$$

where

$$\varphi_{cp}^t(p) = \begin{cases} 1 & \text{if } \lambda_0^t \geq \varepsilon\lambda_2^t, \\ 0 & \text{else} \end{cases}, \tag{3.22}$$

$$\varphi_{c}^t(p) = \begin{cases} 1 & \text{if } \lambda_1^t < \varepsilon\lambda_2^t, \\ 0 & \text{else} \end{cases},$$

$$\varphi_{d}^t(p) = \begin{cases} 1 & \text{if } \lambda_0^t < \varepsilon\lambda_2^t, \\ 0 & \text{else} \end{cases}.$$

Here, λ_i^t are the eigenvalues of C^t of $N(p)^t$. $\mathcal{L}_{cp}, \mathcal{L}_c$ and $\mathcal{L}_d$ define the likelihood of point p to correspond to one of the shape classes defined by $\mathcal{P}_{cp}, \mathcal{P}_c$ and $\mathcal{P}_d$.

The multi-scale principle can also be applied to derive more stable feature values. The computation of I_{cp}, I_c and I_d then is performed by

$$I_{cp,c,d}(p) = \frac{1}{h\sqrt{2\pi}} \int_{\delta_-}^{\delta^+} f_{cp,c,d}^t(p) \cdot e^{-\frac{\|p-t\|^2}{2h^2}} dt, \tag{3.23}$$

$$f_{cp}^t = \lambda_0^t \lambda_1^t / (\lambda_2^t)^2 \tag{3.24}$$

$$f_{c}^t = (\lambda_2^t - \lambda_1^t)\lambda_0^t / (\lambda_2^t \lambda_1^t) \tag{3.25}$$

$$f_{d}^t = \kappa_{RMS}^t \tag{3.26}$$

where κ_{RMS}^t denotes the RMS curvature of polynomial patches at p fitted to $N^t(p)$ and $h = \frac{1}{2}(\delta^+ - \delta^-)$. The feature values $I_{cp}(p), I_c(p)$ and $I_d(p)$ now express the strength of a feature associated with point p at multiple scales.

3.6.1 Examples

In this section we present some examples obtained by applying the proposed point classification method. The examples of Figure 3.1 depict the computed probability values in form of color values which are projected onto the individual points of the original point cloud. Figure 3.1a shows the original point cloud representing a distant water tower at a low resolution level. The data set, at its highest resolution, consists of about 1.7 million sample points (average sample resolution is approximately 0.05 meters). It was acquired by using a terrestrial tripod-based LiDaR scanner. Figure 3.1b, 3.1c and 3.1d show the projected probability values $\mathcal{L}_c$, $\mathcal{L}_d$ and $\mathcal{L}_{cp}$ represented as color values reaching from blue (low probability) to red (high probability).

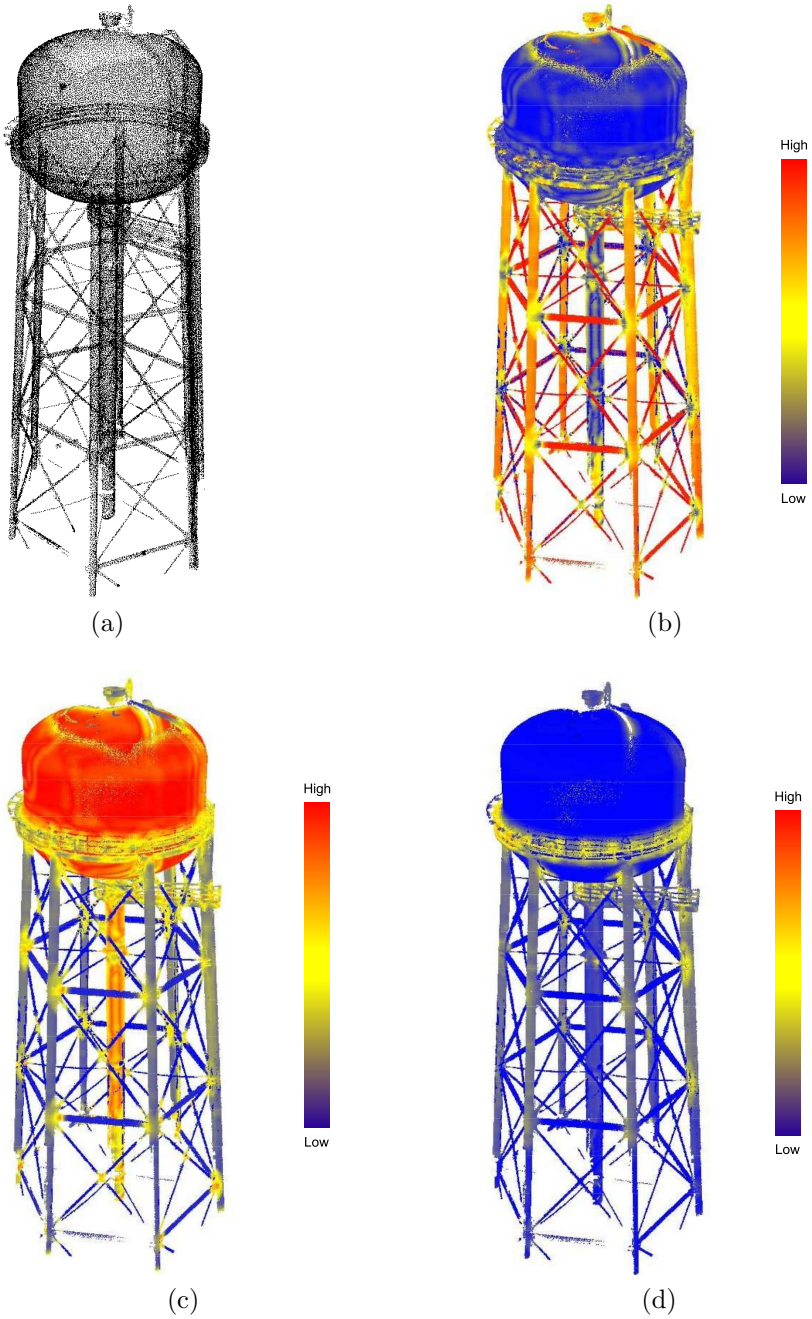


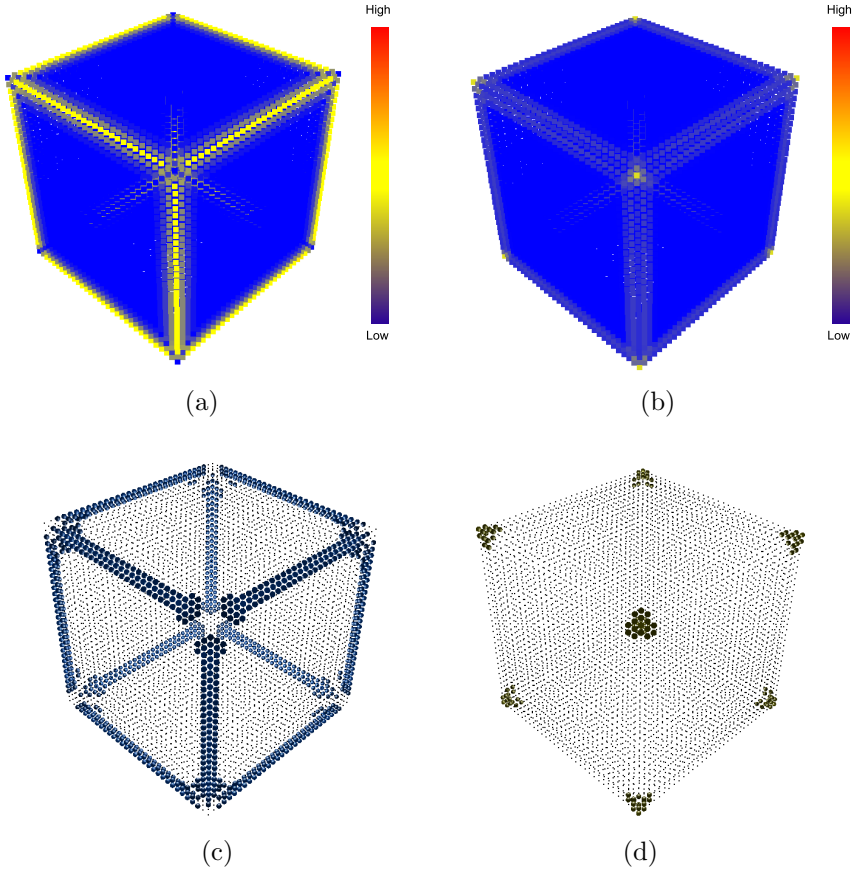
Figure 3.1: (a) original point cloud of a scanned water tower consisting of about 1.7 million points. (b) color coded probability values (blue means low, red means high probability) $\mathcal{L}_c$ projected onto the individual points. (c)-(d) visualization of the probability values $\mathcal{L}_d$ and $\mathcal{L}_{cp}$ (again blue is low, red is high probability).

3.7 Extraction of Feature Graphs

The extraction process requires to differentiate between the different types of features. For that purpose we need to find a conformal basis for comparing the feature strength of feature lines and feature curves with each other. However, this is hard to accomplish, because of the fundamental dissimilarity between the associated features. Thus, using the definition of feature strength of section 3.6 and 3.5 does not allow to compare the different features with each other. It only allows to distinguish between features of the same type. Hence, constructing one graph representation incorporating the different features at the same time is not possible. We solve this problem by establishing two independent graphs, one representing the feature lines, another representing feature curves. Critical regions are incorporated into the graph structure in form of *critical nodes*.

The structural features are detected and extracted following the idea of [34], [118] and [138] of identifying initial seed nodes and performing an edge-growing approach to connect related nodes. The seed nodes for a graph are represented by all points having a minimal type probability. Establishing connections between seed nodes such that the resulting graph reflects the trend of underlying feature lines and feature curves is difficult. Most methods start by connecting seed nodes using the criteria of proximity. However, this can cause a variety of branches inducing perturbations in further propagation. Therefore, controlling the propagation direction plays an important role.

Several extraction methods determine the derivatives of extracted feature values, like curvature, to identify and follow the direction of its minimal descend. Since these values vary considerably e.g., due to noise, this differential approach would lead to poor results. Instead, we define the propagation direction by applying PCA to a set of surrounding nodes. This approach is more stable assuming that the local environment of the actual node is of sufficient size. In contrast, critical nodes are not subject to this graph extraction procedure. Since they represent structural discontinuities, they are used to recover junctions of feature lines and feature curves. The final step consists of graph simplification aiming at a improving the visual representation.



(a) (b)

(c) (d)

Figure 3.2: (a)-(b) visualization of probability values from low (blue) to high (red) for $\mathcal{L}_c$ (a) and $\mathcal{L}_{cp}$ (b). (c) depicts the identified feature curve seed nodes V_c (blue points), (d) shows the identified critical seed nodes CV_c (black points).

3.7.1 Selecting Seed Nodes

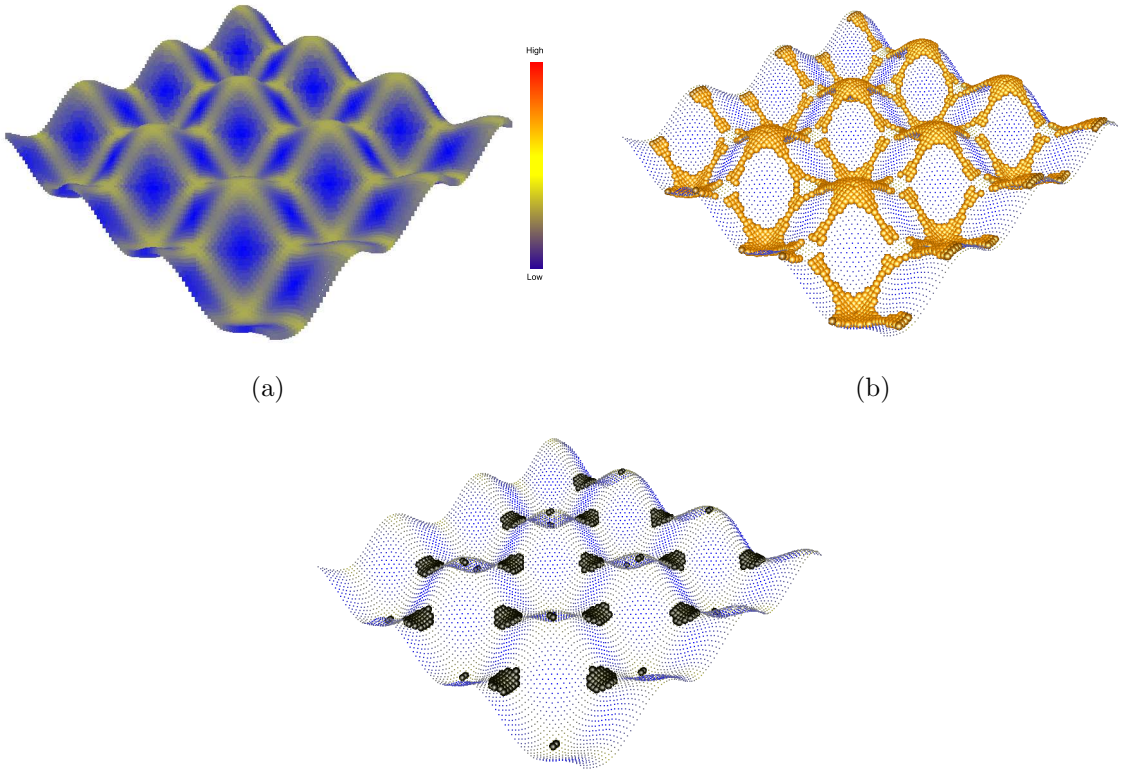
A graph $\mathcal{G} = (V, E)$ formally consists of a set of nodes V and a set of edges $E \subseteq V \times V$ connecting the nodes of V. The *seed nodes*, defining the basis of the individual graphs, are identified by using the probability values assigned to the individual points (see section 3.6). The initial graph for representing feature curve related point structures is defined by $\mathcal{G}_c^0 = (V_c, \emptyset)$ with

$$V_c = \{p \in P | \mathcal{L}_c(p) \geq \mathcal{L}_d(p) \wedge \mathcal{L}_c(p) > \mathcal{L}_{cp}(p)\}. \tag{3.27}$$

The corresponding set of *critical seed nodes* V_{cp} is defined by

$$CV_c = \{p \in P | \mathcal{L}_{cp}(p) \geq \mathcal{L}_d(p) \wedge \mathcal{L}_{cp}(p) \geq \mathcal{L}_c(p)\}. \tag{3.28}$$

Figure 3.2 shows an example regarding a simple test cube data set. Figure 3.2a and Figure 3.2b show the color coded points representing probability values from low (blue color) to high (red color) for $\mathcal{L}_c$ and $\mathcal{L}_{cp}$. The identified feature curve seed nodes and critical seed nodes are shown in Figure 3.2c as blue points and 3.2d as black points.



(a) (b)

(c)

Figure 3.3: (a) visualization of the computed probability values in form of color values mapped to the individual points reaching from blue (low probability) to red (high probability), for $\mathcal{L}_d$. (b) the extracted feature line related seed nodes V_c as yellow points and (c) the corresponding critical seed nodes CV_d as black points.

The seed nodes for the surface-related feature lines are set up differently. The user specifies additional curvature-related thresholds σ_{min} and σ_{max} restricting the features to be extracted with respect to their strength, balancing critical and feature-specific nodes among the points of $\mathcal{P}_d$. For example, the threshold σ_{min} is responsible for separating features like creases and non-creases appearing on a surface. The parameter σ_{max} controls the transition between creases and corners. The initial graph representing surface-related features is $\mathcal{G}_d^0 = (V_d, \emptyset)$ with

$$V_d = \{p \in P | \mathcal{L}_d(p) > \mathcal{L}_{cp}(p) \wedge \mathcal{L}_d(p) > \mathcal{L}_c(p) \wedge \sigma_{min} < I_d(p) \leq \sigma_{max}\}. \tag{3.29}$$

The critical nodes corresponding to $\mathcal{G}_d^0$ are defined as

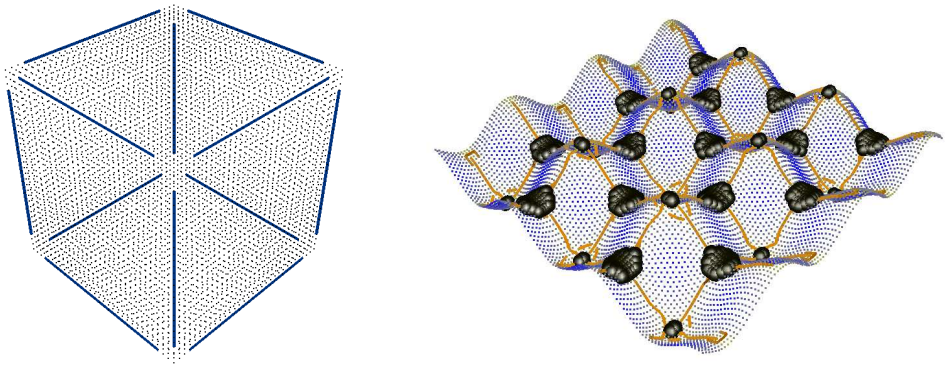
$$CV_d = \{p \in P | \mathcal{L}_{cp}(p) \geq \mathcal{L}_c(p) \wedge \sigma_{max} < I_d(p)\}. \tag{3.30}$$

Figure 3.3 illustrates the extraction of seed nodes associated with surface-related feature lines from a simple test sheet data set. Figure 3.3a shows the color encoded point cloud with respect to the probability values $\mathcal{L}_d$. Figure 3.3b depicts the extracted feature line seed nodes V_d, Figure 3.3c shows the corresponding critical seed nodes CV_d.

In order to obtain a connected graph structure for $\mathcal{G}_c^0$ and $\mathcal{G}_d^0$ we have to establish edges between the seed nodes. The edge propagation step, discussed in the next section, produces a first approximation of the underlying feature lines and -curves.

3.7.2 Edge Propagation

The edge propagation concerns the process of connecting two nodes $v, w \in V$ of a graph $\mathcal{G} = (V, E)$ by an edge $e = (v, w)$. The propagation is performed only among the seed nodes of V_c or V_d. The critical nodes are incorporated by the subsequent step. For the following let us assume V and CV either represent V_c and CV_c, or V_d and CV_d. Principally, each seed node $v \in V$ is allowed to propagate into two directions $\vec{d}$ and $-\vec{d}$. We define $\vec{d}$ to be the principal direction of the propagation neighborhood $\mathcal{M}_r(v)$. The set $\mathcal{M}_r(v)$ is defined by



(a) (b)

Figure 3.4: (a) graph $\mathcal{G}_c$ for the seed nodes of the cube example after edge propagation, (b) graph $\mathcal{G}_d$ for the curved sheet example data set after propagation together with the identified critical seed nodes (black).

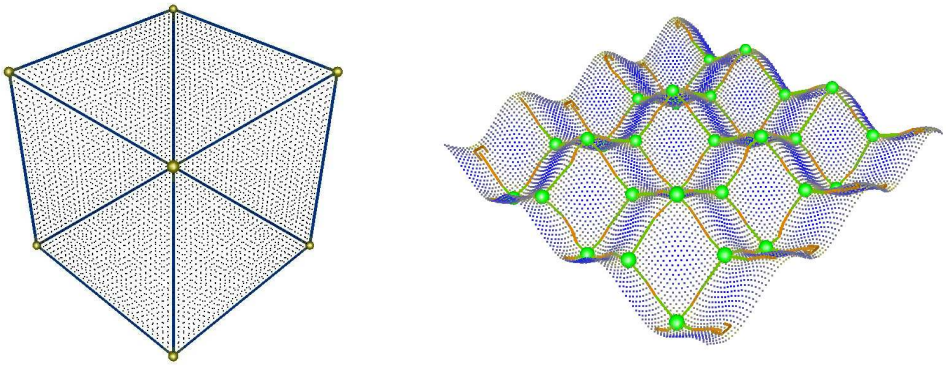
$$\mathcal{M}_r(v) = \{w \in V \cup CV | \|\vec{v} - \vec{w}\| \leq r\} \tag{3.31}$$

with r representing the user-specified LFS. Then, the vector $\vec{d}$ is the eigenvector with the largest eigenvalue of the covariance matrix of $\mathcal{M}_r(v)$. Provided that no other node $u \in V$ is connected to v, with $\vec{d} \cdot (\frac{\vec{u}-\vec{v}}{|\vec{u}-\vec{v}|}) > 0$, v is connected to $w \in \mathcal{M}_r(v)$, where

$$w = \min_{w \in \mathcal{M}_{r,\vec{d}}} (v) \left(\|proj_{\vec{d}}(\vec{w} - \vec{v}) \cdot \vec{d} + \vec{v} - \vec{w}\| / \|\vec{w} - \vec{v}\| \right). \tag{3.32}$$

The nodes $w \in \mathcal{M}_{r,\vec{d}}(v) \subseteq \mathcal{M}_r(v)$ satisfy the inequality $\left(\frac{\vec{w}-\vec{v}}{\|\vec{w}-\vec{v}\|}\right) \cdot \vec{d} > 0$. To prevent unwanted edges, all disconnected nodes $w \in \mathcal{M}_r(v)$ lying closer to v than the most distant connected neighbor are forbidden to propagate. Moreover, connections establishing acute angles with $\vec{d}$ are forbidden. All remaining nodes not connected by an edge are deleted.

Examples of resulting graphs are shown on Figure 3.4a and Figure 3.4b. Rather than using a minimal spanning graph as proposed by Pauly et al. [118], this approach provides greater consistency between edges and the locus of the underlying features.



(a) (b)

Figure 3.5: (a) final graph structure representing the extracted feature curves from the cube test data set (blue lines are feature curves, green points are the extracted critical nodes), (b) final graph representing the surface-related feature lines of the curved sheet test data set (brown lines are feature lines, green points are critical nodes).

