

Institut für Geodäsie und Geoinformation  
Bereich Photogrammetrie

---

# Man-made Surface Structures from Triangulated Point Clouds

## **Inaugural-Dissertation**

zur  
Erlangung des Grades  
Doktor-Ingenieur  
(Dr.-Ing.)  
der  
Hohen Landwirtschaftlichen Fakultät  
der  
Rheinischen Friedrich-Wilhelms-Universität  
zu Bonn

vorgelegt am 30. September 2013 von  
**Falko Schindler**  
aus Leipzig

Referent: Prof. Dr.-Ing. Dr. h.c. mult. Wolfgang Förstner  
Korreferenten: Prof. Dr. Wolf-Dieter Schuh  
Prof. Dr. Konrad Schindler  
Tag der mündlichen Prüfung: 25. November 2013  
Erscheinungsjahr: 2013

Diese Dissertation ist auf dem Hochschulschriftenserver der ULB Bonn elektronisch publiziert ([http://hss.ulb.uni-bonn.de/diss\\_online](http://hss.ulb.uni-bonn.de/diss_online)).

---

## Zusammenfassung

---

### **Künstliche Oberflächenstrukturen aus triangulierten Punktwolken**

Ein Ziel der Photogrammetrie ist die Rekonstruktion der Form und Größe von Objekten, die mit Kameras, 3D-Laserscannern und anderen räumlichen Erfassungssystemen aufgenommen wurden. Während viele Aufnahmetechniken innerhalb von Sekunden triangulierte Punktwolken mit Millionen von Punkten liefern, ist deren Interpretation gewöhnlicherweise dem Nutzer überlassen. Besonders bei der Rekonstruktion künstlicher Objekte (i.S.v. engl. *man-made* = „von Menschenhand gemacht“) ist man an der zugrunde liegenden Oberflächenstruktur interessiert, welche nicht inhärent in den Daten enthalten ist. Diese umfasst die geometrische Form des Objekts, z.B. quaderförmig oder zylindrisch, als auch die zugehörigen Oberflächenparameter, z.B. Breite, Höhe oder Radius. Die Anwendungen sind vielfältig und reichen von industriellen Fertigungskontrollen über architektonische Raumaufmaße bis hin zu großmaßstäbigen Stadtmodellen.

Das Ziel dieser Arbeit ist es, solche Oberflächenstrukturen automatisch aus triangulierten Punktwolken von künstlichen Objekten abzuleiten. Sie sind definiert als ein Verbund ebener und gekrümmter geometrischer Primitive. Modellwissen über typische Primitive und Relationen zwischen Paaren von ihnen soll die Rekonstruktion positiv beeinflussen.

Nachdem wir ein parametrisiertes Modell für künstliche Oberflächenstrukturen formuliert haben, entwickeln wir ein Rekonstruktionsverfahren mit drei Verarbeitungsschritten: Im Rahmen einer schnellen Vorsegmentierung, die lokale Oberflächeneigenschaften berücksichtigt, teilen wir die gegebene vermaschte Oberfläche in ebene Regionen. Unter Verwendung eines Schemas zur Modellauswahl, das auf der Minimierung der Beschreibungslänge beruht, ist diese Oberflächensegmentierung unabhängig von Kontrollparametern und liefert automatisch eine optimale Anzahl an Regionen. Eine anschließende Verbesserung führt eine Menge von ebenen und gekrümmten geometrischen Primitiven ein und fusioniert benachbarte Regionen hierarchisch basierend auf ihrer gemeinsamen Beschreibungslänge. Eine globale Klassifikation und bedingte Parameterschätzung verbindet die datengetriebene Segmentierung mit hochrangigem Modellwissen. Dazu stellen wir die Oberflächenstruktur in Form eines graphischen Modells dar und formulieren Faktoren basierend auf der Likelihood sowie auf apriori Wissen über die Parameterverteilungen und Klassenwahrscheinlichkeiten. Wir leiten die wahrscheinlichste Konfiguration von Flächen- und Relationsklassen mit Hilfe von Belief-Propagation ab und schätzen eine optimale Oberflächenparametrisierung mit Bedingungen, die durch die Relationen zwischen benachbarten Primitiven induziert werden. Der Prozess ist eigens für verrauschte Daten mit Ausreißern

und wenigen Ausnahmeregionen konzipiert, die nicht durch geometrische Primitive beschreibbar sind. Er liefert wasserdichte 3D-Oberflächenstrukturen mit Oberflächenprimitiven verschiedener Art.

Die Leistungsfähigkeit des vorgestellten Verfahrens wird an verschiedenen Datensätzen experimentell evaluiert. Auf kleinen, synthetisch generierten Oberflächen untersuchen wir die Genauigkeit der geschätzten Oberflächenparameter, die Sensitivität bzgl. verschiedener Eigenschaften der Eingangsdaten und bzgl. Modellannahmen sowie die Rechenkomplexität. Außerdem demonstrieren wir die Flexibilität bzgl. verschiedener Aufnahmetechniken anhand realer Datensätze. Das vorgestellte Rekonstruktionsverfahren erweist sich als genau, hinreichend schnell und wenig anfällig für Defekte in den Daten oder falsche Modellannahmen.



### **Man-made Surface Structures from Triangulated Point Clouds**

Photogrammetry aims at reconstructing shape and dimensions of objects captured with cameras, 3D laser scanners or other spatial acquisition systems. While many acquisition techniques deliver triangulated point clouds with millions of vertices within seconds, the interpretation is usually left to the user. Especially when reconstructing man-made objects, one is interested in the underlying surface structure, which is not inherently present in the data. This includes the geometric shape of the object, e.g. cubical or cylindrical, as well as corresponding surface parameters, e.g. width, height and radius. Applications are manifold and range from industrial production control to architectural on-site measurements to large-scale city models.

The goal of this thesis is to automatically derive such surface structures from triangulated 3D point clouds of man-made objects. They are defined as a compound of planar or curved geometric primitives. Model knowledge about typical primitives and relations between adjacent pairs of them should affect the reconstruction positively.

After formulating a parametrized model for man-made surface structures, we develop a reconstruction framework with three processing steps: During a fast pre-segmentation exploiting local surface properties we divide the given surface mesh into planar regions. Making use of a model selection scheme based on minimizing the description length, this surface segmentation is free of control parameters and automatically yields an optimal number of segments. A subsequent refinement introduces a set of planar or curved geometric primitives and hierarchically merges adjacent regions based on their joint description length. A global classification and constraint parameter estimation combines the data-driven segmentation with high-level model knowledge. Therefore, we represent the surface structure with a graphical model and formulate factors based on likelihood as well as prior knowledge about parameter distributions and class probabilities. We infer the most probable setting of surface and relation classes with belief propagation and estimate an optimal surface parametrization with constraints induced by inter-regional relations. The process is specifically designed to work on noisy data with outliers and a few exceptional freeform regions not describable with geometric primitives. It yields full 3D surface structures with watertightly connected surface primitives of different types.

The performance of the proposed framework is experimentally evaluated on various data sets. On small synthetically generated meshes we analyze the accuracy of the estimated surface parameters, the sensitivity w.r.t. various properties

of the input data and w.r.t. model assumptions as well as the computational complexity. Additionally we demonstrate the flexibility w.r.t. different acquisition techniques on real data sets. The proposed method turns out to be accurate, reasonably fast and little sensitive to defects in the data or imprecise model assumptions.