3.7.3 Connecting Critical Nodes

In this step the critical nodes CV are connected to the already established edges of $\mathcal{G}$. Assume CV to be the set of critical nodes, either defined by the set (3.28) or the set (3.30). To recover junctions among feature lines or curve-lines the nodes of CV have to be connected or merged with nodes of $\mathcal{G}$.

We start clustering the nodes of CV by grouping neighbored critical nodes. The resulting clusters are defined by $\mathcal{D} = \{\dots, \mathcal{D}_i, \cdots\}$ with $\mathcal{D}_i = \{w \in CV | \quad v \in \mathcal{D}_i \wedge \|\vec{v} - \vec{w}\| < LFS\}$ and $D_i \cap \mathcal{D}_j = \emptyset$ for $i \neq j$. The initial cluster set is given by $\mathcal{D}^0 = \{\cdots, \{v\}, \{w\}, \cdots\}$ with $v, w \in V$ and $v \neq w$.

We proceed determining the central nodes of each cluster $\mathcal{D}_i$. First, we divide each cluster $\mathcal{D}_i$ into n disjoint sets $\mathcal{D}_i = \bigcup_j^n \mathcal{C}_j$ fulfilling the following conditions:

1. Each subset $\mathcal{C}_j \subseteq \mathcal{D}_i$ has a central node $c_j \in \mathcal{C}_j$ with $c_j = \max_{c_j \in \mathcal{C}_j}\{I(c_j)\}$. The value $I(c_j)$ represents the feature value associated with the node/point c_j.

1. For each central node $c_j \in \mathcal{C}_j$ and $c_k \in \mathcal{C}_k$, with $j \neq k$, the condition $LFS \leq \|\vec{c_j} - \vec{c_k}\|$ holds.

2. For each subset $\mathcal{C}_j$ it is $\|c - c_j\| < LFS$, where c_j is the central node of $\mathcal{C}_j$ and $c \in \mathcal{C}_j$ with $c \neq c_j$.

We identify these subsets by iteratively performing the following steps: First, we sort the nodes $v \in \mathcal{D}_i$ according their feature values in descending order. Using the seed node v having the highest feature value, we set up $\mathcal{C}_j = \{w \in \mathcal{D}_i \mid \|\vec{v} - \vec{w}\| < LFS\}$. In the last step the nodes of $\mathcal{C}_j$ are removed from $\mathcal{D}_i$. These steps are iterated until $\mathcal{D}_i = \emptyset$.

Next, we connect all nodes $v \in V$ to its surrounding critical nodes $w \in CV$ with $\|\vec{v} - \vec{w}\| < LFS$. If a node $v \in V$ is connected to a cluster node $w \in \mathcal{D}_i$, v is reconnected to the corresponding central node c_j, with $c_j \in \mathcal{C}_j$ and $w \in \mathcal{C}_j$.

To obtain the final graph representation, we delete all unconnected critical nodes. Figure 3.5a and Figure 3.5b show the final graphs for the cube and the curved sheet data set.

3.7.4 Graph Simplification

We note that $\mathcal{G}_c$ and $\mathcal{G}_d$ possesses certain unwanted structures like short branches, loops etc. To get rid of these artifacts the following simplification framework is applied. Since most of these simplification steps are common in the field of graph processing the discussion is kept informal.

Collapsing Edges

In order to reduce complexity of the extracted graphs we provide the user with the option to collapse edges whose length is smaller than a predefined parameter $r \in \mathbb{R}^3$. Given an edge $e = (v, w) \in E$ of an graph $\mathcal{G}$, an edge collapse is performed by substituting the nodes v and w by a new node v' and removing e from E. Collapsing an edge e is allowed, if the following criteria are fulfilled:

a) $\|\vec{v} - \vec{w}\| < r$

b) $(v, w) \in E$ and $\nu(v) > 1$ and $\nu(w) > 1$

c) v or w is a non-critical node

Here, $\nu(v)$ defines the number of edges incident to v. Since $\mathcal{G}$ is intended to reflect the underlying feature lines and -curves we force the new introduced vertex v' to assume the position of the node v, if the feature value assigned to v is greater than the feature value assigned to w. Otherwise v' assumes the position of w. In case one of the nodes w or v is a critical node the new node v' assumes the position of the critical node.

Pruning Edges

The remaining graph still exhibits features irrelevant or unwanted for visual exploration. To reduce visual clutter and direct the user to the regions of interest we prune small branches by cutting off all edges $e = (v, w) \in E$ of $\mathcal{G}$, where

- $\|\vec{v} - \vec{w}\| < r$ and
- $\nu(v) = 1 \wedge \nu(w) \neq 2$.

Graph Smoothing

Finally, we enhance the visual appearance of the final graph set $\mathcal{G}$ by applying a graph smoothing step. This step is performed for each node $v \in V$ of $\mathcal{G}$ by linearly interpolating between v and all nodes $w \in V$ incident to v. The new position $\vec{v'}$ for a node v is defined by

$$\vec{v'} = \frac{1}{2}\vec{v} + \frac{1}{|V_e|} \sum_{w \in V_e} \vec{w} \tag{3.33}$$

where V_e denotes the set of nodes connected to v by an edge. The smoothing is not applied to critical nodes or nodes v with $\nu(v) = 1$ or $\nu(v) \geq 3$ (to prevent $\mathcal{G}$ from shrinking).

3.7.5 Results

The introduced method provides a highly effective mean for exploring the underlying structural composition of environmental LiDaR point data sets. The graph-based approach, driven by a few predefined input parameters, extracts and represents three principle feature types in an efficient and reliable way. Most important parameters are the LFS, and σ_{min} and σ_{max} confining the type of the considered surface-related features. The LFS has to be chosen in a way that noise does not significantly interfere with the classification process but also that the extraction of structural information is maintained. Generally, σ_{min} and σ_{max} are less crucial to determine. Once the LFS is found the remaining feature extraction can be performed rather fast in an interactive

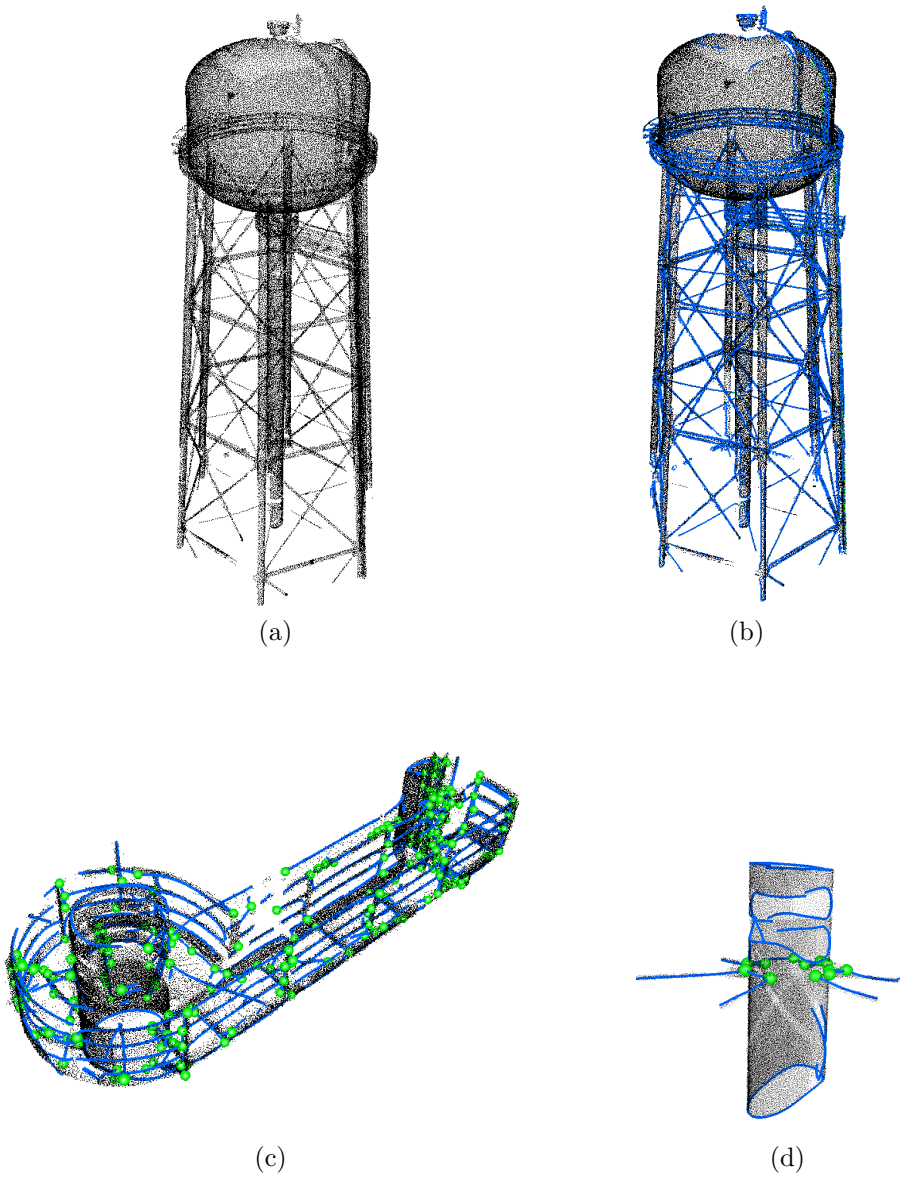
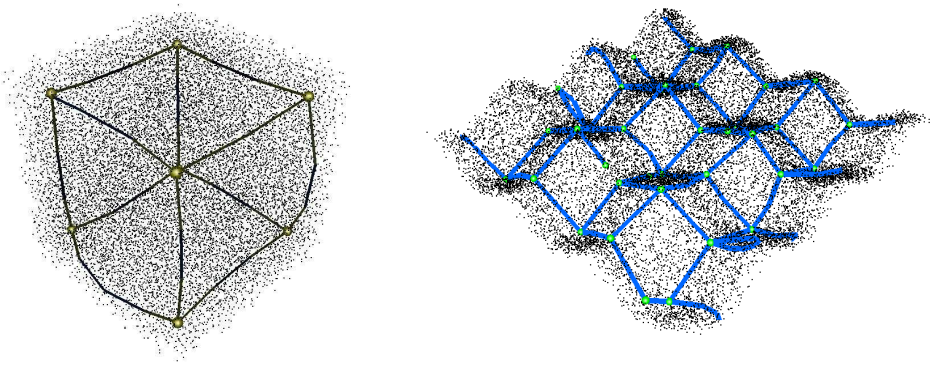


Figure 3.6: (a) the original point cloud of a sampled water tower structure consisting of approximately 1.7 million points, (b) extracted critical nodes and feature curves depicted as green points and blue lines. (c) the extracted feature curves and critical nodes extracted from a complex part of the water tower located at the inside. (d) corresponding feature curves and critical nodes for a crosspiece located at the main pole of the water tower.

manner. The method distinguishes features of three different types and allows classification of points in terms of probability. Moreover, it allows to identify junctions of feature

lines or curve-lines not only by means of combinatorial graph processing it rather incorporates point cloud inherent structural information. The structural discontinuities identified, are captured and represented by the final graph structure in form of critical nodes.



(a) (b)

Figure 3.7: (a) final graph structure representing the extracted feature curves (black lines) and critical regions (green points) from the cube test data set, afflicted with 5% noise, (b) shows the final graph representing the surface-related feature lines (blue lines) and critical regions (green points) extracted from the test sheet data set, afflicted with 5 % noise.

The multi-scale approach provides reliable results up to a noise level of 5-8%. Figure 3.5a shows the resulting graph related to feature curve structures associated with the test cube data set without noise, Figure 3.7a shows the resulting graph captured from the noisy test cube data set afflicted with 5% noise. Figure 3.5b and Figure 3.7b show the feature-graphs extracted from the curved sheet data set without and with 5% noise. The added noise is uniformly distributed in each dimension and is measured as percentage of the length of the diagonal of the smallest bounding box enclosing the given point cloud. The examples demonstrate that all resulting graphs of the test data sets reflect the correct underlying topological conditions.

The method was applied to several environmental data sets mainly collected using a LiDaR scanner mounted on a tripod. For example, Figure 3.6a to Figure 3.6d show the results of processing rather complex point clouds. In Figure 3.6a we see the original point cloud of a water tower scan consisting of approximately 1.7 million points. The extracted feature curve related graph is presented in Figure 3.6b. More detailed parts of the graph are presented in Figure 3.6c and Figure 3.6d. Figure 3.6c shows a relative complex

undersampled substructure of the tower. In this example the underlying feature curves (blue lines) and discontinuities (green points) have been almost completely recovered. The last object depicted by Figure 3.6d is a crosspiece of the main pole of the lower part of the tower having mounted stiffeners. The extracted feature graph exhibits critical nodes colored in green representing discontinuities between surface- and curve-like structures.

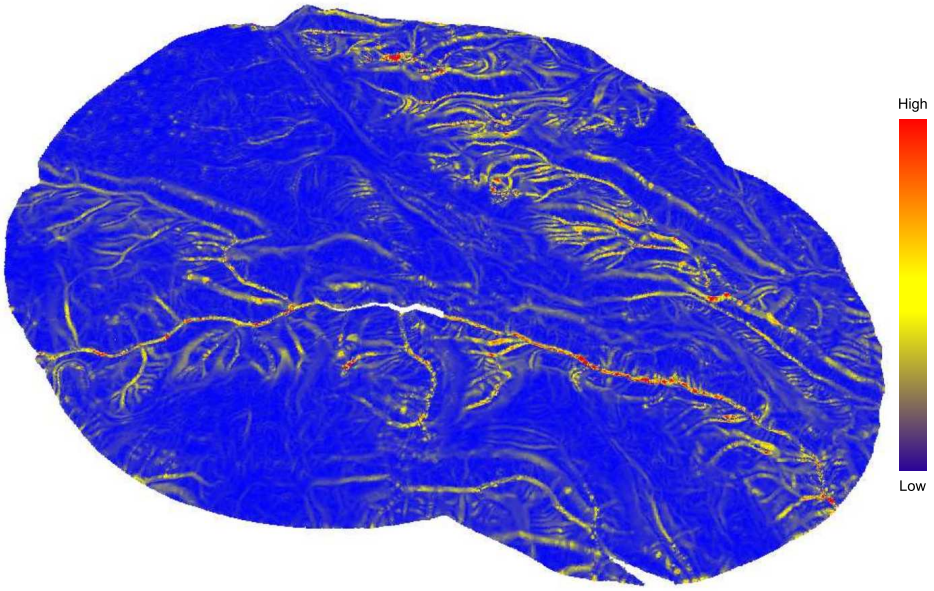
The environmental point cloud shown in Figure 3.8 represents a scan of the San Andreas Fault in southern California, USA. Since the scan was performed airborne, the acquired point cloud captures a variety of different artifacts and is afflicted with a relatively high amount of noise. Figure 3.8a depicts the computed curvature estimates, the subsequent feature line extraction process is based on. The curvature values reach from low (blue colors) to high (red colors). Figure 3.8b depicts the corresponding surface-related feature graph highlighting ridges of the landscape. The critical nodes, depicted as green points, represent underlying discontinuities or features having curvature values which exceed the user-specified limit σ_{max}. These critical nodes are associated with visually prominent landmarks like mountain peaks, tapered ridges or acute hills. The parameter values were chosen to capture ridges associated with relatively high curvature values. This example was produced using $\sigma_{min} = 0.2$, $\sigma_{max} = 1.5$ and $LFS \approx 0.0046\%$ (again measured in percentage of the length of the diagonal of the smallest bounding box containing the entire point cloud P).

Since determining likelihood and feature values is computationally expensive the point classification step is decoupled from the actual graph extraction process and performed in a pre-processing step. This allows the user to extract feature graphs in a nearly interactive manner. To accomplish the pre-processing more efficiently, the process is multi-threaded. The time complexity for the point classification is estimated to be $O(k^2 \cdot n \cdot log(n))$ (with n number of input points, k the average number of points the LFS neighborhood contains). The complexity of the graph extraction is assumed to be $O(m \cdot log(m) \cdot n)$ (with m being the number of identified seed nodes). This has been validated by our observations. The pre-processing of the water tower data set (see Figure 3.1), consisting of about 1.7 million points at the finest resolution level, using an Intel Core 2 Duo processor, was completed after 3 hours and 29 minutes. The subsequent graph extraction was accomplished after approximately 116 seconds. Performing the point classification for the geographic data set in Figure 3.8 was completed after 72 minutes. The final graph was constructed after 29 secs. The computational overhead primarily is caused by reloading required point data from disc.

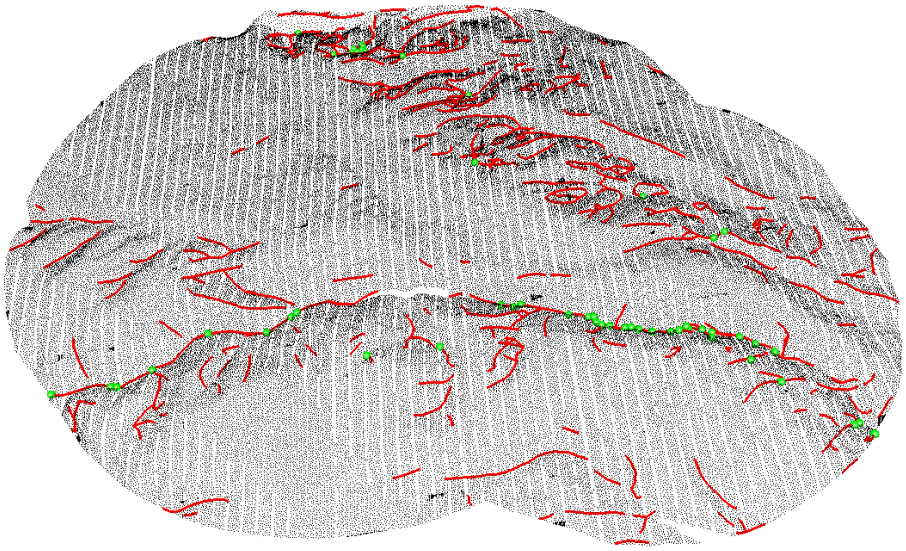
3.8 Conclusions and Possible Future Work

This chapter introduces a novel approach for extracting and visualizing features from LiDaR data relying on a probability-based point classification scheme. This classification allows to distinguish between points by comparing the principle shapes of the immediate point neighborhoods. The generated feature graphs are established in a way that they represent surface-related features like ridges and creases as well as curve-related point structures and discontinuities. The method was designed as an interactive, user-controlled method, also offering a high degree of automation. The provided examples demonstrate the benefits for measuring, identifying and extracting feature from unorganized noisy point clouds. One specific strength is the ability for a user to influence the extraction process and obtain instant visual feedback.

This work is fundamental for future research, e.g. to detect and measure deformation of environmental structures such as buildings and bridges exposed to terrestrial influences. Future work also may be directed toward feature-based segmentation of terrestrial data classifying objects of different type, such as trees, buildings, streets, etc. Another goal is to be able to extract topological information from landscapes by extracting and comparing feature lines from multiple scans, taken at different times, to identify deformations by comparative visualization methods.



(a)



(b)

Figure 3.8: (a) geological data set showing a part of the San Andreas Fault in southern California, USA, having a length of 6 kilometers. The data set consists of approximately 600,000 points. The points are colored with subject to the computed curvature values from blue (low curvature) to red (high curvature). (b) resulting feature graph approximating the underlying surface-related feature lines, depicted in red. The discontinuities are represented by green nodes.

4 Surface Reconstruction

4.1 Overview

Surface reconstruction from unorganized point data sets in 3D space is a very complicated and challenging problem. In most cases, no additional information is provided or included that can be used describing the connectivity or topology of surface features in point data. Depending on the characteristics of the sampled objects (e.g., buildings, mechanical parts, environments, etc.) and the conditions under which a scan is performed (e.g., scan noise, scan artifacts, overlaps) the generated point clouds may become noisy and highly complex. These problems motivate the effort of finding new methods providing more accurate and reliable reconstructions. Thus, a desirable reconstruction algorithm should be able to deal with complicated, noisy and non-uniformly distributed point samples and should provide a reliable and a smooth approximation of the original surface. As well, the reconstructed surface models should have a data representation supporting further processing e.g., animation, photo realistic rendering, or deformation.

This chapter is intended to provide the reader with an insight into the problems of surface reconstruction from unstructured 3D point clouds. It also introduces new methods regarding surface reconstruction from complex 3D point data as it primarily emerges in the field of reverse engineering or geoscience. The basic contributions of this chapter are:

- The introduction of a new octree-based method for surface reconstruction from unorganized 3D point data. The proposed approach is capable of generating multiresolution surface representations in a fast and accurate way. Depending on the chosen refinement level the reconstructed surface model automatically adapts to the level specific topological conditions. The reconstructed surfaces are of high

quality. The application of the method is intended for moderately sized point clouds of objects consisting of two-manifold surfaces with boundaries.

- In order to be able to perform reliable surface reconstruction for data sets having high complexity a new method following a volumetric reconstruction approach is discussed. The proposed method adapts the reconstruction to the topology of the point cloud intrinsic structures by differentiating between surface and non-surface related point structures. This approach enables to apply surface reconstruction to very complex and noisy data sets.

The chapter is organized as follows: Section 4.2 gives a brief overview of the challenges associated with surface reconstruction. We review some fundamental concepts concerning surface representations and provide a formal introduction to the surface reconstruction problem. Related work is presented in section 4.3. The new octree-based multi-resolution surface reconstruction methods is introduced in Section 4.4. The implicit feature-based reconstruction approach is presented in Section 4.5. Each method is discussed in detail and results are provided, proofing its correctness and applicability. Section 4.7 provides a conclusion and discusses possible directions for future work.