---

# Contents

---

<b>Notation</b>	<b>11</b>
<b>1 Introduction</b>	<b>13</b>
1.1 Motivation . . . . .	14
1.2 Goal and achievements . . . . .	15
1.3 Related work . . . . .	16
1.3.1 Surface segmentation . . . . .	16
1.3.2 Object reconstruction . . . . .	19
1.4 General concept . . . . .	22
1.5 Organization of the thesis . . . . .	22
<b>2 Theoretical background</b>	<b>25</b>
2.1 Triangular meshes . . . . .	26
2.1.1 Point cloud triangulation . . . . .	27
2.1.2 Surface normals and curvature . . . . .	29
2.2 Shortest paths on surfaces . . . . .	34
2.2.1 Dijkstra's algorithm . . . . .	34
2.2.2 Fast marching method . . . . .	35
2.3 Parameter estimation in a Gauss-Helmert model . . . . .	36
2.3.1 Maximum-likelihood and least-squares estimation . . . . .	37
2.3.2 Homogeneous parameter vectors . . . . .	37
2.3.3 Homogeneous observation vectors . . . . .	39
2.3.4 Constraints between observations only . . . . .	39
2.3.5 Outlier re-weighting . . . . .	40
2.4 Model selection via description length minimization . . . . .	40
2.4.1 Points without model . . . . .	42
2.4.2 Points on one plane . . . . .	42
2.4.3 Points on multiple planes . . . . .	43
2.4.4 Outliers . . . . .	43
2.5 Inference in graphical models . . . . .	44
2.5.1 Factor graphs . . . . .	44
2.5.2 Marginal probability via the sum-product algorithm . . . . .	46
2.5.3 Most probable variable setting via the max-sum algorithm . . . . .	47

<b>3</b>	<b>Concept for reconstructing models of man-made surfaces</b>	<b>51</b>
3.1	Task specification . . . . .	51
3.1.1	Type of input data . . . . .	52
3.1.2	Assumptions about the surface triangulation . . . . .	52
3.1.3	Objective of the proposed reconstruction framework . . . . .	54
3.1.4	Criteria for the experimental evaluation . . . . .	55
3.2	Concept . . . . .	56
3.2.1	Model for man-made surface structures . . . . .	56
3.2.2	Reconstruction framework at three levels . . . . .	60
<b>4</b>	<b>Segmentation via Dijkstra farthest point sampling (FPS)</b>	<b>63</b>
4.1	Surface segmentation via farthest point sampling . . . . .	63
4.2	Advanced sampling strategies . . . . .	66
4.2.1	Incremental-decremental sampling . . . . .	67
4.2.2	Automatic stopping criterion . . . . .	69
4.2.3	Refinement with Lloyd iterations . . . . .	72
4.3	Advanced distance metrics . . . . .	74
4.3.1	Non-planar regions . . . . .	75
4.3.2	Extrinsic distance metrics . . . . .	76
<b>5</b>	<b>Description length minimizing hierarchical face clustering</b>	<b>79</b>
5.1	Hierarchical face clustering (HFC) . . . . .	80
5.2	Minimizing the description length . . . . .	81
5.2.1	Description length reduction when merging two regions . . . . .	82
5.2.2	Description length reduction when diffusing vertices . . . . .	82
5.2.3	Decision strategy . . . . .	83
5.3	Regularizing the boundary length . . . . .	85
5.4	Different surface types . . . . .	86
5.4.1	Planar surfaces . . . . .	86
5.4.2	Quadratic surfaces . . . . .	86
5.4.3	Freeform surfaces . . . . .	88
5.5	Combination of MDL-based HFC and DijkstraFPS . . . . .	89
<b>6</b>	<b>Surface model reconstruction</b>	<b>91</b>
6.1	Factor graph of surface structures . . . . .	91
6.2	Unary factors . . . . .	93
6.2.1	Likelihood of points given the parametrization . . . . .	94
6.2.2	Parameter distribution given a surface class . . . . .	95
6.2.3	Surface priors . . . . .	97
6.3	Ternary factors . . . . .	98
6.3.1	Joint parameter distribution given a relation class . . . . .	99
6.3.2	Relation priors . . . . .	101
6.4	Classification using the max-sum algorithm . . . . .	101
6.5	Surface parameter estimation . . . . .	103
<b>7</b>	<b>Empirical evaluation</b>	<b>107</b>
7.1	Demonstration of the complete algorithm . . . . .	108
7.2	Experiments with synthetic data . . . . .	110
7.2.1	Accuracy of surface parameters . . . . .	111
7.2.2	Sensitivity w.r.t. various data properties . . . . .	112