4.2 Basics

4.2.1 Surfaces

In general, a *surface* S is known to be a compact, orientable, 2D manifold embedded in 3D space. A surface is said to be a *closed surface* if the manifold has a boundary, otherwise it is known to be an *open surface*. In general there are two types of surfaces: Explicit and implicit surfaces; Explicit surfaces define the precise location of a surface in 3D space mostly by using simplicial representations like triangular meshes or smooth definitions based on parametric equations, called *parametric surfaces*. In contrast, implicit surface representations describe a surface as a particular iso-contour of a corresponding scalar function. The subsequent sections gives a brief overview about the two basic concepts.

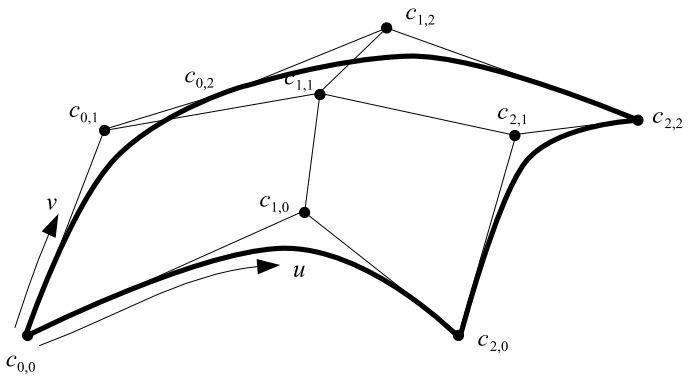


Figure 4.1: Principle configuration of a quadratic B-spline surface patch with the corresponding coefficient grid.

4.2.2 Parametric Surfaces

The most common way of defining a surface S is by using *parametric surface* representations. A parametric surface is a surface in 3D Euclidean space defined by a bivariate parametric equation of the form

$$f : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \tag{4.1}$$

The most important surface representations in the field of geometric modeling are polynomial- and Bézier-/B-Spline surfaces.

Polynomial Surfaces

Polynomial surfaces or surface patches can be derived from *polynomial curves*. A polynomial curve of degree k is defined by the sum of multiple weighted monoms:

$$f(u) = c_k u^k + c_{k-1} u^{k-1} + \cdots + c_0 \tag{4.2}$$

where $c_k \in \mathbb{R}^3$ are the real valued coefficients of f and $u \in [u_0, u_n]$ with $u_0 \leq u_n$ is called the domain parameter. The interval specifies the support region within the polynomial is defined. A polynomial surface patch of degree k then is defined by extending the expression 4.2 to

$$f(u,v) = \sum_{i=0}^{k} \sum_{j=0}^{l} c_{i,j} u^i v^j, \tag{4.3}$$

where $k \geq l$, and $c_{i,j} \in \mathbb{R}^3$ are the coefficients. This form of surface representation is widely used in the fields of computer graphics. Usually, they are defined in a way that the resulting surface approximates or interpolates a set of sample points in 3D space. However, this process is difficult and inefficient. Changing one coefficient affects the entire polynomial which requires the readjustment of all others. Another drawback is that polynomial surfaces become unstable in cases the sample points are inappropriately distributed. Other surface models based on Bézier and B-Splines provide a solution which allows to overcome these disadvantages and provide more flexibility in the modeling process.

Bézier and B-Spline Surfaces

The most important form of parametric surfaces are so called Bézier and B-Spline surfaces. These surfaces are the generalization of Bézier and B-Spline curves. Both, Bézier and B-Spline curves of degree k can be written as

$$f(u) = \sum_{i=0}^{n} c_i \phi_i^k(u) \tag{4.4}$$

where the $c_i \in \mathbb{R}^3$ are the *control points* or *base points*, and $\phi_i^k(u)$ are the *basis functions*. For Bézier curves, where $n = k$, the basis functions are called *Bernstein polynomials* and are defined by

$$\phi_i^k(u) = B_i^k(u) = \binom{k}{i} (1-u)^{k-i} u^i, \tag{4.5}$$

where $u \in [0,1]$. For B-spline curves, the basis functions are defined recursively by

$$\phi_i^k(u) = N_i^k(u) = \frac{u - u_i}{u_{i+k} - u_i} N_i^{k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} N_{i+1}^{k-1}(u) \tag{4.6}$$

$$N_i^0(u) = \begin{cases} 1 & if \quad u_i \leq u < u_{i+1} \\ 0 & else \end{cases} \tag{4.7}$$

where $u \in [u_{k-1}, u_n]$, and $U = \{u_0, u_1, \cdots u_{n+k+1}\}$ with $u_0 \leq u_1 \leq \cdots \leq u_{n+k+1}$ is the so called *knot vector*. B-splines curves are the generalization of Bézier curves, if $n = k$ holds, the basis functions $N_i^k(u)$ become equal to $B_i^k(u)$.
To describe *spline surfaces*, the expression 4.4 can be generalized into

$$f(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} c_{i,j} \psi_{i,j}^k(u, v) \tag{4.8}$$

The most common form of spline surfaces are so called *tensor product* surfaces whose basis functions are the product of two of the familiar one-dimensional Bernstein or B-spline basis functions. For example, a tensor product B-spline surface of degree k is defined by

$$f(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} c_{i,j} N_i^k(u) N_j^k(v) \tag{4.9}$$

An example of such a B-Spline surface patch is presented by Figure 4.1. B-spline surfaces in computer graphics, especially in geometric modeling, are popular and widely-used, amongst others due to the following properties:

- In contrast to polynomial surface patches, the surface shape is directly related to the control points (convex hull property).
- Depending on the degree and the number of control points, control point modifications only affect the surface locally.
- The quality of the final surfaces representation can be adapted by introducing or removing control points.

A more detailed insight into parametric surface models like polynomial-, Bézier- and B-spline surfaces is provided by [43].

4.2.3 Implicit Surfaces

In contrast to parametric surfaces, where the surface coordinates are given as function values, the surface coordinates of an implicit surface are treated as functional arguments. Generally, implicit surfaces are defined by a real valued function of the form

$$f(p_x, p_y, p_z) = c. \tag{4.10}$$

The function f maps point coordinates $(p_x, p_y, p_z) = p \in \mathbb{R}^3$ to values $c \in \mathbb{R}^n$. For most applications $n = 1$, and c is a constant. The closure of points $p \in \mathbb{R}^3$ fulfilling equation 4.10 is called an *implicit surface* where f is the *implicit surface function* (also known as *scalar field* , *potential field*, or *potential function*). When $c = 0$, the set of points $M = \{p \in \mathbb{R}^3 | f(p) = 0\}$ is called *zero set* of f, the corresponding surface is termed *zero surface*.

The implicit surface functional f typically is specified either by discrete samples organized by 3D grids, mathematical functions or procedural methods. Discrete samples usually represent physical quantities (like density, temperature etc.), obtained from e.g., CT-/MRI scanner or generated by physical simulation algorithms. In this context, the surface described by f is termed *iso-surface*, the corresponding constant c is called *iso-value*. A popular way for extracting such iso-surfaces in form of simplicial meshes is provided by the so called Marching Cubes (MC) algorithm [97] (see section 4.2.8). Iso-surfaces are popular for scientific visualization (e.g., medical, fluid, or material data), especially in cases of varying iso-values c. A more detailed introduction to implicit surfaces is presented by [102].

Meta-Surfaces

Informally, *meta-surfaces* are "iso-surfaces" whose locus is defined by varying iso-values. Let g be a continuous map $g : \mathbb{R}^3 \to \mathbb{R}$ which assigns to each point $p \in \mathbb{R}^3$ an iso-value $c \in \mathbb{R}$ (g is called *iso-value function*), and let f be the implicit surface function. Then, a meta-surface implicitly is defined by the closure of points fulfilling the following

condition

$$f(p_x, p_y, p_z) = g(p_x, p_y, p_z), \tag{4.11}$$

where $(p_x, p_y, p_z) = p \in \mathbb{R}^3$ are points in 3D space. Thus, meta-surfaces in a sense are the generalization of iso-surfaces.

4.2.4 Meshes

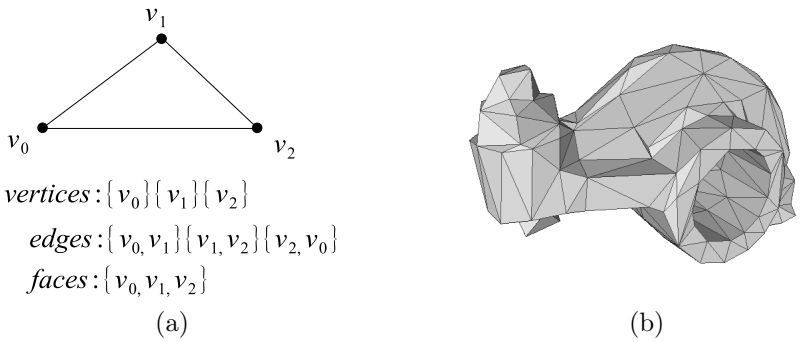


Figure 4.2: (a) principal composition of a 2D simplicial complex, illustrating the connectivity between zero-, one-, and two-dimensional simplicial elements. (b) example of a triangular mesh.

Intuitively spoken, a *mesh* is a piecewise continuous surface representation consisting of faces which amongst each other are connected along their edges. Formally, a mesh M is given by the pair (K, V), where: $V = \{v_0, v_1, \cdots, v_n\}$ with $v \in \mathbb{R}^3$ is a set of vertices defining the geometric characteristics of the mesh; K is a *simplicial complex* linking the so called *simplicial elements*, or *simplices*, together; Here, a simplex consisting of $d+1$ vertices is called a *d-dimensional simplex* or a *d-simplex*. A 0-simplex is a vertex, a 1-simplex is an edge, a 2-simplex is a face (see Figure 4.2a). Thus, the simplicial complex K describes the topological connectivity of M, whereas the vertex set V defines the geometric characteristics. Figure 4.2b shows an example of a 3D mesh complex consisting of vertices, edges and triangular faces. When considering the valence of vertices, defined by the number of edges incident to a vertex v, we can distinguish between regular and irregular meshes. In regular meshes all faces and vertices have the same size and valence. This does not hold for irregular meshes.

Since most graphics hardware is designed to efficiently render large amounts of geometric primitives like triangles or polygons, meshes are the most important form for representing surfaces.

4.2.5 Hybrid Meshes

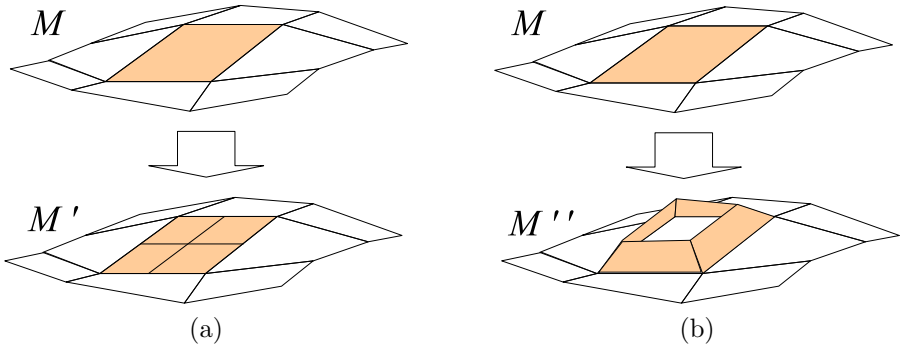


Figure 4.3: Examples illustrating the regular (a) and the irregular mesh refinement (b).

A *Hybrid mesh* (HM) [62], formally, is an hierarchical mesh representation obtained by applying a set of refinement operations to an existing mesh M. The operations can be classified into *regular* and *irregular refinement* operations. The regular mesh refinement subdivides the mesh through replacing each face by a number of subfaces. For instance, if the mesh is a quadrilateral mesh, each face is replaced by four subfaces (see Figure 4.3a). This operation induces a finer mesh M'. Irregular mesh refinement, on the other hand, states the process of removing a face f and replacing it by another set of faces incident to the boundaries of f (see Figure 4.3b). In general, the operation is restricted to create faces in a way that the resulting mesh M'' remains two-manifold.

The corresponding data structure is tree like: The faces of the original mesh are root nodes of independent trees. The subnodes represent the individual subfaces introduced by the regular mesh refinement. If irregular refinement is applied, the tree is extended about new root nodes representing the established irregular faces; Hybrid meshes are perfectly suited for representing surfaces at multiple refinement levels exhibiting different topological characteristics.

4.2.6 Surface Reconstruction

Given a set of n sample points $X = \{x_i, \cdots, x_n\}$ with $X \subseteq \mathbb{R}^3$ lying on an "unknown" surface S, then *surface reconstruction* states the process of finding a surface S' which approximates S as precise as possible. In some cases the points might be afflicted with noise $x_i = y_i + e_i$, where $y_i \in M$ is a point on the surface and $e_i \in \mathbb{R}^3$ is a so called error vector. In general, the success of surface reconstruction amongst others (see Section 2.4) depends on two important parameters:

a) The signal to noise ratio (SNR) (see section 2.4.1).

b) The point density ρ (see section 2.4.3).

These quantities may vary considerably within certain areas of the point cloud. If one of these exceeds a certain threshold, the reconstruction process becomes crucial. The threshold is not a fix parameter, it rather varies depending on the type of data set considered, the topology of the unknown surface S, and the type of surface reconstruction algorithms applied. Most of the reconstruction algorithms a priori assume the topology to be known e.g., they are based on the assumption that the points exclusively represent a surface. However, sometimes this does not hold, causing the algorithms to fail. Hence, incorporating topological information to the reconstruction is important.

4.2.7 Least Squares Surface Approximation

The *least squares* (LS) *surface approximation* refers to fitting continuous surface patches to a set of points $P = \{p_0, \cdots, p_n\} \subseteq \mathbb{R}^3$. Given an "unknown" surface S, the points $p \in P$ are located on, to be approximated by the parametric bivariate function f. The goal is to determine the coefficients of f in a way that f approximates S, as precise as possible. This is achieved through minimizing the squared residuals between the points of P and the function values of f, expressed by

$$E_f = \sum_{x,y \in M} \|f(x,y) - p_i\|^2, \tag{4.12}$$

where $M \subseteq \mathbb{R}^2$ denotes the parameter domain. Mostly, f either is represented by a bivariate polynomial, a Bézier, or a B-spline function (cf. section 4.2.2). A more detailed

discussion of the principles of the least squares fitting can be found in Appendix B. In addition, several other forms of LS surface approximation exist e.g., most popular, the moving least squares (MLS) approach (cf. section 3.4.2) [5] [104].

This approximation concept can be transferred to reconstruct implicit surface representations from a set of unorganized points. Let us assume g to be an implicit function, and $P \subseteq \mathbb{R}^3$ be the points to be approximated by the zero set of g. There are several possibilities the function g may be realized. One important approach is to express g in terms of *radial basis functions* (RBF) centered at constraints locations. Using RBFs we can write the interpolation function g as

$$g(x) = \sum_{j=0}^{n} d_j \phi(x - r_j) + q(x), \tag{4.13}$$

$$\phi(x) = |x|^3 \tag{4.14}$$

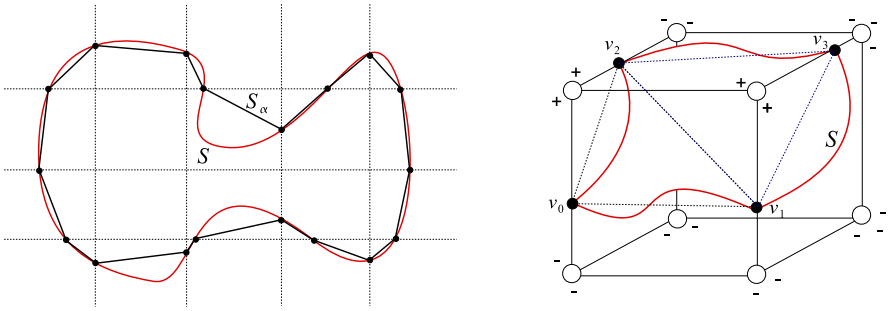
where r_j are the locations of the constraints, d_j are the coefficients, $q(x)$ is a degree one polynomial that accounts for portions of g, and $\phi(x)$ is the biharmonic radial basis function. RBFs are radially symmetric about its center and they can can be defined in such a way that they will automatically solve additional differential equations. A solution for g then can be found by minimizing

$$E_g = \sum_{p \in P} \left(g(p) - c\right)^2 \tag{4.15}$$

according the coefficients d_j, where $c = 0$. One possibility to extract surface representations from g in form of triangular meshes is provided by the marching cubes algorithm.

4.2.8 Marching Cubes

The *marching cubes*(MC) algorithm was first introduced by Lorensen and Cline [97], and operates as follows:
Given a 3D grid structure, consisting of small cubes that cover the support region of an implicit trivariate function g, the marching cubes algorithm locally generates a mesh for each cube that intersects the underlying zero set surface S of g (see Figure 4.4a).



(a) (b)

Figure 4.4: (a) 2D example in which a quadrilateral grid is laid over a 1D manifold S. The points at which S intersects the grid are connected by lines in a way the resulting line complex S_α approximates the shape of S, (b) illustrates the mesh generation regarding a single voxel.

Each vertex of a cube is associated with a function value of g. The algorithm computes intersection points between the surface and the grid edges of a cube whose grid vertices have opposite signs. These intersection points serve as the vertices of the corresponding surface mesh. In general, the intersection points are computed by linear interpolation of g between the endpoints of an intersected edge and are connected by a triangular mesh. This is accomplished in a way that the resulting mesh, inside a cube, approximates the locus of the underlying zero-surface (see Figure 4.4b). This is performed by taking into account the principal cases a surface can intersect a cube.

Depending on the grid resolution, the MC-algorithm produces a reliable and accurate surface approximation. Problems appear when the resolution of the cube-grid is too coarse. In this case, the algorithm may miss the extraction of intersection points inducing a mesh not reflecting the correct topology of the approximated iso-surface.

Over the last years, several modifications of the MC-algorithm have been proposed, intending to improve the efficiency and the accuracy of the algorithm e.g., [4] [3] [30] [77]. Instead of using cubes, another approach called *marching tetrahedra* [59] uses the simplest form of simplicial volume elements, namely tetrahedra. In particular, MC is applied for extracting iso-surfaces from volumetric data obtained from e.g., medical imaging systems or scientific simulation data.

4.3 Related Work

This section is intended to provide an overview about most important methods regarding surface reconstruction from unorganized 3D point data. Depending on the properties of the input data and the representation of the reconstructed surfaces we can distinguish between the following reconstruction methods:

4.3.1 Parametric Surface Reconstruction

A possible approach to obtain surfaces from unorganized point clouds is to fit surfaces to the input points [56], such as fitting polynomial [93] or algebraic surfaces [122]. To be able to fit surfaces to a set of unorganized points it is necessary to have information about the topology of the point cloud inherent surfaces or to have some form of parametrization in advance. For example, Eck and Hoppe [40] generate a first parametrization using their approach presented in [73] and fit a network of B-Spline patches to the initial surface. This allows to reconstruct surfaces of arbitrary topology. A competing spline-based method is provided by Guo [61]. Another form of surface reconstruction algorithm applying high-level model recognition is presented in [123].

Alexa et al. [8] introduced an approach for reconstructing point set surfaces from point clouds based on Levin's MLS projection operator. Further approaches following the idea of locally fitting polynomial surface patches to confined point neighborhoods are proposed in [7] [112] [46] [37]. In [104] the authors introduce a MLS reconstruction scheme which takes into account local curvature approximations to enhance the quality of the generated surfaces. One of the main problems associated with the MLS-based techniques is that, in general, they have to adapt to meet the underlying topological conditions. Depending on the type of input data this can be rather challenging. Another drawback is, that the MLS-technique in general is not capable of constructing surfaces having sharp features. One attempt for solving this problem was proposed by Fleishman et al. [46].

Whenever accuracy matters, adaptive methods are sought, capable of providing multiple levels of resolution, subdivision surfaces, for example, can be used together with wavelets [137] to represent highly detailed objects of arbitrary topology. In addition, such level-of-detail representations are well suited for further applications, like view-dependent rendering, multi-resolution editing, compression, and progressive transmis-

sion. In addition to adaptive polyhedral representations, subdivision surfaces provide smooth or piecewise smooth limit surfaces similar to spline surfaces. In [72] Hoppe et al. introduce a method for fitting subdivision surfaces to a set of unorganized 3D points.

4.3.2 Implicit Volumetric Surface Reconstruction

These reconstruction techniques transfer the input point cloud into volumetric representations implicitly defining the underlying surfaces. Using methods like the MC-algorithm [97] allows to extract explicit surfaces representations in form of polygonal meshes. Another possible approach for extracting iso-surfaces is provided by using particle-based methods as presented by [31].

One of the greatest challenges in reverse engineering is the reconstruction of watertight solids using implicit distance-based surface representations. A first method was described by Hoppe et al. [73]. Their approach constructs a Voronoi-based signed distance function (SDF), where the sign is determined by propagating oriented surfaces along a minimum spanning tree, from which the final surface can be extracted by contouring. Following this idea, the methods proposed by [158], [23] and [109] rely on defining SDFs whose zero level sets represent the desired surfaces. Once a distance field or a binary voxel-based representation is available, meshes are easily extracted by adaptive fairing methods. Surface reconstruction based on SDFs is quite popular, and so far a great variety of approaches have been proposed [32] [74] [114] [154] [15].

More recently, surface reconstruction based on scattered data approximation using globally supported RBFs was performed by Turk and O'Brien [144], Carr et al. [29] [28]. Othake et al. [116] and Morse et al. [108] employ compactly supported RBFs. The authors of [116] also extend the original RBF construction approach to multi-scale RBFs which allows to adaptively increase the approximation quality. Shen et al. [132] improved the original method by incorporating additional normal information in the RBFs allowing a more stable reconstruction. Another important form of implicit surface reconstruction using so called *blobby* or *meta-ball* surfaces is presented by [110]. The work of Taubin [141] and [58] is devoted to fit algebraic surfaces to a given collection of points.

The authors of [115] propose a multi-level partitioning of unity implicit surface reconstruction approach. This method locally fits and blends quadratic functions involving an octree-based space partitioning scheme. A technique closely related to the zero-set approaches are level-set algorithms by Whitaker [153] and Zhao et al. [158]. These

techniques define a function attracting the level set surface toward the original data points.

Another approach for reconstructing implicit surfaces from point sets with associated normals is introduced by Kazhdan [79] [99]. Their approach is based on defining a 3D indicator function (defined to be 1 inside and 0 outside the considered object) and the extraction of the appropriate iso-surface. The method draws advantage of a natural relationship between oriented point sampled surfaces and an indicator function. However, reliable normal vectors have to be available in advance.

4.3.3 Geometric Constrictive Approaches

Another class of reconstruction algorithms are computational geometry approaches. These algorithms usually extract the surface from previously computed Delaunay- or dual complexes. The reconstruction is based on mathematical guarantees but relies on clean data e.g., noisy, and non-regularly sampled points perturb the reconstruction process and may cause the algorithms to fail.

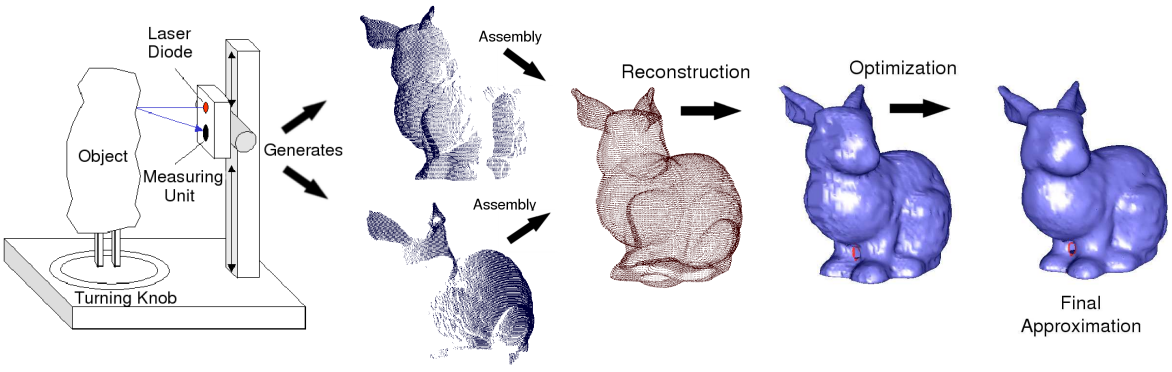
An early work concerning a Delaunay-based surface reconstruction scheme was provided by [24]. Following this idea, methods like the crust algorithm introduced by [10] have been developed exploiting the structure of the Voronoi diagrams of the input data. Other works [52] [12] [105] aimed at improving the original crust algorithm regarding efficiency and accuracy. The cocone algorithm [11] evolved from the crust algorithm provides further enhancements. Based on this work Dey et. al. [35] introduced the tight cocone. Other Delaunay/Voronoi-based reconstruction algorithms are presented by [84], [36].

One challenge concerns the separation of proximal sheets of a surface [10]. When considering local surface components, it may be helpful to construct a surface parametrization, i.e. a one-to-one mapping from a proper domain onto the surface. Having a surface of arbitrary topology split into a set of graph surfaces, for example by recursive clustering [69], one can reduce the reconstruction problem to scattered-data approximation in the plane [21]. A very powerful meshless parametrization method for reverse engineering is described by Floater and Reimers [47].

A completely different approach is the construction of α-shapes described by Edelsbrunner and Mücke [41]. Depending on a single radius α, their method collects all simplices (e.g. points, lines, triangles, tetrahedra, etc.) fitting into an α-sphere. The

method efficiently provides a data structure valid for all choices of α, such that a user may interactively adapt α to obtain a proper outer boundary of a point cloud. Out-of-core methods like ball-pivoting [20] employ the same principle, rolling a ball of sufficiently large radius around the point cloud filling in all visited triangles. Other methods e.g., [14] and [149], exploit octree-based grid structures to guide surface reconstruction.

4.4 Multi-Resolution Surface Reconstruction



(a) (b) (c) (d) (e)

Figure 4.5: Principle steps of reverse engineering: (a) point cloud acquisition by 3D object scanner, mostly laser range scan devices; (b) slices scanned from different views, (c) combined point cloud, (d) reconstructed mesh, (e) result after post-processing.

Computer-based surface models are indispensable in several fields of science and engineering. For example, the design and manufacturing of vehicles, such as cars and aircrafts, would not be possible without sophisticated CAD and simulation tools predicting the behavior of the product. On the other hand, designers often do not like working on virtual models, though sophisticated tools, like immersive VR-environments are available. Hence, a designer may produce a physical prototype made from materials of his choice that can be easily assembled and shaped like clay models. Reverse engineering is the process of reconstructing digital representations from physical models. The overall reverse-engineering framework mainly is composed of four steps (see Figure 4.5): data acquisition, pre-processing, surface reconstruction, and post-processing;

The point cloud acquisition generally is performed by stationary scanning devices, like laser-range or computer-tomography scanners. In the case of a 3D laser scanner, the

surface is sampled by one or more laser beams. The distance to the surface is typically measured by the time delay or by the reflection angle of the beam. After taking multiple scans from various sides or by rotating the object, the sampled points are combined into a single point cloud, from which the surface needs to be reconstructed (see Section 2.2).

Pre-processing of the data may be necessary, due to sampling errors, varying sampling density, and registration errors. Regions covered by multiple scans, for example, may result in noisy surfaces since tangential distances between nearest samples may be much smaller than the sampling error orthogonal to the surface. In this case, it is necessary to remove redundant points introduced by combining different points from multiple scans. In other regions, the density may be lower due to cavities and highly non-orthogonal scanning. If additional information, like a parametrization originating from each scan is available, interpolation can be used to fill these gaps.

In the present section, a powerful algorithm for multi-resolution surface extraction and -fairing, based on hybrid-meshes [62], from unorganized 3D point clouds is proposed (cf. [80] and [81]). The method uses an octree-based voxel hierarchy computed from the original points in an initial *hierarchical space partitioning* (HSP) process. At each octree level, the *hybrid mesh wrapping* (HMW) extracts the outer boundary of the voxel complex, taking into account the shape on the next coarser level. The resulting meshes both are linked into a structure with subdivision connectivity, where local topological modifications guarantee the resulting meshes are two-manifold. Subsequently, a *vertex mapping* (VM) procedure is proposed to project the mesh vertices onto locally fitted tangent planes. A final post-processing step aims on improving the quality of the generated mesh. This is achieved by applying a mesh relaxation step based on the constricted repositioning of mesh vertices tangential to the approximated surface.

4.4.1 Overview

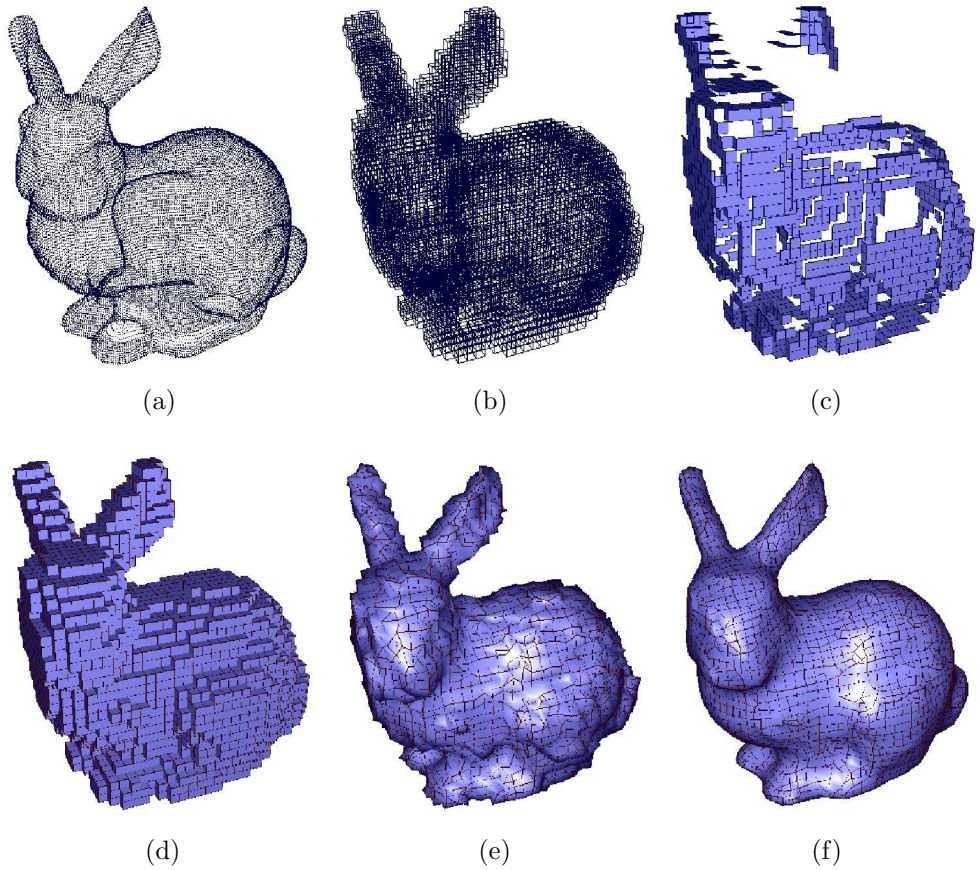


Figure 4.6: (a)-(f) principle steps of the proposed multi-resolution surface reconstruction approach. (a) input point cloud of the Stanford bunny data set, consisting of 35,947 points, (b) voxel grid after 4 subdivision steps, (c) and (d) extracted voxel hull before and after application of the HMW operation, (e) surface mesh after vertex projection, (f) final surface after mesh relaxation.

This work aims at finding an efficient way to extract a connected quadrilateral two-manifold mesh out of a given 3D point cloud in a way that the underlying "unknown" surface is approximated as accurate as possible. The resulting adaptive reconstruction method is based upon the repetitive application of the following steps:

- Starting from an initial bounding voxel enclosing the original point cloud (see Figure 4.6a), the hierarchical space partitioning creates a voxel set by recursively subdividing each individual voxel into eight subvoxels. Empty subvoxel are not

subject to subdivision and are deleted. Figure 4.6b presents an example of a generated voxel grid.

- The outer boundary of the generated voxel complex is extracted by the HMW operation. This exploits the voxel-subvoxel connectivity between the current and the next coarser voxel grid. The resulting mesh is obtained by subdividing the coarser mesh (cf. Figure 4.6c) and adapting its topology at locations where voxel have been removed (see Figure 4.6d).
- The final vertex mapping locally constricts the mesh toward the point cloud (cf. Figure 4.6e). All vertices are projected onto local tangent planes defined by the points of the individual voxels. The resulting mesh is relaxed toward its final position by applying additional post-processing (see Figure 4.6f).

4.4.2 Hierarchical Spatial Partitioning

The HSP presumes an already existing voxel set V^j defining the voxel complex at level j. At the coarsest level this set V^0 consists of the bounding voxel enclosing the entire point cloud. To obtain V^{j+1} the following steps are performed:

1 Subdivide every voxel $v \in V^j$ into 8 subvoxels.

2 Assign the points of v to the corresponding subvoxel and delete empty subvoxels.

The task of managing and maintaining the originated non-empty set V^{j+1} is accomplished by the usage of an octree data structure. As required by the succeeding HM wrapping operation we need certain connectivity information facilitating the localization of proximate voxel neighborhoods. The algorithm applied uses an efficient octree-based navigation scheme related to the approach of Bhattacharya [22].

4.4.3 Hybrid Mesh Wrapping

The most difficult part of this work concerns the extraction of a two-manifold mesh from the generated voxel complex V^{j+1}. For the following let M^j denote the set of faces representing the existing mesh corresponding to the voxel complex V^j, where the term face abstracts the quadrilateral sidepart of a voxel. M^0 defines the face patches of V^0 forming the hull of the bounding voxel. Starting from the existing mesh M^j, we obtain the next finer mesh representation M^{j+1} by performing regular and irregular refinement operations. This includes the subdivision of M^j and the introduction of new faces at M^{j+1} inducing local changes in the mesh topology. These operations require M^j to meet the following conditions:

- Each face $f \in M^j$ is associated with exactly one voxel $v \in V^j$ (no two voxel can be associated with the same face). Thus the maximum number of faces associated with a voxel is restricted to six.
- The mesh represented by M^j is a two-manifold mesh.
- A face is linked to each of its proximate neighbor face. n is called a neighbor of (or adjacent to) the face f, if both share a common voxel edge. With limitation of one neighbor per edge the number of possible neighbor faces of f is four.

4.4.4 Regular Mesh Refinement

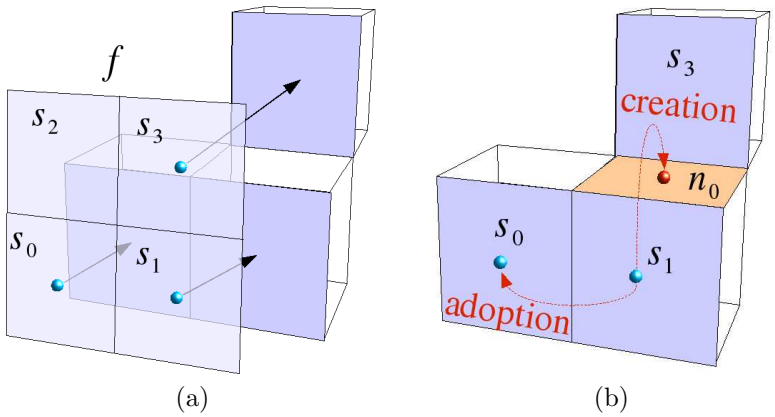


Figure 4.7: (a) regular refinement: Principles of the subface projection operation; (b) irregular mesh refinement: Principles of face creation and adoption;

To accomplish the acquisition of the boundary hull associated with V^{j+1} the first step concerns the regular refinement of M^j. The refinement is achieved by subdividing each face $f \in M^j$ into four subfaces. To guarantee the resulting mesh fulfills the conditions outlined above, we assign the subfaces of f to voxel of V^{j+1}. For the following let $v \in V^j$ be the voxel assigned to $f \in M^j$ and s be a subface of f. The assignment procedure projects s onto the corresponding subvoxel of v as illustrated in Figure 4.7a. f is restricted only to be projected onto the immediate subvoxel of v. Since we only address the extraction of the outer hull of V^j, additional rules have to be defined preventing faces of M^j to capture parts of the interior hull of the voxel complex e.g., in cases the surface is not closed. Thus, the subface s can not be assigned to a subvoxel of v if this subvoxel already is associated with an face on the opposite. Subfaces that can not be assigned because no corresponding subvoxel exist they could be projected onto, are removed. The resulting refinement operator $\mathcal{R}$ defines the set $N^{j+1} := \mathcal{R}(M^j)$, with $N^0 = M^0$ representing the collection of the created and projected subfaces (cf. Figure 4.6c). So far N^{j+1}, consisting of unconnected faces, represents the base for the subsequent *irregular mesh refinement*, in which the final mesh connectivity is recovered.

4.4.5 Irregular Mesh Refinement

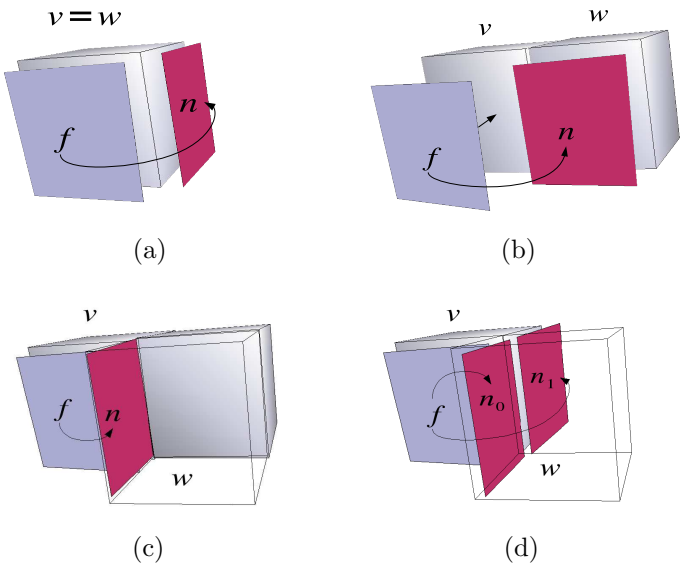


Figure 4.8: Graphical interpretation of the face-propagation rules concerning the possible cases.

The *irregular mesh refinement* recovers the mesh connectivity i.g., it reconnects the faces of N^{j+1} and closes resulting breaks in the mesh structure induced by the regular mesh refinement. This procedure is based on the propagation of faces or the adoption of existing proximate neighbor faces (see Figure 4.7b). More precisely: Assume f to be a face of N^{j+1}, the irregular refinement detects existing neighbor faces in N^{j+1} sharing a common voxel edge or creates new faces in case no neighbor is found. Considering the configuration of the principle voxel neighborhoods there are three possible cases a neighbor face of f can be adapted/created. Figure 4.8 depicts these cases. For the following considerations let $n \in N^{j+1}$ specifying the "missing" neighbor face of f, $v \in V^{j+1}$ the voxel associated with f, and $w \in V^{j+1}$ the voxel associated with n. Exploiting the voxel neighborhood relations, the propagation/adoption of n is performed according the summarized rules below:

1. In case that v and w are identical the creation/adoption of n is admitted if f does not already share an edge with a face on the opposite of v (see Figure 4.8a).

2. The faces n and f share a common edge, the corresponding voxel w and v adjoin a common face (see Figure 4.8b). The creation/adoption of n is allowed if no other face is associated with w, vis-a-vis from n (additional *cavity rule*).

3. The voxel v and w adjoin a common edge. In this case we have to differentiate between two cases: a) an additional voxel adjoins v (see Figure 4.8c). In this case the creation/adoption of n is permitted. b) only v and w adjoin a common edge (see Figure 4.8d). This case requires to detect the underlying topological conditions. If the underlying surface passes v and w, we adopt/create n_0, otherwise, if the points within v and w represent a break of the underlying surface we adopt/create n_1. The right case is filtered out by comparing the principal orientation of the points within v and w.

The irregular refinement procedure is performed as followed: Given two initial sets $A^0 = N^{j+1}$ and $B^0 = \emptyset$, once a new neighbor face n is created/adopted, it is added to $A^{i+1} = A^i \cup \{n\}$. Simultaneously, every $n \in A^i$ which is fully connected to all of its existing neighbor faces is removed $A^{i+1} = A^i \setminus \{n\}$ and attached to $B^{j+1} = B^j \cup \{n\}$. This procedure is repeated until $A^i = \emptyset$ or $A^{j+1} = A^j$.

Applying the irregular mesh refinement operator $\mathcal{I}$ on N^{j+1} results in $M^{j+1} = \mathcal{I}(N^{j+1})$, where $M^{j+1} = A^i \cup B^j$ represents the final mesh at level $j+1$. Figure 4.7b illustrates the propagation procedure, where one neighbor is created and another adopted.

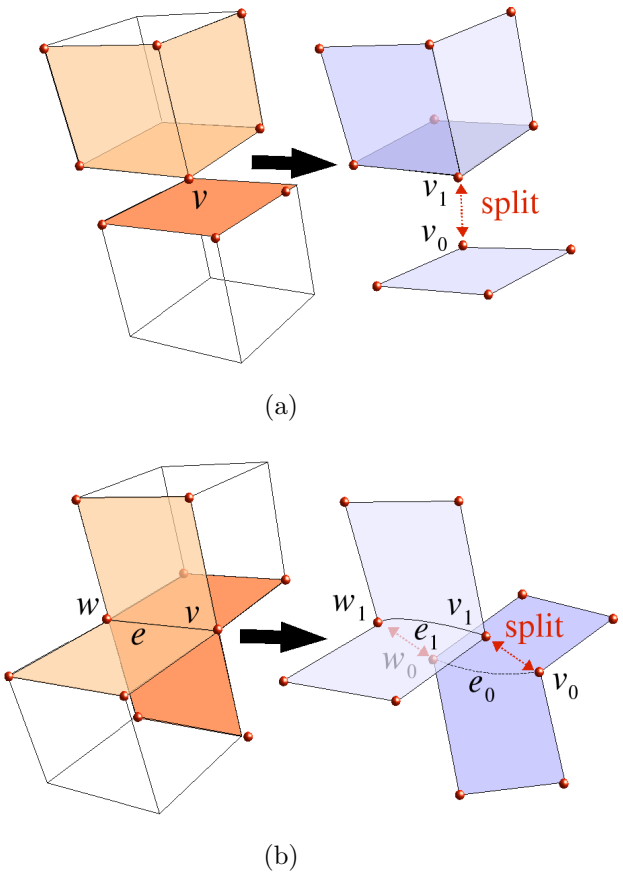


Figure 4.9: (a) example in which a non-manifold mesh structure is resolved by a vertex-split, (b) resolving non-manifold mesh structure by first applying an edge-split followed by two vertex-splits.

To force M^{j+1} to maintain the two-manifold condition each case at which the propagation of a face leads to a non-manifold mesh structure e.g., more than two faces share an edge, is identified and the mesh connectivity is resolved by applying vertex- and edge-splits. Figure 4.9a and Figure 4.9b illustrate the principles according to these the non-manifold mesh structures are resolved. In the depicted cases the connectivity of the vertices v and v_1, v_2 cause the mesh to be non-manifold. We avoid this by simply splitting the corresponding vertices and edges (see right part of Figure 4.9a and Figure 4.9b).

4.4.6 Vertex-Mapping

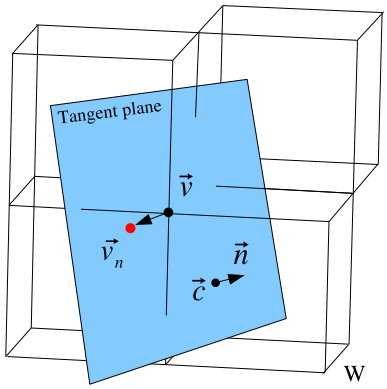


Figure 4.10: Vertex projected onto the tangent plane defined by the points P_v of the adjacent voxel set W.