---

7.2.3	Sensitivity w.r.t. model assumptions . . . . .	115
7.2.4	Computational complexity . . . . .	118
7.3	Experiments with real data . . . . .	120
7.3.1	Accuracy analysis on calibration objects . . . . .	120
7.3.2	Flexibility w.r.t. different acquisition techniques . . . . .	121
7.3.3	Sensitivity w.r.t. model assumptions . . . . .	131
7.3.4	Computational complexity . . . . .	134
<b>8</b>	<b>Conclusion</b>	<b>137</b>
8.1	Summary . . . . .	137
8.2	Outlook . . . . .	139
<b>A</b>	<b>Gauss-Helmert model for estimating surface primitives</b>	<b>141</b>
A.1	Initialization . . . . .	141
A.2	Derivatives of the functional model . . . . .	143
	<b>List of Figures</b>	<b>147</b>
	<b>List of Tables</b>	<b>149</b>
	<b>List of Algorithms</b>	<b>151</b>
	<b>Bibliography</b>	<b>153</b>



---

## Notation

---

### Geometry

$\mathcal{X}, \mathcal{V}, \mathcal{T}$	point, vertex, triangle
$u, v \in \{1, \dots, V\}$	vertex indices
$t \in \{1, \dots, T\}$	triangle indices
$\text{ne}(\cdot)$	set of neighboring vertices, triangles or graph nodes
$\mathbf{X} = [X, Y, Z]^T$	Euclidean 3D coordinate vector
$(x, y), (u, v)$	2D parametrizations for surfaces
$F_1, F_2, \mathbf{n}, \kappa$	first/second fundamental forms, normal, curvature
$ \cdot $	magnitude, norm, determinant or cardinality
$\text{tr}(\cdot)$	trace
$\mathbf{a}$	coefficients of a local graph surface
$M, U, D, V$	moment matrix and its SVD: $M = UDV^T$

### Sampling density

$\gamma$	curve
lfs	local feature size
$\epsilon$	sampling density

### Intrinsic distances

$\mathbf{D} = [D_v]$	distance map, one value per vertex $v$
$\mathbf{F} = [F_v]$	friction map, one value per vertex $v$
$d_u^v, s_u^v$	curvature-adaptive/Euclidean distance from $u$ to $v$
$\nabla \cdot$	gradient
$v_0, v^*$	seed vertex, farthest vertex
$\mathcal{Q}$	queue of vertices in wave front
$O(\cdot)$	big O notation: asymptotic computational complexity

### Vertex labeling

$\mathbf{l} = [l_v]$	labeling for each vertex $v$
$l \in \{1, \dots, L\}$	possible labels
$L_{\text{inc}}, L_{\text{dec}}$	$L$ after incremental/decremental segmentation
$l^+, l^-$	new label, label subject to removal

### Description length

$\Phi$	description length
lb, ln, log	binary, natural and general logarithm
$r, \epsilon$	coordinate range and resolution
$B$	boundary length
$\bar{\cdot}$	outlier

**Surface model**

$\boldsymbol{\theta}$	parameters of one surface region
$\boldsymbol{\theta}_n, \boldsymbol{\theta}_c, \theta_d, \theta_r, \theta_A$	normal, centroid, distance to origin, radius, slope
$\boldsymbol{\theta}_{L_h}, \boldsymbol{\theta}_{L_0}$	homogeneous and Euclidean part of Plücker line $\mathbf{L}$
$z, \alpha$	zenith distance of a normal, opening angle of a cone
$t$	threshold

**Graphical models**

$\mathbf{x} = \{x_i\}$	variable nodes
$\mathbf{f} = \{f_j\}$	factor nodes
$i = 1, \dots, I$	variable indices
$j = 1, \dots, J$	factor indices
$\mu, \pi$	forward and backward messages

**Classification**

$P, p, \phi$	probability, probability density, potential
$\mathcal{D}$	given data
$Z$	partition function
$\mathbf{s} = [s_l]$	surface classes for each region
$\mathbf{r} = [r_k]$	relation classes for each pair of adjacent regions
$s \in \{1, \dots, S\}$	possible surface classes
$r \in \{1, \dots, R\}$	possible relation classes
$l = 1, \dots, L$	region indices
$k = 1, \dots, K$	relation indices
$\mathcal{N}(\cdot), F(\cdot), \chi^2(\cdot)$	normal, Fisher and chi-squared distribution
$\Phi(\cdot)$	CDF of the standard normal distribution
$\nu$	degrees of freedom
$\delta$	contradiction angle
$H_0, H_1$	null and alternative hypothesis
$q$	test statistic
$E(\cdot)$	expectation