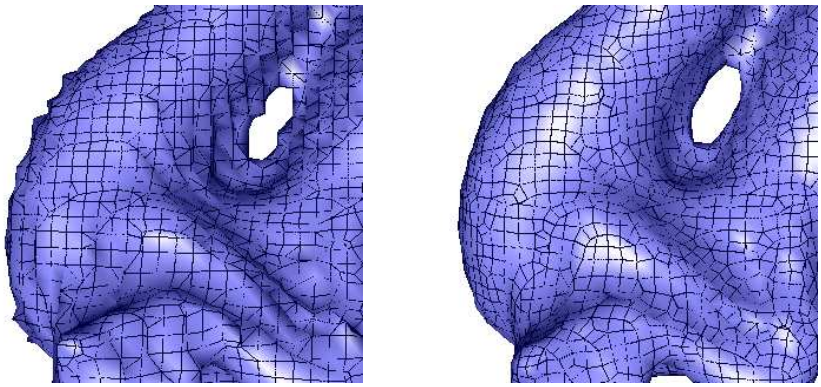
The next step concerns moving the mesh toward the "unknown" surface by projecting the mesh vertices onto the local tangent planes defined by the set of proximate sample points. This is accomplished for each vertex v of the mesh M by first identifying the voxel set W adjoining v and collecting the enclosed points P_v. Next, we fit a plane to the points P_v by computing the centroid $\vec{c}$ and the plane normal $\vec{n}$ obtained from the covariance matrix C of P_v. The normal $\vec{n}$ is defined by the eigenvector associated with the smallest eigenvalue of C. Together with the former position of the vertex $\vec{v}$ we are able to compute the new coordinates of $\vec{v_n}$ by

$$\vec{v_n} = \vec{v} - ((\vec{v} - \vec{c}) \cdot \vec{n})\vec{n} \ . \tag{4.16}$$

To be able to perform this projection the number of points of P_v has to be $|P_v| \geq 3$. Otherwise, points from adjacent voxels need to be added from surrounding voxel.

Post-Processing

To improve the quality of the generated mesh we perform an additional mesh optimization step. Based on the principles of Laplacian smoothing, the vertices of the mesh are repositioned by first computing the centroid of the directly connected neighbor vertices. In a subsequent step these centroids are again projected onto the tangent planes of the corresponding point sets according to equation (4.16). Generally, mesh-optimization is



(a) (b)

Figure 4.11: (a) part of the reconstructed surface mesh from the dragon data set before smoothing was applied, (b) corresponding mesh after few relaxation steps.

a repetitive process, applied several times to obtain the most possible gain in surface quality, see Figure 4.11a and Figure 4.11b.

4.4.7 Results

Performance

To find the overall time complexity we have to look at every step of the algorithm separately. We begin discussing the spatial decomposition analysis: In order to distribute the points contained by a voxel set V^{k-1} to their respective subvoxel we need to determine the subvoxel affiliation of every point. This leads to an computational complexity of $O(|P|)$ in every refinement step. Considering the HM wrapping we have to differentiate between the regular $\mathcal{R}$ and the irregular $\mathcal{I}$ operation but keep in mind that both are interdependent. Due to the fact that $\mathcal{R}$ basically depends linearly on the number of faces of $|M^{k-1}|$ and hence on $|V^{k-1}|$ we obtain a complexity of $O(|V^{k-1}|)$. Since it is difficult to estimate the number of attempts needed to find M^k we cannot reveal accurate statements concerning the computation time for $\mathcal{I}$. Based on empirical observations, an average time complexity of $O(k \cdot |V^k|)$ holds.

Assuming that $|V^k| \ll |P|$ with constant k (say $0 < k \leq 10$) the combined results of the particular sub-processes leads to an overall complexity of $O(|P|)$, concerning one single refinement step. The complexity to completely generate a mesh representing the unknown surface at refinement level k averages $O(k \cdot |P|)$.

The following time tables confirm the above discussed propositions. Table 4.1 shows the measured computation times for several data sets[1] performed on an Intel Pentium 4 system with 1.6 GHz and 256 MB main memory. Table 4.2 presents more detailed results for the individual reconstruction steps for the Stanford dragon point data set.

Object	points	faces	ref. level	time[sec] reconstr.	time[sec] opt.
Rabbit	8171	25234	6	1.058	0.244
Dragon	437645	72955	7	5.345	0.714
Buddha	543652	56292	7	5.170	0.573

Table 4.1: Computation time table regarding several input models whereas the VM and the mesh-optimization (4 passes) are performed after the last reconstruction step.

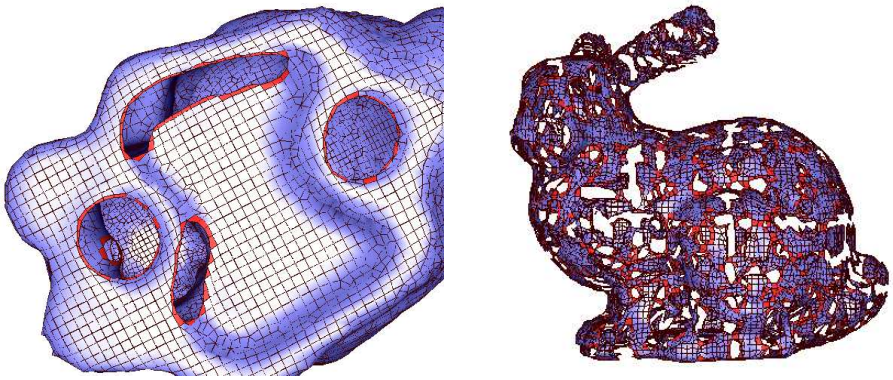
Ref.-Level	Spatial-Partitioning [sec]	HM-Extraction $\mathcal{R}$ [sec]	HM-Extraction $\mathcal{I}$ [sec]	Vertex-Mapping [sec]	Opt. [sec]	Complete [sec]
1	0.240	< 0.001	< 0.001	0.019	< 0.001	0.262
2	0.330	< 0.001	< 0.001	0.068	< 0.001	0.401
3	0.388	< 0.001	0.002	0.241	0.002	0.634
4	0.401	< 0.001	0.014	0.676	0.008	1.100
5	0.278	0.002	0.056	0.787	0.050	1.173
6	0.362	0.008	0.170	0.928	0.180	1.648
7	0.662	0.039	0.717	1.683	0.725	3.826

Table 4.2: Time values for each reconstruction step of the Stanford dragon.

Robustness and Quality

As shown by the examples our reconstruction method delivers meshes of good quality, as long as the resolution of the voxel complex does not exceed a point density induced threshold. Topological features, such as holes and cavities were always detected correctly after a proper number of refinements, see figure 4.12a. The HM wrapping rules prevent the underlying object from caving by simultaneously covering the real surface completely. Table 4.3 shows the measured L_2-errors obtained by processing point clouds of different models for several refinement levels.

[1] `http://graphics.stanford.edu/data/3Dscanrep/`



(a) (b)

Figure 4.12: (a) correctly detected holes in the bunny model, (b) fragmentation of the mesh due to lacking density of point cloud.

ref. level	1	2	3	4	5	6	7
L_2-error Bunny	25,86	9.47	3.60	1.13	0.41	0.14	-
L_2-error Buddha	-	51.92	18.80	7.61	3.47	1.58	0.66

Table 4.3: L_2-error values of the Bunny and Buddha reconstruction for each ref. level (measured in lengths of diagonal of the Bounding Box).

In case the voxel refinement level exceeds a specific bound imposed by sampling density, the resulting mesh may be incomplete in some areas, due to empty voxels, see figure 4.12b. If a very fine mesh is desired, one can fix the topology at a coarser level and apply subdivision and vertex mapping in consecutive steps.

[1]Web address: `http://graphics.stanford.edu/data/3Dscanrep/`

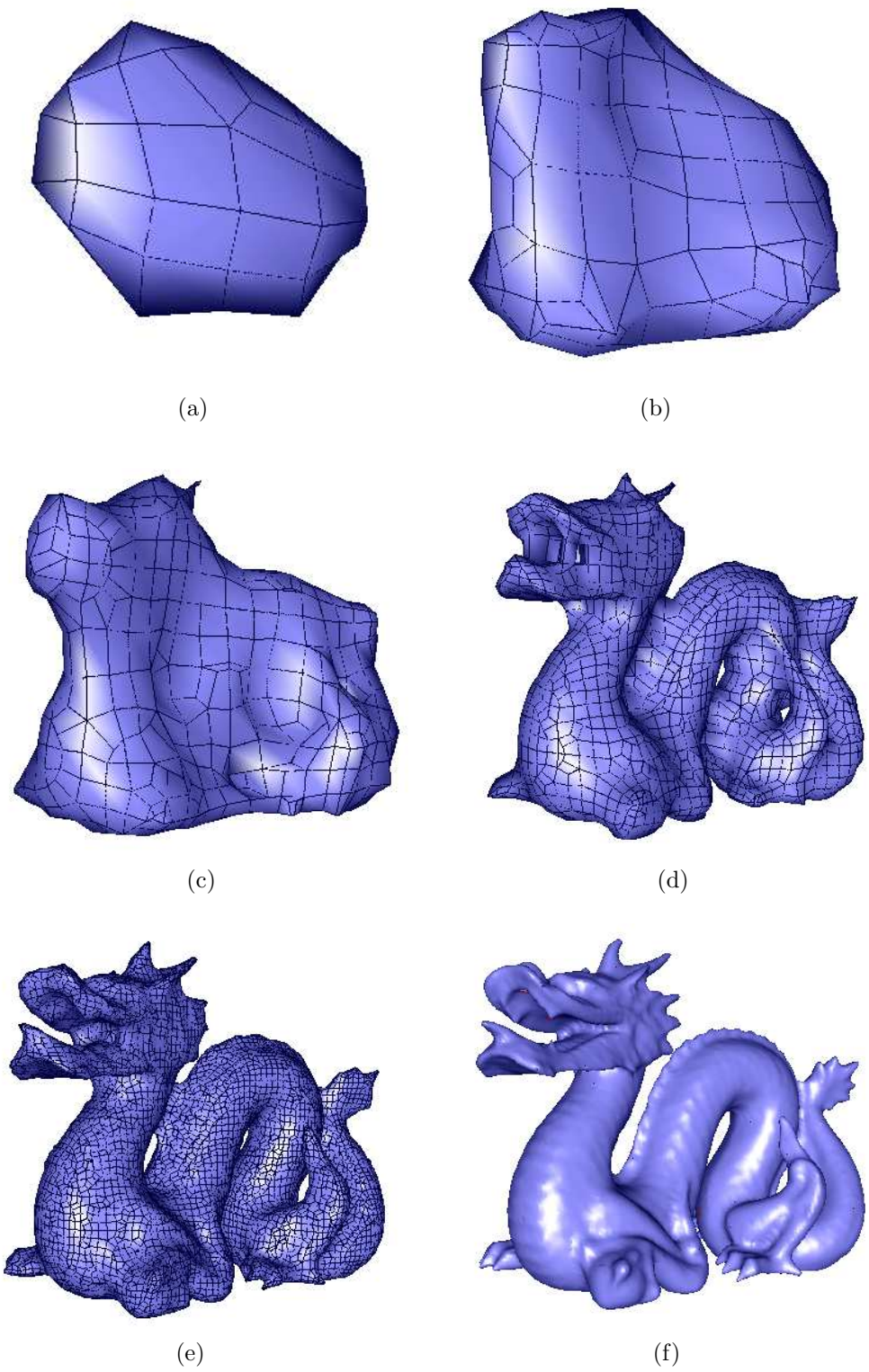
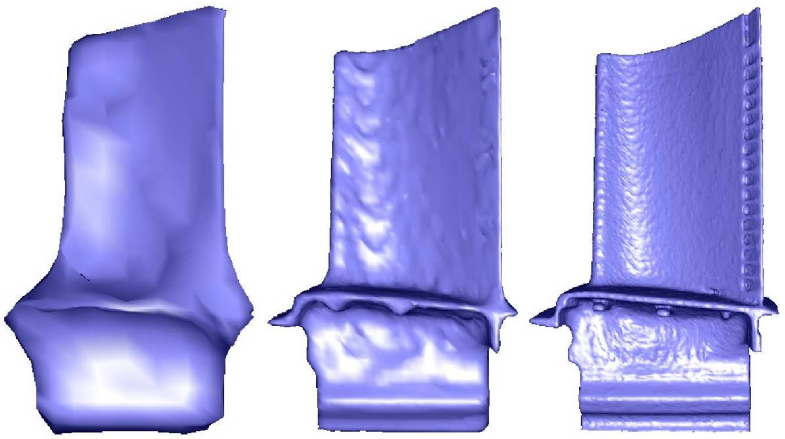
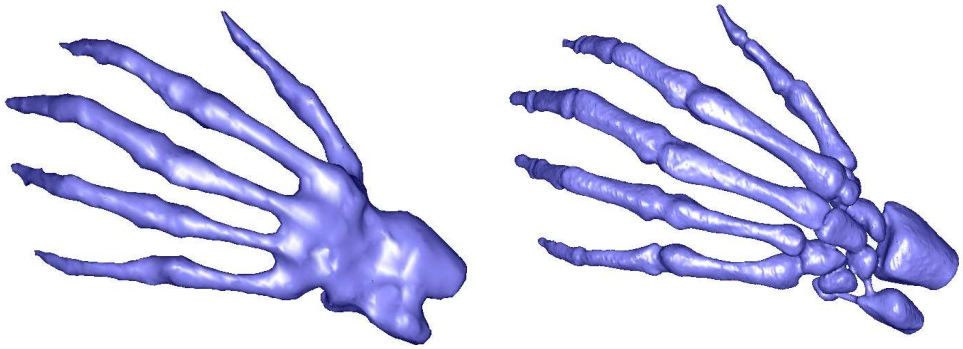


Figure 4.13: (a)-(f) different reconstruction-levels, from level one to six, of the Stanford dragon point data set [1] consisting of 437,645 points.



(a) (b) (c)

Figure 4.14: (a)-(c) reconstructed surface mesh of a turbine blade [1] (882,954 number of points) at different resolutions. (a) mesh at refinement level two, (b) at refinement level four and (c) at refinement level six.



(a) (b)

Figure 4.15: (a)-(b) reconstructed surfaces for the skeleton hand data set [1] (327,323 number of points). (a) mesh at refinement level three, and (b) at refinement level five.

[1]Web address: `http://www.cc.gatech.edu/projects/large\_models/`

4.5 Feature-Based Implicit Surface Reconstruction

The exploration of point data sets, like environmental LiDaR data, is a challenging problem of current interest. In addition to the vast amount of points obtained from terrestrial and airborne scanning, data complexity is increased by e.g., noise, occlusion, alternating sample density and overlapping samples (cf. Section 2.4). Despite of the high scanning resolution, undersampling occurs at small and fractured components like fences and leaves of trees. The best way to deal with such structures is to classify them into different feature groups (see previous Section 3.5) and select only surface-like structures for extraction.

In this section a new surface reconstruction approach is introduced [82] that allows to recover surface meshes from relative complex point data sets. Since the previously introduced multi-resolution reconstruction method does not distinguish between surface and non-surface related point structures, it rather assumes the input points always originate from surface structures, applying it to noisy and complex point data would not lead to reliable results. To perform reliable surface reconstruction a feature-based reconstruction approach taking into account the structural composition of the acquired point cloud is proposed. It extends the former approach to adapt the mesh extraction in a way that only surface-related point structures are approximated. This is accomplished by first performing a structural analysis of the point cloud. In chapter 3 a method is introduced that filters point data based on a probability-like classification of local point set neighborhoods into curve-like, surface-like, and solid-like shapes. The associated feature values were used to construct a feature-graph that can be visualized together with the original data to guide the user when exploring vast amounts of points without any additional information. In this section, a similar feature classification scheme is applied assigning feature values to the individual points that define whether a point corresponds to a surface or not. The proposed approach then constructs a *feature field* function approximating these feature values from which an initial surface representation is extracted. In this case, the extracted surface does not represent the outer hull of a voxel complex, instead it approximates a corresponding iso-surface. The locus of points associated with the local extrema of this implicit "feature function" most probably represents the underlying surface structures. Thus, moving the initial surface toward these local extrema, incorporating operations adapting the topology of the surface meshes, yields the final surface approximation. Since the proposed approach is meant to com-

plement the visualization of raw laser-scanned data, it provides a mesh-representation exhibiting a user specified topology. The topology can be controlled, similar to choosing a radius for alpha shapes in 3D space [41], in a way that the extracted surfaces adapt to the point cloud specific properties. A further advantage is that the extracted surface representation is continuous and thus can be used for further post-processing.
Compared to the previous surface reconstruction method, this section provides the following contributions:

- It introduces a new method that automatically classifies points of complex and noisy point clouds according their correspondence to surfaces.
- It presents an approach for constructing an implicit surface representation using trivariate multi-resolution B-Splines that describes how likely regions of the corresponding point cloud domain relate to surface-like structures.
- Based on the MC-algorithm the proposed method allows the extraction of meta-surfaces in form of triangular meshes representing the "unknown" surfaces.

This novel reconstruction approach allows a robust and selective surface reconstruction able to complement the visualization of raw scanned point data. User input is used to control the topology of the final surface representation by setting an iso-value threshold.

The remainder of this section is structured as follows: Section 4.5.1 provides an overview about the steps of the proposed method. In Section 4.5.2, Section 4.5.3 and Section 4.5.4 we discuss the individual steps of the reconstruction approach in detail. The results are presented in Section 4.5.6 followed by a comparison of the reconstruction method to the previous approach of Section 4.4. The conclusions are provided in Section 4.7.

4.5.1 Overview

The proposed method aims at generating an implicit representation of the underlying surface(s) in form of a 3D scalar field, named feature field. From this feature field the final surface representation is extracted using a MC-based approach. The proposed feature-based surface reconstruction consists of the following steps:

1. Pre-processing (sec. 4.5.2). This includes the organization of the input points into an octree-based voxel structure (Section 4.5.2) as well as the determination and

assignment of feature values to the individual points (Section 4.5.2). These feature values in a certain sense express the likelihood that a point is part of a surface structure by measuring the "planarity" of a point's local neighborhood.

2. The approximation/interpolation of the computed feature values using 3D (trivariate) hierarchical B-splines provides the scalar valued feature field function (see Section 4.5.3).

3. The extraction of an initial surface mesh, associated with an user-specified iso-value (Section 4.5.4) by using the MC-algorithm.

4. The adaption of the initial surface representation to approximate the meta-surface which represents the local maxima of the corresponding feature field (see Section 4.5.5). This includes additional refinement strategies increasing the quality of the final surface mesh.

4.5.2 Pre-Processing

Hierarchical Point Cloud Decomposition

For purposes of efficient data processing the proposed reconstruction method applies the octree-based data organization scheme used by the former approach (see Section 4.4.2). Considering the applied surface reconstruction algorithm, this form of data management has several advantages: Besides the efficient handling and visualization of point clouds as well as the acceleration of certain point operations, it facilitates the construction of the feature field function, since it makes possible applying the hierarchical B-spline approximation scheme. It also supports a straightforward MC-based iso-surface extraction by incorporating the edges of the multi-level voxel grid;

Determining Feature Values

The determination of the feature values expressing the correspondence of individual points to surfaces is based on the approach described in Section 3.5. The feature values are defined in a way that they measure the "planarity" of the local neighborhood $N(p)$ of a point $p \in P$, whereas $N(p)$ either can be defined by the set of k-nearest neighbor points of p, or the set of points lying within a specified radius r of p. In principle, we want the feature value of p to be high, if $N(p)$ corresponds to a smooth surface and low otherwise.

This is achieved by analyzing the geometric shape of $N(p)$ by relating the eigenvalues $\lambda_0 \leq \lambda_1 \leq \lambda_2$ of the covariance matrix $\mathbf{C}$ of $N(p)$ amongst each others. It holds that $N(p)$ has a spherical shape if $\lambda_0 \approx \lambda_1 \approx \lambda_2$, a planar shape if $\lambda_0 \ll \lambda_1 \wedge \lambda_1 \approx \lambda_2$ and a cylindrical shape if $\lambda_0 \approx \lambda_1 \wedge \lambda_1 \ll \lambda_2$. We combine this information and define the surface feature value f of a point p to be

$$f(p) = \frac{\lambda_1 - \lambda_0}{\lambda_2}. \tag{4.17}$$

The size of $N(p)$ has major impact on the obtained value f and is an important influence factor controlling the quality of the final surface representation. Choosing $N(p)$ to be small compared to the noise ratio or the point density results in strongly varying feature field interfering with the reconstruction process. Forcing $N(p)$ to be too large will degrade the quality of the extracted surface. This effect is discussed more detailed in Section 4.5.6.

4.5.3 Scattered Data Approximation

We now consider the approximation of the introduced feature values $f_i \in \mathbb{R}$ assigned to the input points $p_i \in P \subset \mathbb{R}^3$. Obtaining a smooth global feature function F approximating the given feature values f_i becomes a scattered data problem. This problem can be formalized as follows: Given a finite set of scattered data points $\{(p_i, f_i)|p \in \mathbb{R}^3, f_i \in \mathbb{R}\}$, construct a function $F : \mathbb{R}^3 \rightarrow \mathbb{R}$ that minimizes the residuals $\|F(p_i) - f_i\|$. There are several possible ways for accomplishing this task like using approximation schemes based on, e.g., RBFs [48] [121] or spline fitting techniques [92] [57]. A good overview of existing scattered data interpolation/approximation techniques can be found in [50] [95] [121] [129]. The introduced method follows the ideas presented in [21] [156] using hierarchical B-splines defined over the rectangular regions of a corresponding quadtree. This approach is able to provide a smooth and adaptive approximation of the feature field by a set of hierarchical trivariate B-spline functions. Extended to the 3D case, starting from the original bounding voxel enclosing the entire point cloud, the approximation scheme consists of the following steps:

1. The initial approximation of the determined feature values associated with the individual points by triquadratic Bézier functions.