**Parameter estimation**

$N, U, G$	number of observations, parameters and constraints
$R$	redundancy
$\sigma^2, \boldsymbol{\Sigma}, W$	variance, covariance matrix, weight matrix
$\mathbf{y}, \mathbf{v}, \mathbf{p}, \mathbf{g}$	observations, residuals, parameters, constraints
$\mathbf{c}_g$	residuals of the constraints
$A, \hat{B}$	Jacobians of constraints w.r.t. parameters/observations
$J$	Jacobian of transformation into reduced tangent space
$N(\cdot)$	vector normalization
$\text{sgn}, \text{null}(\cdot), I_n$	sign, null space, $n \times n$ identity matrix
$N, \mathbf{b}$	normal equation matrix and right-hand side
$\Omega$	sum of squared residuals
$X_v$	sum of squared coordinate residuals for point $v$
$\tilde{\cdot}, \hat{\cdot}, \cdot_0$	true, estimated, approximate value
$\Delta \cdot$	difference w.r.t. approximate values
$\cdot_r$	vector reduced to tangent space
$w(\cdot), \rho(\cdot)$	weight function, re-weighting function



# CHAPTER 1

---

## Introduction

---

The core task of photogrammetry has always been to determine the shape and location of visually observed objects. Techniques for detecting individual points in images and forward-intersecting their 3D location are established since decades and have recently been applied to data sets with millions of images and billions of points exploiting parallel computing techniques. Novel sensors for acquiring depth images in real-time complement the capabilities of modern photogrammetric systems. Against the backdrop of large amounts of data captured with low effort on the one hand and advanced methods from the field of machine learning and pattern recognition on the other hand, the problem of automatically recognizing an abstract object or surface model gains in importance. This thesis proposes a novel framework for reconstructing such surface models from unordered point clouds of man-made objects. It serves as an essential preprocessing step for subsequent interpretations and analyses.

---

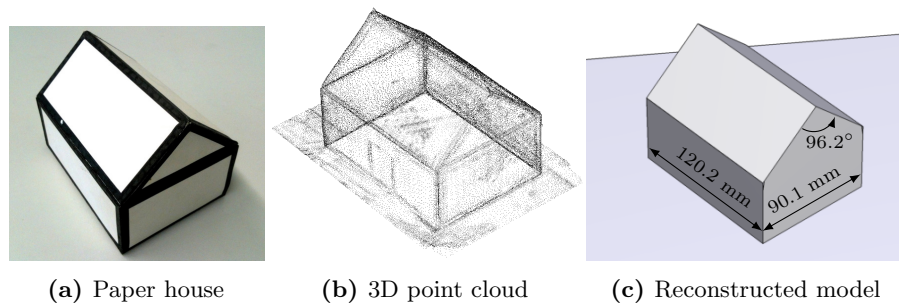
<b>1.1</b>	<b>Motivation . . . . .</b>	<b>14</b>
<b>1.2</b>	<b>Goal and achievements . . . . .</b>	<b>15</b>
<b>1.3</b>	<b>Related work . . . . .</b>	<b>16</b>
1.3.1	Surface segmentation . . . . .	16
1.3.1.1	Segmentation via distance transforms . . . . .	16
1.3.1.2	Hierarchical mesh simplification . . . . .	17
1.3.1.3	Primitive fitting . . . . .	18
1.3.1.4	Watershed segmentation . . . . .	18
1.3.2	Object reconstruction . . . . .	19
1.3.2.1	Labeling cell complexes . . . . .	20
1.3.2.2	Knowledge-based 3D building reconstruction . . . . .	20
1.3.2.3	Roofs and large-scale city models . . . . .	21
<b>1.4</b>	<b>General concept. . . . .</b>	<b>22</b>
<b>1.5</b>	<b>Organization of the thesis. . . . .</b>	<b>22</b>

---

## 1.1 Motivation

Recognizing and measuring 3D objects has always been a major field of research in geodesy, photogrammetry and computer vision. Originally the focus used to be on precisely *measuring* individual object dimensions like width and length of a rectangular building footprint. In contrast, recent approaches try to *recognize* the rectangular shape automatically from a large bulk of data like images or laser scans, before estimating relevant object dimensions.