2. Joining the individual Bézier functions into one continuous B-spline representation by applying the knot removal scheme proposed by Bertram et al. [21].

3. The Refinement of the approximation by performing above steps to the voxel grid at the next finer octree level.

Recursively applying these steps provides an hierarchical approximation of the feature values.

Initial Bézier Fitting

Let $P_i = \{(p_j, f_j)\}$ be a set of n scattered data points associated with the voxel V_i. Then, P_i is approximated by a trivariate Bézier function of the form

$$F_i(p_j) = \sum_{i=0}^{l} \sum_{j=0}^{l} \sum_{k=0}^{l} c_{i,j,k} \phi_{i,j,k}^{l}(p_j), \tag{4.18}$$

where $\phi_{j,j,k}^{l}(p_j)$ are the corresponding trivariate basis functions of degree l. The solution for F_i is found by solving Equation (4.5.3) according the basis coefficients $c_{i,j,k}$. In case, an overdetermined system is given, F_i is determined by minimizing the sum of the squared residuals φ with respect to the basis coefficients $c_{i,j,k}$

$$\varphi = \sum_{j=0}^{n} \|F_i(p_j) - f_j\|^2. \tag{4.19}$$

To deal with an under-determined system we add additional constraints forcing F_i to become zero in regions where no scattered data samples are given. The additional term has the form

$$\psi = \sum_{i=0}^{l} \sum_{j=0}^{l} \sum_{k=0}^{l} (c_{i,j,k})^2 \ . \tag{4.20}$$

Minimizing

$$\varepsilon = \varphi + \psi \longrightarrow min \tag{4.21}$$

requires solving a relatively small linear system of equations yielding the desired function F_i. Since we are interested in obtaining a continuous representation we apply further processing.

Global B-spline Approximation

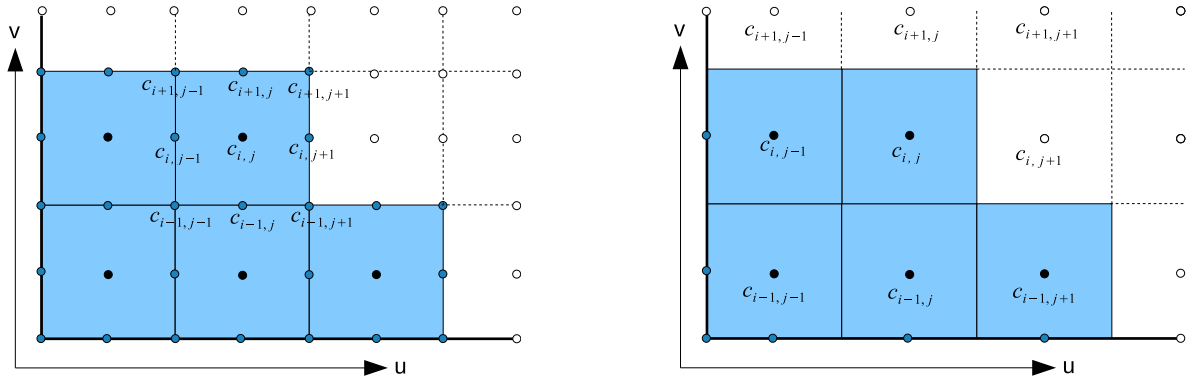


Figure 4.16: (a) Initial distribution of the basis coefficients defining corresponding biquadratic B-spline functions within an adaptive quad-grid consisting of empty and non-empty quadrilateral. Blue points represent zero-valued basis coefficients located at the domain boundaries of the data set having knot multiplicity three, black and white points are basis coefficients having knot multiplicity one. The white points are zero-valued basis coefficients of empty quadrilaterals. (b) quad-grid after knot removal.

To obtain a continuous approximation, we merge the individual Bézier functions F_i into a global B-spline representation F. The Bézier functions of degree l associated with the individual voxels can be represented as one global piecewise continuous B-spline function in which the control points at the voxel boundaries (faces, edges, vertices) have multiplicity l. The multiplicity of the knots associated with control points at the inside of a voxel is one. Figure 4.16a shows a 2D example that illustrates the distribution of the control points corresponding to a biquadratic B-spline over a quadrangular domain. The grid consists of empty (white) and non-empty (blue) quads. If a basis coefficient is located on the domain boundary of the point data set, its value is forced to zero. The

domain boundary of the data set in the 3D case is defined by the vertices, the edges and the faces of the smallest voxel enclosing the entire point set. In regions where voxels are classified as empty the control points are set to zero (depicted white). Removing knots at the voxel boundaries that are not located on the domain boundaries, yields the desired B-spline representation (see Figure 4.16b). In the triquadratic case the resulting B-spline is C^1 continuous. The knot removal may affect the quality of the approximation and thus demands re-fitting the remaining coefficients $c_{i,j,k}$. This is accomplished by sequentially readjusting each coefficient

$$c_{i,j,k} = \frac{\sum_{p \in V_i} \left(f_p - \sum_{x=0, x \neq i}^{l} \sum_{y=0, y \neq j}^{l} \sum_{z=0, z \neq k}^{l} c_{x,y,z} \phi_{x,y,z}(\vec{p}) \right)}{\sum_{p \in V_i} \phi_{i,j,k}(\vec{p})}, \tag{4.22}$$

where $\vec{p} \in \mathbb{R}^3$ denotes the coordinates of the points $p \in V_i$ in parameter space, and $f_p \in \mathbb{R}$ terms the feature value associated with p. Each adjustment of $c_{i,j,k}$ again has impact on the quality of the approximation in the proximate neighbor voxel. Hence, the re-fitting becomes an iterative process. The process is terminated if the local approximation error $|\sum_{p \in V_i} F(p) - f_p|$ falls below a given error bound or a maximum number of iterations have been performed. The resulting function F then represents a continuous approximation of the feature field function getting zero at the point clouds domain boundaries.

Problems arise in cases, where the least squares residual is small, but the slopes or curvatures of the trivariate B-spline functions are extremely high. This, especially, occurs in regions where the distribution of the points or the function values is one-sided. For example, in "empty" voxel regions the function values may take arbitrarily large values which results in a poor representation. To encounter this problem, additional constraints are incorporated forcing the final representation to be smooth. The enhanced approximation scheme incorporates the minimization of the first principal derivatives of the B-spline representation F at the positions of the individual points.

Recursive Refinement

Let $L_j = \{(P_i^j, F_i^j), \cdots\}$ be a set where P_i^j are the scattered point sets associated with the individual voxels at octree level j and F_i^j be the corresponding approximating B-spline functions. Starting at octree level j we determine F_i^j for each $P_i \neq \emptyset$. The subsequent merging of adjacent functions F_i^j is performed by applying the above intro-

duced knot removal. This leads to a globally C^1 continuous representation F^j at level j. In order to obtain L_{j+1} we recompute the feature values

$$\Delta f^{j+1}(p) = \Delta f^j(p) - F^j(p) \tag{4.23}$$

for each point, where $\Delta f_p^0 = f_p$ are the initial feature values assigned to p. This yields the basis for the subsequent approximation step performed at level $j+1$. Again, applying the least squares fitting procedure and merging the individual Bézier functions at level $j+1$ results in the continuous representations ΔF_i^{j+1} of the underlying residuals Δf_i^{j+1}. The final approximation is defined by

$$F^{j+1} = F^j + \Delta F^{j+1}. \tag{4.24}$$

The choice of the level at which the approximation is started and terminated has major impact on the performance and the quality of the final approximation. For example, starting the computation at a low resolution level e.g., on the root of the octree, and performing a large number of refinement steps would be computationally expensive. Otherwise, performing too few refinement steps may result in poor approximations. The B-spline refinement can be stopped, if a specific maximal refinement level is reached or the approximation error $|\sum_{p \in V_i} F^j(p) - f_p|$ approaches a predefined limit.

4.5.4 Initial Surface Extraction

Given a continuous function F approximating or interpolating the feature values assigned to the individual points, the first step aims at the extraction of a base mesh from F using the MC-algorithm [97] (see Section 4.2.8). The quality of the extracted mesh depends on the resolution of the voxel as well as on the quality of F. The MC algorithm generates two-manifold meshes reflecting the topology of the iso-surface at the given extraction level. In general, varying the iso-value results in base meshes having different topological characteristics. At this point the user has to control the surface topology by choosing the iso-value, such that the extracted mesh reflects the desired topology. This approach is inspired by the alpha-shape reconstruction method by Edelsbrunner and Mücke [41], where a simplicial Delaunay-complex is constructed for a point set, erasing all simplices

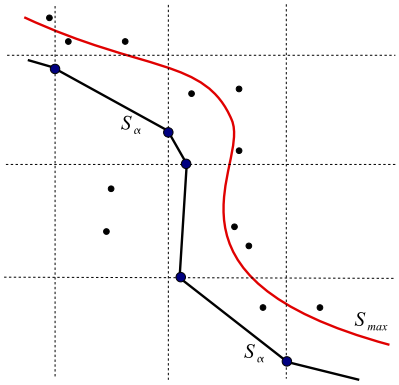


Figure 4.17: Two-dimensional example illustrating the extraction of the base mesh S_α via the MC algorithm regarding a specified iso-value α. The locus of the local maxima of F to be approximated by S_α is represented by the curve S_{max}.

whose circumscribed hypersphere exceeds a predefined radius alpha. Figure 4.19 shows some examples regarding a simple test data set. It illustrates the variation in topology, for extracting an initial surface from a precomputed feature function F, by varying the iso-value from low to high.

4.5.5 Approximation of Local Maxima

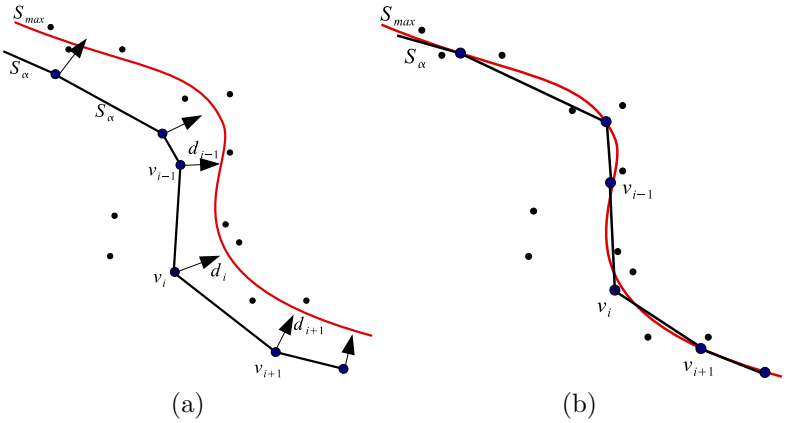


Figure 4.18: Examples illustrating the principles of approximating the meta-surface $S_{\max}$ representing the locus of local maxima of F. (a) initial approximation S_α with vertices v_i and corresponding gradient vectors d_i, before the vertex displacement is applied. (b) resulting mesh after the displacement of the vertices v_i along the vectors d_i toward S_{max}.

Next, the generated base is adapted to approximate the meta-surface representing the locus of local maxima of F. The set of points associated with this meta-surface is defined by $S_{\max}$. In a sense, $S_{\max}$ represents "ridge-lines" within the feature field. Depending on the topological characteristics of F the subsets of S_{max} can be associated with surfaces, lines or isolated points. As mentioned above, the function F approximates feature values assigned to the points P within the interval $[0, 1]$ where the value 1 (0) is associated with points assumed to represent surface (non-surface) related structures. Based on this, the surface related subsets of S_{max} most probably represent the point clouds inherent surfaces. The following procedure is intended to approximate these structures. Using the extracted base mesh, the meta-surface $S_{\max}$ is obtained by iteratively applying the following steps:

1. Moving the mesh toward the local maxima of the feature field F by displacing the mesh vertices (cf. Figure 4.18).

2. In addition, an adaptive strategy for updating the mesh connectivity, to improve the quality of the final representation, is applied.

Vertex Displacement

Moving the mesh toward S_{max} is achieved by displacing the vertices along the vector field ∇F. Let $\vec{d}$ be the corresponding displacement vector of vertex v, then v is relocated by iteratively moving v along $\vec{d}$ toward S_{max}. Since F approximates the local trend of the feature values assigned to points distributed in 3D space, the maximum values of F most probably occur in regions the corresponding point cloud has high feature values. Hence, projecting the vertices along $\vec{d}$ into these regions is obvious. Analog to the vertex mapping in Section 4.4.6, the new position $\vec{v}_{new}$ of v is defined by

$$\vec{v}_{new} = \vec{v} - \vec{d} \cdot (\vec{d} \cdot (\vec{v} - \vec{c})), \tag{4.25}$$

where $\vec{v} \in \mathbb{R}^3$ denotes the position of vertex v, and $\vec{c}$ represents the weighted centroid of the proximate point neighborhood $N(v)$ of v determined by

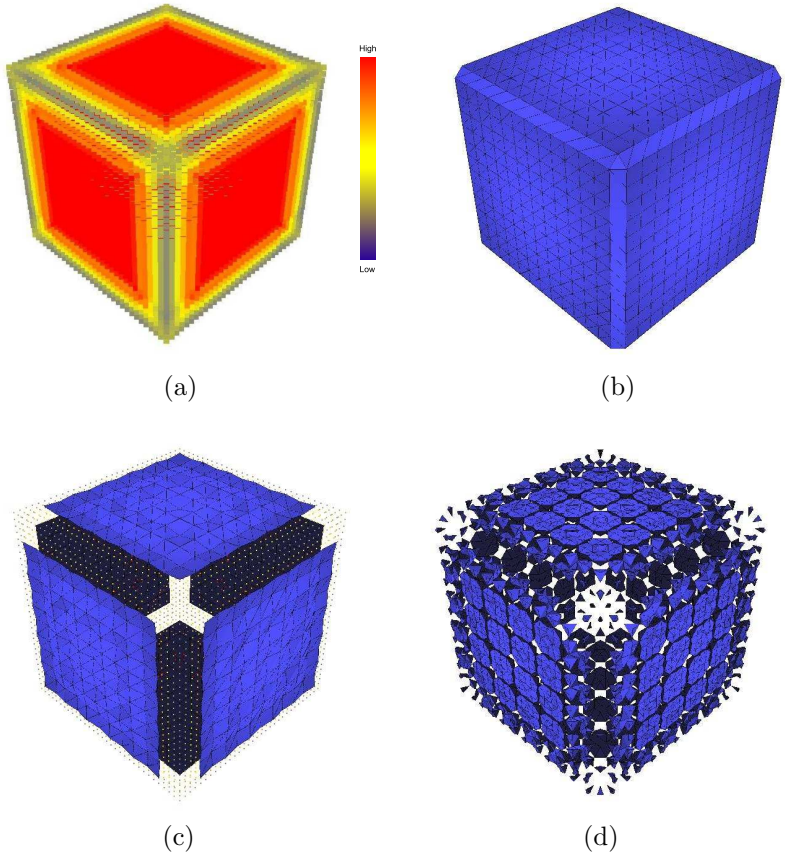


Figure 4.19: Example data set illustrating the effects of varying the iso-value at the initial extraction procedure. (a) Input point cloud of a simple test cube data set with points colored from blue to red according the correspondence of points to surface-related structures. (b)-(d) extracted base meshes obtained for iso-values (b) 0.4, (c) 0.6 and (d) 0.9.

$$\vec{c} = \sum_{p \in N(v)} \frac{f_p}{\sum_{q \in N(v)} f_q} \vec{p}, \tag{4.26}$$

$$\tag{4.27}$$

where f_p denotes the feature value assigned to the point p. The size of the neighborhood $N(v)$ is determined by the radius of the one-star neighborhood of vertex v. The one-star neighborhood of v is given by all vertices that are directly connected to v. Let r be the range to the most distant vertex corresponding to the one-star neighborhood of v, then

$N(v)$ is defined to be $N(v) = \{p \in \mathbb{R}^3 | \|v - p\| < r\}$.

Adapting Mesh Connectivity

This step becomes necessary when triangles corresponding to different mesh regions converge or intersect each other. To obtain the final surface representation we collapse vertices of affected triangles. The collapsing criterion depends on the size of the involved triangles. Collapsing two vertices v and w, not connected, is allowed, if the following condition holds:

$$\|\vec{v} - \vec{w}\| < r_{\min}, \tag{4.28}$$

where $r = min\{\cdots, r_i, \cdots\}$ with r_i representing the radius of the smallest sphere enclosing the triangle $t_i \in T_v \cup T_w$. The sets T_v and T_w consist of the triangles connected to v and w. If condition (4.28) is satisfied, the vertices v and w are collapsed and substituted by a new vertex v_{new}. The position of v_{new} is computed by $\vec{v}_{new} = \frac{1}{2}(\vec{v} + \vec{w})$. The collapsing operation takes into account additional rules preventing the mesh to break or to collapse. Additional operations are incorporated maintaining the mesh consistence e.g., by removing duplicated triangles. As a consequence of the vertex collapse, the resulting mesh is no longer guaranteed to be two-manifold.

After repositioning the vertices the resulting mesh might not approximate S_{max} in an optimal way. One possible reason is that the resolution of the mesh in some regions is too coarse. Subdividing selected triangles introduces new vertices at the centers of the triangles. Again, forcing these vertices toward $S_{\max}$ further increases the quality of the surface mesh. The decision whether a triangle is subdivided or not depends on the following error measure

$$\varepsilon = \left(\frac{\overline{f} - f_c}{\|\nabla F_c\|} \right)^2, \tag{4.29}$$

where $\overline{f}$ denotes the average feature value of the vertices of the considered triangle, f_c and ∇F_c represent the feature value and the gradient at the triangles center. We allow subdividing a triangle if $\varepsilon > r^2$, with r being the radius of the smallest sphere enclosing

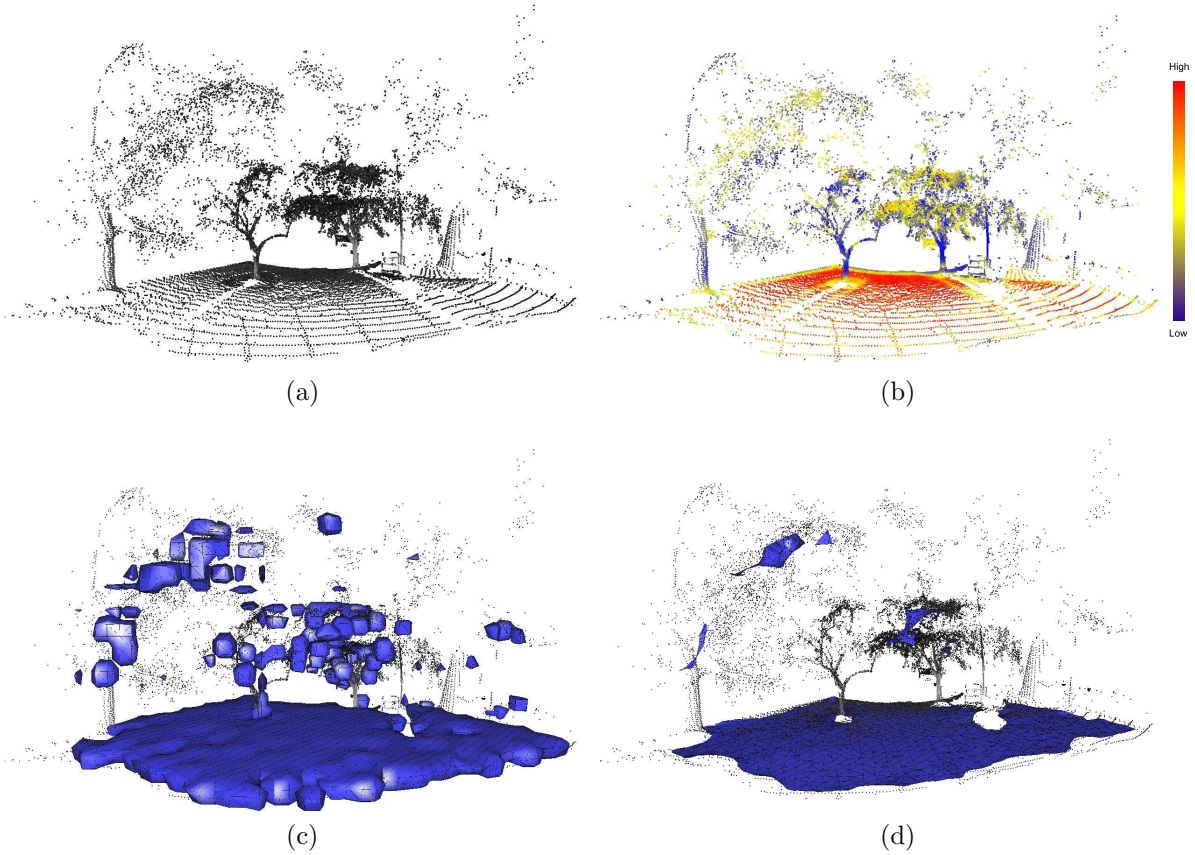
the corresponding triangle. Subdividing triangles introduces new vertices w at the center of the triangles. Those vertices, in a subsequent step, are moved toward S_{max} following the rules of vertex displacement (cf. Section 4.5.5). On the other hand, triangles are collapsed if they have a bad edge-to-edge aspect ratio. The quality of the mesh is further improved by applying a mesh relaxation similar to the relaxation proposed in Section 4.4.6.

Since the subsets of S_{max} in some cases represent points or curves, it may happen that the vertex displacement forces the mesh to collapse into point- or a curve-like structures. In this case the corresponding mesh structures are deleted.

4.5.6 Results

This section provides some results obtained by applying the proposed method to different point data sets. The used data sets are well suited to demonstrate the ability of the method to reconstruct surfaces from complex and noisy point data sets.

Figure 4.20a presents the raw input point cloud showing an environmental scene (having about 50,000 points) consisting of quite complex structures. Applying the proposed point classification scheme delivers feature values for each point. The classification was performed using the local point neighborhood $N(p)$ determined by the all points lying within a given range r of the considered point p. In the following, r is specified in terms of percentage of the diagonal of the smallest sphere enclosing the entire point cloud. Figure 4.20b visualizes these values by mapping colors ranging from blue to red to the points. Red colors indicate points in near-planar regions, whereas blue colors indicate points that are not assumed to be part of surfaces. In this example r was chosen to be 0.02. As the image shows, mainly the points at the ground are identified as surface-related structures, but also some isolated point structures at the crowns. This is due to the fact that the local neighborhoods of the isolated points are rather planar and thus the points misleadingly are assumed to lie on a surface. Figure 4.20c shows the base mesh extracted for the iso-value 0.6. Besides capturing the ground structure the initial mesh also captures the identified structures at the crowns. Figure 4.20c shows the final mesh after the application of the vertex displacement and the adaption of the mesh connectivity. The ground surface is almost completely recovered, except for a small part located at the tree trunk in the middle. Since the point cloud does not exhibit surface-like character at the area at which the trunk merges with the ground, the local



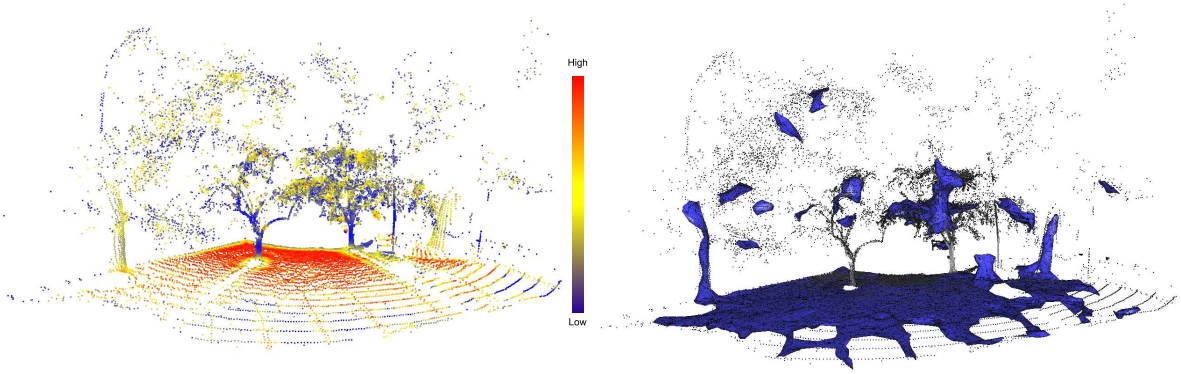
(a) (b)

(c) (d)

Figure 4.20: (a) original point cloud showing an environmental scene. (b) feature values, computed by the applied point classification scheme, which are mapped in form of color values to the corresponding points (red indicates surface-related, blue non-surface related points). (c) reconstructed intial mesh, (d) final surface representation.

point environment rather has cylindrical or spherical character, the affected region is not captured by the final surface.

The reconstruction process strongly depends on the choice of the neighborhood size considered for performing the point classification. Figure 4.21a demonstrates the effect of choosing the neighborhood size to be to small with respect to the density of the points located at the ground. In this example the radius determining the neighborhood $N(p)$ was chosen to be $r = 0.016$. The generated surface mesh, depicted by Figure 4.21b, now does not capture the entire ground. The points in the front at which the feature values are low (blue colors) are not approximated by the surface. Due to the small neighborhood size chosen the points are not assumed to be part of the ground surface



(a) (b)

Figure 4.21: (a) Visualization of the feature values for a chosen neighborhood size $r = 0.016$. (b) the final surface representation.

and thus not part of the final surface mesh.

In both examples the chosen parameters are equal, except for r. The B-spline based approximation of the feature values was started at octree level five and terminated at level ten. The cube grid used for the MC-based extraction was determined by partitioning the original bounding voxel into $40 \times 40 \times 40$ cubes.

The next example demonstrates the stability of the reconstruction approach by applying it to a modified point set. Figure 4.22a shows the point cloud of the Stanford bunny data set where noise is added in form of randomly distributed point samples. In this case, the boundary voxel, considered for adding the additional samples, is centered at the centroid of the point cloud and has the diagonal length of two times the length of the most distant point to the centroid. The number of additional noise samples in this example equals $80,000$ points, the original number of points of the bunny data set is approximately $35,000$. Figure 4.22b shows the color coded point cloud indicating the computed feature values. Again, blue represents non-surface like, red represents surface like point structures. The chosen neighborhood size equals $r = 0.01$. It is observable that the classification process identifies most points corresponding to the original point cloud. Applying the approximation scheme followed by the MC mesh extraction delivers the initial base mesh depicted in Figure 4.22c. The approximation was started at level six and was terminated at octree level ten. Here, the cube grid used for the MC-based mesh extraction was determined by partitioning the original bounding voxel into $80 \times 80 \times 80$ cubes. The final mesh after performing several repositioning and smoothing

Computation step	Time [sec.] environmental data set (50,000 points)	Time [sec.] modified bunny data set (115,000) points
Determination of feature values	46.95	72.34
Performing approximation	145.11	1107.84
Extracting base mesh	13.2	141.39
Vertex displacement (1st iteration)	7.65 (3,900 vertices)	415.12 (26,000 vertices)

Table 4.4: Computation times for each step of the algorithm concerning the two example data sets. The approximation was performed using four iterations at each processed octree level.

steps is depicted in Figure 4.22d. This example shows that the method provides reliable results in cases the point cloud becomes extremely noisy. Several experiments have been performed using an increased number of additional noise samples. The reconstruction provides acceptable results up to $100,000$ noise samples. Theoretically, the reconstruction is possible as long as the density of the points, corresponding to underlying surfaces, locally is higher than the density of the proximate noise samples. However, if the noise samples form surface-related point structures close to the original point samples the reconstruction might fail at these regions.

The average computational complexity of the proposed algorithm is estimated to be $O(k^2 \cdot n \cdot log(n))$, where k is the average number of points within $N(p)$ and n is the number of input data points. In case that $k << n$ the complexity $O(n \cdot log(n))$ rather holds. The complexity of the mesh extraction is estimated to be $O(m \cdot log(n))$ and primarily depends on the number of extracted mesh vertices m and the number of input points. Computationally, most expensive are the approximation of the feature values by F and the vertex displacement. Hence, it was decided to perform the determination of feature values and their approximation in a pre-processing step. This allows the user to perform the actual surface reconstruction more efficiently. Table 4.4 presents the computation times for the results depicted by Figure 4.20 and Figure 4.22. The processing was performed on a desktop computer equipped with an Intel Core2 Duo Processor and 4GB main memory.

4.6 Comparison

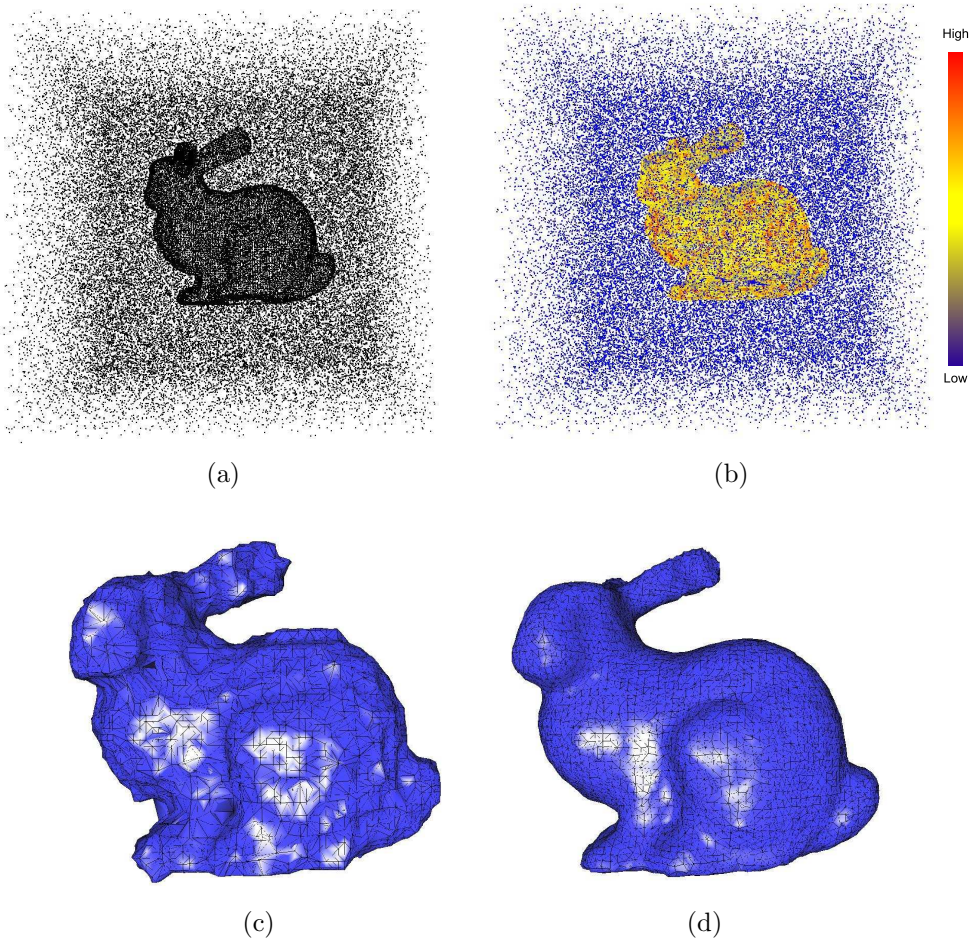
To outline the principal advantages and disadvantages a brief comparative discussion of both introduced approaches is provided. The first approach allows a reliable and precise surface reconstruction from unorganized point sets. However, applying it to noisy point data sets causes the reconstruction to fail. Figure 4.23a and Figure 4.23b shows the results of applying the multi-resolution surface reconstruction approach of Section 4.4 to the noisy bunny point cloud depicted in Figure 4.22a. Figure 4.23a shows the reconstructed surface mesh at octree level five, Figure 4.23b shows the mesh at level six. Here, the reconstruction fails completely. When reducing the number of noise samples to $25,000$, the method is able to provide a mesh capturing the surface of the bunny. Figure 4.23c shows the obtained surface mesh at reconstruction level five, Figure 4.23d shows the surface mesh obtained at level six. However, the quality of the reconstruction is low.

In contrast to the first approach, the feature-based reconstruction provides good reconstructions also concerning the original bunny data set. However, in some areas mainly at regions where the feature values are low due to e.g., high surface curvature the reconstruction might not exactly approximate the underlying surface. In some cases, the method trades accuracy for stability, especially, when surfaces featuring high curvature have to be reconstructed. This problem mainly is caused by the way the points are classified i.e., their relation to surface-like features is measured. Enhancing this classification is important for increasing surface quality and hence a potential direction for future work.

4.7 Conclusions

In this chapter two novel approaches regarding surface reconstruction from unorganized point clouds are introduced. The first method organizes the point cloud into an hierarchical voxel structure from which an initial surface representation is obtained by tessellation. Projecting the generated surface mesh onto tangent planes defined by proximate points produces the final surface representation. However, in cases the point cloud does not exclusively contain point structures that are approximable by surfaces, the method fails. The second approach proposed overcomes this drawback by incorporating an additional point classification scheme which allows to distinguish between surface- and non-surface

related point structures. Following the principles of the first approach an initial surface mesh is extracted from which the final representation is obtained by mesh repositioning. In contrast to the first approach this base mesh is derived from a 3D scalar valued feature field approximating the feature values assigned to the individual points. The topology of the final surface representation is controlled by the user. The produced examples show that the proposed method is capable to reconstruct continuous surfaces from complex and noisy point clouds, even if the local point density varies considerably. This makes it interesting for reconstructing surfaces from environmental point data, as provided by LiDaR scanners, containing a high variety of different point structures. However, it has to be adapted to be able to process large data sets more efficiently. Another drawback concerns the quality of the final reconstruction. Problems arise if the surfaces to be reconstructed feature regions with high curvature. Mainly, this problem can be traced back to the way the point classification is performed. Improving the applied scheme may solve this problem and would be one possible direction for future work.



(a) (b)

(c) (d)

Figure 4.22: (a)-(d) examples illustrating the surface reconstruction from a point cloud enriched with 80,000 noise samples. (a) original point cloud containing the Stanford bunny data set consisting of approx. 35,000 points, (b) point cloud featuring color values indicating the estimated strength of the correspondence of points to surface-related structures (blue indicates low, red indicates high correspondences). (c) the extracted base mesh, (d) the final surface mesh.

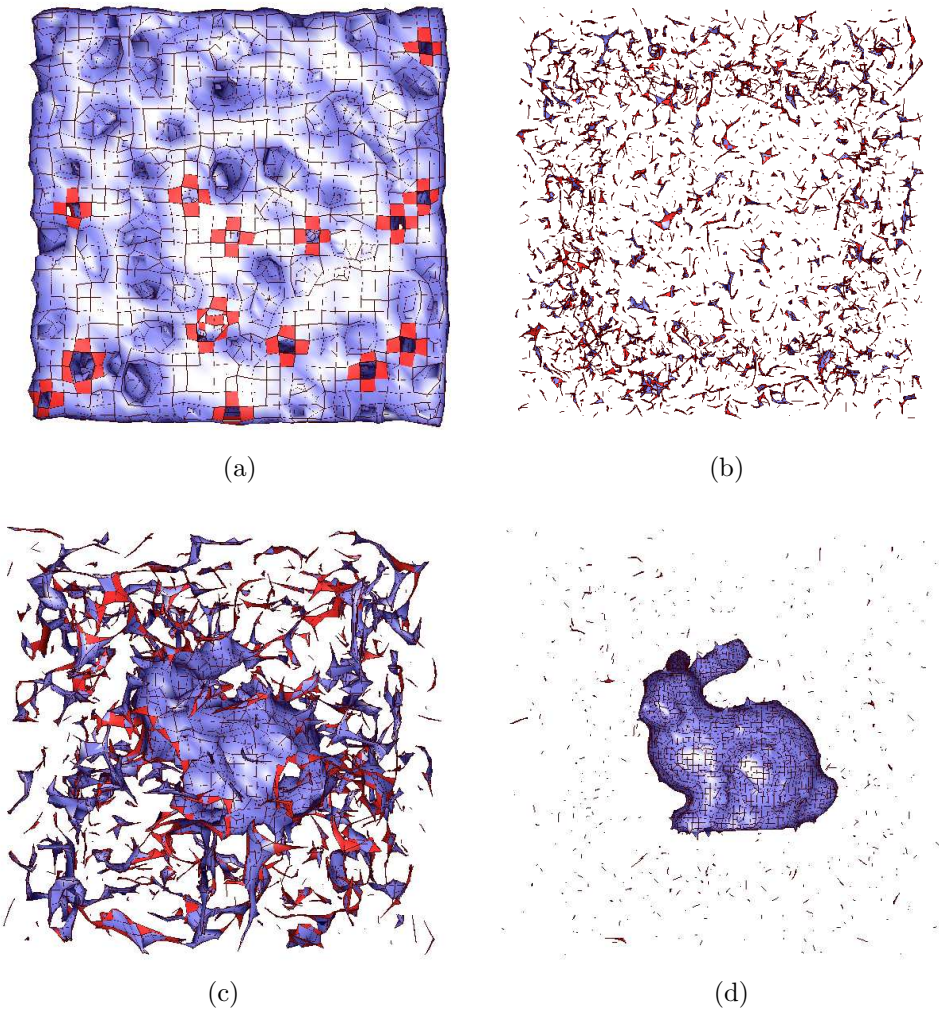


Figure 4.23: Surface reconstruction examples regarding the modified bunny point data set using the multi-resolution reconstruction approach. (a)-(b) resulting surface mesh at refinement level five (a) and level six (b), using the bunny data set enriched with $80,000$ noise samples. (c)-(d) final mesh at refinement level five (c) and at refinement level six (d), using the bunny point data set enriched with $25,000$ noise samples. Boundary faces are colored red.

5 Summary

This dissertation provides contributions in the field of point cloud acquisition as well as in the field of extracting point cloud inherent information in form of surfaces and feature graphs.

The work proposes a framework regarding the acquisition of point data from 2D images sequences using a maximum likelihood based camera calibration scheme that allows accurate reconstructions. This includes a discussion regarding the reliability of intensity- and feature-based correspondence image point matching schemes.

Furthermore, the thesis introduces an approach for analyzing point clouds using a probability-based classification of local point neighborhoods into curve-like, surface-like, and solid-like shapes. This allows to perform a detailed structural analysis of the point data at its full resolution. Based on the introduced point classification scheme a new approach regarding feature extraction is discussed. The feature values associated with the individual points are used to perform a stable and reliable extraction. The generated feature graphs facilitate the exploration of large, complex and noisy environmental point clouds in an intuitive way.

The thesis also addresses the extraction of surface features from point clouds. A novel method allowing the adaptive reconstruction of surfaces is presented. The introduced algorithm is able to efficiently provide surface representations reflecting the topology of the input point cloud at multiple resolutions. Following the idea of this approach a new feature-based surface reconstruction method, partially user guided, is introduced suitable to provide stable results even when focusing high complex and noisy point clouds. The method is based on applying a point based classification scheme, similar to the previous one. Using the generated feature values, a continuous implicit function quantifying whether a point locally represents a surface or not is constructed. A surface then is extracted with a variant of marching cubes which is transformed into a final representation most likely representing the point cloud inherent surface structures.

The work introduces interesting approaches covering several fields of point cloud pro-

cessing. Possible directions for future work mainly concern the extension of the proposed methods in a way that they allow the efficient processing of large-scale environmental point data sets. Another issue concerns the enhancement of the point classification scheme for determining the feature values associated with the individual points more efficiently.

A Determining the Gauss-Weingarten Matrix

When considering bivariate quadratic polynomials of the form

$$f(x, y) = c_{0,0} + c_{1,0}x + c_{0,1}y + c_{2,0}x^2 + c_{1,1}xy + c_{0,2}y^2$$

we can determine the principle curvature values and the principle curvature direction directly from the Gauss-Weingarten matrix. The Gauss-Weingarten matrix is defined as

$$W = G^{-1}B \tag{A.1}$$

with

$$G = \begin{pmatrix} f_u^2 & f_u f_v \\ f_u f_v & f_v^2 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} nf_{uu} & nf_{uv} \\ nf_{uv} & nf_{vv} \end{pmatrix} \tag{A.2}$$

where n donates the normal vector. f_x and f_{xx}, f_{xy} denote the first and second partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial^2 f}{\partial^2 x}$, $\frac{\partial^2 f}{\partial x \partial y}$. Assuming biquadratic polynomials we can rewrite G and B to be

$$G = \begin{pmatrix} 1 + c_{1,0} & c_{1,0}c_{0,1} \\ c_{1,0}c_{0,1} & 1 + c_{0,1} \end{pmatrix} \tag{A.3}$$

$$B = \frac{1}{\sqrt{1 + c_{1,0}^2 + c_{0,1}^2}} \begin{pmatrix} 2c_{2,0} & c_{1,1} \\ c_{1,1} & 2c_{0,2} \end{pmatrix}. \tag{A.4}$$

B Least Squares Fitting

We review the basic concept of fitting a function to a set of unorganized function values. Given a set of n points $p_i \in P$ in d-dimensional space space $\mathbb{R}^d$ with associated function values y_i. The least squares approximation determines the coefficients $c = (c_1, \cdots, c_m)^T$ for a function of the form

$$f(x) = \sum_{j=0}^{m} c_j B_j(x), \tag{B.1}$$

where $B_j(x)$ represent the corresponding basis functions. The approximation is accomplished by finding coefficients in a way that the residual

$$\varepsilon(c) = \sum_{i=0}^{n} \left(f(p_i) - y_i\right)^2, n \geq m, \tag{B.2}$$

is minimized. Solving the following linear system of equations for the coefficients c_k delivers the function f that best approximates the given function values y_i:

$$\frac{\partial \varepsilon}{\partial c_k} = 2 \sum_{i=0}^{n} \left(\sum_{j=0}^{m} c_j B_j(p_i) - y_i \right) B_k(p_i) = 0. \tag{B.3}$$

This can be rewritten in matrix form as

$$BB^T c = By, \tag{B.4}$$

with $b_{ij} = B_i(p_j)$ and $c = (c_0, ..., c_m)^T$. The solution for c is given by

$$c = (BB^T)^{-1}By, \tag{B.5}$$

In case the system is singular (e.g., all points are colinear) no solution for c is found. For a more detailed introduction see [43].

Bibliography

[1] The digital michelangelo project. Website. `http://graphics.stanford.edu/projects/mich/`.

[2] OpenCV. Website. `http://opencv.willowgarage.com/wiki/`.

[3] *The asymptotic decider: Resolving the ambiguity in marching cubes*, 1991.

[4] *Dual Marching Cubes*, 2004.

[5] M. Alexa, , J. Behr, D. Cohen-or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *Proceedings of the conference on Visualization '01*, pages 21–28, 2001.

[6] M. Alexa and A. Adamson. On normals and projection operators for surfaces defined by point sets. In *In Proc. of the Eurographics Symposium on Point-Based Graphics*, pages 149–156, 2004.

[7] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[8] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *Conference on Visualization*, pages 21–28. IEEE Computer Society, 2001.

[9] A. Alharthy and J. Bethel. Detailed building reconstruction from airborne laser data using a moving surface method. In *20th Congress of International Society for Photogrammetry and Remote Sensing*, pages 213–218, 2004.

[10] N. Amenta, M. Bernd, and D. E. Kolluri. The crust and the beta-skeleton: combinatorial curve reconstruction. In *Graphical Models and Image Processing, 60*, pages 125–135, 1998.

[11] N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 213–222, 2000.

[12] N. Amenta, S. Choi, and R. K. Kolluri. The power crust. In *6th ACM symposium on Solid Modeling and Applications*, pages 249–266. ACM Press, 2001.

[13] S. Ar, G. Montag, and A. Tal. Deferred, self-organizing bsp trees. *Computer Graphic Forum*, 21(3):269–278, 2002.

[14] S. Azernikov, A. Miropolsky, and A. Fischer. Surface reconstruction of freeform objects based on multiresolution volumetric method. In *ACM Solid Modeling and Applications*, pages 115–126, 2003.

[15] Bajaj, C. L., Bernardini, Fausto, Xu, and Guoliang. Automatic reconstruction of surfaces and scalar fields from 3d scans. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 109–118, 1995.

[16] A. Belyaev and E. Anoshkina. Detection of surface creases in range data. In *Mathematics of Surfaces*, pages 50–61, 2005.

[17] P. Benkö and T. Várady. Direct segmentation of smooth, multiple point regions. In *Geometric Modeling and Processing - Theory and Applications*, page 169. IEEE Computer Society, 2002.

[18] P. Benkö and T. Várady. Segmentation methods for smooth point regions of conventional engineering objects. *Computer-Aided Design*, 36(6):511–523, 2004.

[19] J. Bentley. Multidimensional binary search trees used for associative searching. In *Communications of the ACM*, pages 509–517, 1975.

[20] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorith for surface reconstruction. pages 349–359, 1999.

[21] M. Bertram, X. Tricoche, and H. Hagen. Adaptive smooth scattered-data approximation for large-scale terrain visualization. In *Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization*, pages 177–184, 2003.

[22] P. Bhattacharya. Efficient neighbor finding algorithms in quadtree and octree, 2001.

[23] J. Bloomenthal, C. Bajaj, J. Blinn, M.-P. Cani-Gascuel, A. Rockwood, B. Wyvill, and G. Wyvill. *Introduction to implicit surfaces*. Morgan KAufman, Inc., San Francisco, 1997.

[24] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4):266–286, 1984.

[25] T. Bonfort and P. Sturm. Voxel carving for specular surfaces. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 591, 2003.

[26] M. Botsch and L. Kobbelt. High-quality point-based rendering on modern gpus, 2003.

[27] R. L. Carceroni and K. N. Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape & reflectance. *Computer Vision, IEEE International Conference on*, 2:60, 2001.

[28] J. Carr, R. Beatson, J. Cherry, T. Mitchel, W. Fright, B. McCallum, and T. Evans. Smooth surface reconstruction from noisy range data. In *Proc. SIGGRAPH Computer Graphics*, 2001.

[29] J. Carr, R. Beatson, B. McCallum, W. Fright, T. McLennan, and T. Mitchel. Smooth surface reconstruction from noisy range data. In *Proc. ACM GRAPHITE*, pages 119–126, 2003.

[30] R. S. Chen, C. Zhou, and M. S. Kankanhalli. Adaptive marching cubes. *The Visual Computer*, 11:202–217, 1995.

[31] P. Crossno and E. Angel. Isosurface extraction using particle systems. In *Proceedings of the 8th conference on Visualization '97*, pages 495–ff, 1997.

[32] B. Cureless and M. Levoy. A volumetric method for building complex models from range images. In *ACM Siggraph '96*, pages 303–312, 1996.

[33] C. Dachsbacher, C. Vogelgsang, and M. Stamminger. Sequential point trees. *ACM Transactions on Graphics (TOG)*, 22(3):657–662, 2003.

[34] J. I. Daniels, L. K. Ha, T. Ochotta, and C. T. Silva. Robust smooth feature extraction from point clouds. In *IEEE International Conference on Shape Modeling and Applications, 2007. SMI '07.*, pages 123–136, June 2007.

[35] T. K. Dey and S. Goswami. Tight cocone: A water tight surface reconstructor. In *Proc. 8th ACM Sympos. Solid Modeling Appl.*, pages 127–134, 2003.

[36] T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. In *20th Annual Symposium on Computational Geometry*, pages 330–339. ACM Press, 2004.

[37] T. K. Dey and J. Sun. An adaptive mls surface for reconstruction with guarantees. In *3rd Eurographics Symposium on Geometry Processing*, pages 43–52, 2005.

[38] M. DoCarmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, second edition, 1976.

[39] I. Douros and B. F. Buxton. Three-dimensional surface curvature estimation using quadric surface patches. In *Scanning*, 2002.

[40] M. Eck and H. Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *ACM Siggraph*, pages 325–334, 1996.

[41] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[42] P. Eisert, E. Steinbach, and B. Girod. Multi-hypothesis, volumetric reconstruction of 3-d objects from multiple calibrated camera views. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 6:3509–3512, 1999.

[43] G. Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Morgan Kaufmann, 5 edition, 2002.

[44] O. D. Faugeras, E. Le Bras-Mehlman, and J. D. Boissonnat. Representing stereo data with the delaunay triangulation. *Artif. Intell.*, 44(1-2):41–87, 1990.

[45] S. Fleishman, D. Cohen-or, M. Alexa, and C. T. Silva. Progressive point set surfaces. In *ACM Transactions on Graphics (TOG)*, pages 997–1011, 2003.

[46] S. Fleishman, D. Cohen-Or, and C. Silva. Robust moving least-squares fitting with sharp features. In *ACM Siggraph*, pages 544–552, 2005.

[47] M. Floater and M. Reimers. Meshless parameterization and surface reconstruction. 18(2):77–92, 2001.

[48] T. Foley. Scattered data interpolation and approximation with error bounds. In *In Computer Aided Geometric Design*, pages 163–177, 1986.

[49] T. Foley and J. Sugerman. Kd-tree acceleration structures for a gpu raytracer. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 15–22, 2005.

[50] R. Franke. Scattered data interpolation: Test of some methods. *In Mathematics of Computation*, 38(157):181–200, 1982.

[51] T. Fromherz and M. Bichsel. Shape from multiple cues: Integrating local brightness information. In *Proceedings of the Fourth International Conference for Young Computer Scientists, ICYCS 95*, pages 855–862, 1995.

[52] S. Funke and E. A. Ramos. Smooth-surface reconstruction in near-linear time. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 781–790, 2002.

[53] P. Gargallo and P. Sturm. Bayesian 3d modeling from images using multiple depth maps. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:885–891, 2005.

[54] E. Gobbetti and F. Marton. Layered point clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models, 2004.

[55] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics*, 23(1):45–63, 2004.

[56] A. Goshtasby and W. D. O'Neill. Surface fitting to scattered data by a sum of gaussians. *Comput. Aided Geom. Des.*, 10(2):143–156, 1993.

[57] B. Gregorsky, B. Hamann, and K. Joy. Reconstruction of b-spline surfaces from scattered data points. In *Computer Graphics International 2000*, pages 163–170, 2000.

[58] G. Guennebaud and M. Gross. Algebraic point set surfaces. In *In Proceedings SIGGRAPH 07*, 2007.

[59] A. Gueziec and R. Hummel. Exploiting triangulated surface extraction using tetrahedraldecomposition. 1(4):328–342, 1995.

[60] S. Gumhold, X. Wang, and R. MacLeod. Feature extraction from point clouds. In *10th International Meshing Roundtable*, pages 293–305, 2001.

[61] B. Guo. Surface reconstruction: from points to splines. 29(4):269–277, 1997.

[62] I. Guskov, A. Khodakovsky, P. Schroeder, and W. Sweldens. Hybrid meshes: Multiresolution using regular and irregular refinement. In *ACM Symposium on Geometric Modeling (SoCG)*, pages 264–272, 2002.

[63] B. Hamann. Curvature approximation for triangulated surfaces. *Geometric Modelling, Computing Suppl. 8*, pages 139–153, 1993.

[64] K. Hammann. 3d reconstruktion von oberflächen aus 2d bilddaten mittels maximum-likelihood parameterschätzung. Master's thesis, Department of Computer Science, University of Kaiserslautern, Germany, 2009.

[65] C. Harris and M. Stephens. A combined corner and edge detector. In *The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[66] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, March 2004.

[67] R. C. T. Hartley, R. Gupta. Stereo from uncalibrated cameras. In *IEEE Computer Society Conference Proceedings on Computer Vision and Pattern Recognition (CVPR) '92*, pages 761–764, 1992.

[68] R. I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *ECCV '92: Proceedings of the Second European Conference on Computer Vision*, pages 579–587, London, UK, 1992. Springer-Verlag.

[69] B. Heckel, A. Uva, and B. Hamann. Cluster-based generation of hierarchical surface models. In *Scientific Visualization, Dagstuhl*, pages 113–222, 1997.

[70] K. Hildebrandt, K. Polthier, and M. Wardetzky. Smooth feature lines on surface meshes. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, pages 85–90. Eurographics Association, 2005.

[71] M. Hopf and T. Ertl. Hierarchical splatting of scattered data. In *In VIS 03: Proceedings of the 14th IEEE Visualization 2003 (VIS03*, pages 433–440, 2003.

[72] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *ACM SIGGRAPH 1994 Conference Proceedings*, pages 295–302, 1994.

[73] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *ACM SIGGRAPH '92 Conference Proceedings*, pages 71–78, 1992.

[74] K. Hormann, U. Labsik, M. Meister, and G. Greiner. Hierarchical extraction of iso-surfaces with semi-regular meshes. In *ACM Solid Modeling and Applications*, pages 53–58, 2002.

[75] B. K. P. Horn and B. G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[76] D. Horn, J. Sugerman, M. Houston, and P. Hanrahan. Interactive k-d tree gpu raytracing. In *Proc. of the 2007 symposium on Interactive 3D graphics and games hardware*, pages 167–174, 2007.

[77] G. Johansson and H. Carr. Accelerating marching cubes with graphics hardware. In *CASCON '06: Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, page 39, New York, NY, USA, 2006. ACM.

[78] S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:103, 2001.

[79] M. Kazhdan. Reconstruction of solid models from oriented point sets. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, page 73. Eurographics Association, 2005.

[80] P. Keller, M. Bertram, and H. Hagen. Multiresolution surface reconstruction from scattered data based on hybrid meshes. In *IASTED VIIP*, pages 616–621, 2005.

[81] P. Keller, M. Bertram, and H. Hagen. Reverse engineering with subdivision surfaces. In *Computing 2007*, pages 127–134, 2007.

[82] P. Keller, O. Kreylos, M. H. Bertram, E. S. Cowgill, L. H. Kellogg, B. Hamann, and H. Hagen. Volumetric surface reconstruction from point clouds using a feature-based approach (to be published).

[83] P. Keller, O. Kreylos, M. Vanco, M. H. Bertram, E. S. Cowgill, L. H. Kellogg, B. Hamann, and H. Hagen. Extracting and visualizing structural features in environmental point cloud lidar data sets (to be published).

[84] R. Kolluri, J. R. Shewchuk, and J. F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 11–21, 2004.

[85] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 82–96, 2002.

[86] P. Krsek, G. Lukács, and R. Martin. Algorithms for computing curvatures from range data. In *The Mathematics of Surface VIII*, pages 1–16. Information Geometers. Winchester, UK, 1998.

[87] K. N. Kutulakos. Shape from the light field boundary. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 53, 1997.

[88] K. N. Kutulakos. Approximate n-view stereo. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part I*, pages 67–83, London, UK, 2000. Springer-Verlag.

[89] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *Int. J. Comput. Vision*, 38(3):199–218, 2000.

[90] P. Lancaster and K. Salauskas. Surfaces generated by moving least squares methods. In *Mathematics of Computation 87*, pages 141–158, 1987.

[91] C. Lange and K. Polthier. Anisotropic smoothing of point sets. *Computer Aided Geometric Design*, 22:680–692, 2005.

[92] S. Lee, G. Wolberg, and S. Shin. Scattered data interpolation with multilevel b-splines. *IEEE - Transactions on Visualization and Computer Graphics*, 3(3):228–244, 1997.

[93] Z. Lei, M. M. Blane, and D. B. Cooper. 3l fitting of higher degree implicit polynomials. 2(4):148–153, 1996.

[94] D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998.

[95] S. Lodha and R. Franke. Scattered data techniques for surfaces. In *Proceedings Dagstuhl Conf. Scientific Visualization*, pages 182–222, 1999.

[96] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. pages 61–62, 1987.

[97] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[98] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pages 674–679, 1981.

[99] M. B. M. Kazhdan and H. Hoppe. Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*, 2006.

[100] H. S. Makoto Kimura and T. Kanade. 3d voxel construction based on epipolar geometry. In *Proceedings of the 1999 International Conference on Image Processing*, volume 3, pages 135 – 139, October 1999.

[101] A. Manessis, A. Hilton, P. Palmer, P. McLauchlan, and X. Shen. Reconstruction of scene models from sparse 3d structure. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2666, 2000.

[102] C. Marie-Paule. *Introduction to implicit surfaces*, chapter Physically based animation. Jules Bloomenthal, 1997. Published under the name Marie-Paule Cani-Gascuel.

[103] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 369–374, 2000.

[104] B. Mederos, L. Velho, and L. H. de Figueiredo. Moving least squares multiresolution surface approximation. In *Sibgraphi*, 2003.

[105] L. V. B. Mederos, N. Amenta, L. Velho, and L. de Figueiredo. Surface reconstruction from noisy point clouds. In *Eurographics Symposium on Geometry Processing*, 2005.

[106] L. V. B. Mederos and L. H. de Figueiredo. Robust smoothing of noisy point clouds. In *SIAM Conference on Geometric Design and Computing*, 2003.

[107] D. D. Morris and T. Kanade. Image-consistent surface triangulation. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1332, 2000.

[108] B. S. Morse, T. S. Yoo, P. Rheingans, D. T. Chen, and K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 78, 2005.

[109] S. Muraki. Volumetric shape description of range data using "blobby model". In *ACM SIGGRAPH Proceedings in Computer Graphics*, pages 227–235. ACM Press, 1991.

[110] S. Muraki. Volumetric shape description of range data using "blobby model". *SIGGRAPH Comput. Graph.*, 25(4):227–235, 1991.

[111] P. J. Narayanan, P. W. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 3, 1998.

[112] A. Nealen. An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation. In *Technical Report, TU Darmstadt*, 2004.

[113] J. Neumann and Y. Aloimonos. Spatio-temporal stereo using multi-resolution subdivision surfaces. *Stereo and Multi-Baseline Vision, IEEE Workshop on*, 0:0103, 2001.

[114] G. Nielson, G. Graf, R. Holmes, A. Huang, and M. Phielipp. Shrouds: optimal separating surfaces for enumerated volumes. In *Eurographics Symposium on Visualization (VisSym)*, pages 75–84, 2003.

[115] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph.*, 22(3):463–470, 2003.

[116] Y. Othake, A. Belyaev, and H.-P. Seidel. A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions. In *Shape Modeling Interantional 2003*, pages 153–161, 2003.

[117] M. Pauly, M. Gross, and L. Kobbelt. Efficient simplification of point sampled surfaces. In *Proc. 13th IEEE Visualization Conference*, pages 163–170, 2002.

[118] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 22(3):281–290, 2003.

[119] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. In *ACM Siggraph*, pages 641–650, 2003.

[120] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives, 2000.

[121] M. Powell. Radial basis functions for multivariate interpolation. In *Algorithms for Approximation of Functions and Data*, pages 143–168, 1987.

[122] V. Pratt. Direct least-squares fitting of algebraic surfaces. *SIGGRAPH Comput. Graph.*, 21(4):145–152, 1987.

[123] R. Ramamoorthi and J. Arvo. Creating generative models from range images. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 195–204, 1999.

[124] A. Razdan and M. Bae. Curvature estimation scheme for triangle meshes using biquadratic Bézier patches. *Computer-Aided Design*, 37(14):1481–1491, 2005.

[125] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *3D Data Processing, Visualization and Transmission*, pages 486–493. IEEE Computer Society, 2004.

[126] S. Rusinkiewicz and M. Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH*, pages 343–352, 2000.

[127] M. Sainz, R. Pajarola, and R. Lario. Points reloaded: Point-based rendering techniques revisted. pages 121–128, 2004.

[128] H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2049, 1999.

[129] L. Schumaker. Fitting surfaces to scattered data. In *Approximation Theory II*, pages 203–268, 1976.

[130] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. *IJCV*, 25(3), 1999.

[131] E. Shaffer and M. Garland. Efficient adaptive simplification of massive meshes. In *Proc. 12th IEEE Visualization Conference*, pages 127–134, 2001.

[132] C. Shen, J. O'Brian, and J. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *In Proc. of ACM SIGGRAPH 2004*, pages 896–904, 2004.

[133] J. Shi and C. Tomasi. Good features to track. In *Proc. of the Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[134] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the Conference on Computer Vision and Pattern Recognition '94*, pages 593–600, 1994.

[135] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. A survey of methods for volumetric scene reconstruction from photographs. pages 81–100, 2001.

[136] G. G. Slabaugh, W. B. Culbertson, T. Malzbender, M. R. Stevens, and R. W. Schafer. Methods for volumetric reconstruction of visual scenes. *Int. J. Comput. Vision*, 57(3):179–199, 2004.

[137] E. Stollnitz, T. DeRose, and D. Salesin. *Wavelets for Computer Graphics–Theory and Applications*. Morgan Kaufmann, 1996.

[138] G. Stylianou and G. Farin. Crest lines for surface segmentation and flattening. *IEEE Transaction On Visualization and Computer Graphics*, 10(5):536–544, Sept. 2004.

[139] R. Szeliski. A multi-view approach to motion and stereo. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1157, 1999.

[140] R. Szeliski. Stereo algorithms and representations for image-based rendering. In *In British Machine Vision Conference*, pages 314–328, 1999.

[141] G. Taubin. An improved algorithm for algebraic curve and surface fitting. *Proceedings of the Fourth International Conference on Computer Vision '93*, 11(14):658–665, 1993.

[142] C. J. Taylor. Surface reconstruction from feature based stereo. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 184, 2003.

[143] W.-S. Tong and C.-K. Tang. Robust estimation of adaptive tensors of curvature by tensor voting. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 27(3):434–449, 2005.

[144] G. Turk and J. F. O'Brien. Shape transformation using variational implicit functions. In *The Proceedings of ACM SIGGRAPH 99*, pages 335–342, 1999.

[145] M. Vančo and G. Brunnett. Direct segmentation of algebraic models for reverse engineering. *Computing*, 72(1-2):207–220, 2004.

[146] M. Vančo and G. Brunnett. Geometric preprocessing of noisy point sets: an experimental study. *Computing*, 79, Special issue on Geometric Modeling, 2007.

[147] T. Várady, M. A. Facello, and Z. Terék. Automatic extraction of surface structures in digital shape reconstruction. *Computer-Aided Design*, 39(5):379–388, 2007.

[148] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):475–480, 2005.

[149] J. Wang, M. Oliveira, and A. Kaufman. Reconstructing manifold and non-manifold surfaces from point clouds. In *IEEE Visualization*, pages 415–422, 2005.

[150] N. Werghi, A. Ashbrook, R. B. Fisher, and C. Robertson. Faithful recovering of quadric surfaces from 3d range data. In *2nd International Conference on 3D Digital Imaging and Modeling*, pages 280–289. IEEE Computer Society, 1999.

[151] N. Werghi, R. Fisher, C. Robertson, and A. Ashbrook. Modeling Objects having Quadric Surfaces Incorporating Geometric Constraints. In *5th European Conference on Computer Vision-Volume II*, pages 185–201. Springer-Verlag, 1998.

[152] T. Weyrich, M. Pauly, R. Keiser, S. Heinzle, S. Scandella, and M. Gross. Post-processing of scanned 3d surface data. In *Proceedings of Eurographics Symposium on Point-Based Graphics 2004*, pages 85–94, 2004.

[153] R. Whitaker. A level-set approach to 3d reconstruction from range data. *International Journal of Computer Vision*, 29(3):203–231, 1998.

[154] Z. Wood, M. Desbrun, P. Schroeder, and D. Breen. Semi-regular mesh extraction from volumes. In *IEEE Visualization*, pages 275–282, 2000.

[155] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Fast and robust detection of crest lines on meshes. In *ACM Symposium on Solid and physical modeling*, pages 227–232. ACM Press, 2005.

[156] W. Zhang, Z. Tang, and J. Li. Adaptive hierarchical b-spline surface approximation of large-scale scattered data. In *In Proc. Pacific Graphics '98*, pages 8–16, 1998.

[157] Y. Zhang and C. Kambhamettu. On 3d scene flow and structure estimation. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:778, 2001.

[158] H.-K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. In *IEEE Proceedings in Variational and Level Set Methods in Computer Vision*, pages 194–201. IEEE Computer Society, 2001.

[159] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: an interactive system for point-based surface editing. In *ACM Siggraph*, pages 322–329, 2002.

Curriculum Vitae

Name:	Keller, Patric
Date of birth:	08/17/1980
Place of birth:	Dahn

Education

1986 - 1991	Elementary School Fischbach bei Dahn
1991 - 1996	Public School in Dahn
1996 - 1999	Berufliches Gymnasium Wirtschaft, Pirmasens **Certificate:** General qualification for University entrance (Abitur)
1999 - 08/2005	Study of Computer Science at the University of Kaiserslautern, Germany **Master Thesis:** "Adaptive surface reconstruction from scattered data" **Degree:** Diplom-Informatik (Dipl.-Inf.)
04/2006 - 11/2009	PhD Student at University of Kaiserslautern Since 01/2007: Graduate Student at the International Research Training Group (IRTG) "Visualization of Large and Unstructured Data Sets - Applications in Geospacial Planning, Modeling and Engineering"
2008 & 2009	Research visits at the Institute for Data Analysis and Visualization (IDAV) at UC Davis, CA, USA

Work Experience

02/2007 - 08/2009	Researcher at the Fraunhofer Institute für Techno- und Wirtschaftsmathematik (ITWM) in Kaiserslautern, Germany
Since 09/2009	Member of the research staff at the Dept. of Computer Science at the University of Kaiserslautern, Germany