Fig. 1.1 illustrates the relation between the 3D point cloud of a small man-made object and the corresponding surface model. While the point cloud does not contain any semantical information like the type of surface each point represents or which point corresponds to sharp edges or corners, the surface model consists of multiple connected, parametrized surface regions of a certain type. Surface properties and inter-surface relations, e.g. the orthogonality of two adjacent planes, are explicitly encoded and allow the derivation of measures like the distance of two parallel planes or the angle between adjacent surface parts. High-level semantics, however, like a plane being a wall, window or part of the roof, are not contained in the surface model and are not within focus of this thesis.



**Figure 1.1:** Surface structure of a small object. A paper house (a) is captured using a laser scanner yielding a set of unordered 3D points (b). The reconstructed surface model (c) contains both a segmentation into piece-wise planar surface regions as well as a parametrization for each region. This allows to automatically derive measures like the object's dimensions or building-specific properties like the gable angle.

Applications of automatic shape recognition approaches are manifold:

**Reverse engineering and production control.** In the field of industrial design and production the method of reverse engineering has become popular. It aims at recovering a 3D virtual model of an existing mechanical object like a part of an engine. Similarly, one tries to reconstruct the geometry of manufactured objects for quality control.

**Architecture and craft.** Both architects and craftsman rely on floor plans of rooms, apartments or whole buildings. While it is time consuming and error-prone to capture the topology and to measure width and height of each wall individually, an automated shape recognition and measuring system, e.g. based on laser scanned point clouds, would simplify and accelerate this process.

**City models.** Much effort is put into creating 3D city models from aerial laser scans, from aerial imagery or from mobile mapping systems. While some mapping services rely on user-driven building modelling, other services try to reconstruct buildings automatically.

All above-mentioned applications concentrate on man-made objects due to their common and usually relatively simple structure: Most man-made surfaces are a compound of multiple planar or curved surfaces. In contrast, natural objects like terrain or vegetation are far more complex in topology and geometry and thus require specially tailored approaches and specific model knowledge. In this thesis man-made objects and the reconstruction of their underlying surface structure is at the center of our attention.

## 1.2 Goal and achievements

The goal of this thesis is as follows. Given a set of triangulated 3D points representing an object's surface, we want to develop a framework for automatically recognizing and parametrizing the underlying surface structure.

The surface structure should comprise multiple surface regions of prespecified types and relations between pairs of adjacent, watertightly connected regions. Furthermore, we want to estimate a parametrization for each region such that the individual surfaces are close to their corresponding 3D points.

Many 3D data acquisition methods yield unordered point clouds, i.e. without known connectivities between neighboring points. However, well established algorithms exist for triangulating such point clouds, also referred to as surface reconstruction, shape reconstruction or meshing. We will discuss some of them in Section 2.1.1. Although existing approaches do not yield perfect results in the presence of noisy data, low sampling density or holes, we will choose one of them for preprocessing our reconstruction framework that best matches our requirements.

Before briefly describing the general concept of the proposed framework in Section 1.4, we collect relevant related work from the field of surface segmentation and object reconstruction in the following section.

**Author publications.** The contents of this thesis have been partly published in the following articles and conference proceedings:

**Schindler and Förstner (2011)** Fast marching for robust surface segmentation. In U. Stilla, F. Rottensteiner, H. Mayer, B. Jutzi, and M. Butenuth (Eds.), *Photogrammetric Image Analysis*, Volume 6952 of *Lecture Notes in Computer Science*, pp. 147–158. Springer Berlin Heidelberg.

**Schindler, Förstner, and Frahm (2011)** Classification and reconstruction of surfaces from point clouds of man-made objects. In *IEEE International Conference on Computer Vision, Workshop on Computer Vision for Remote Sensing of the Environment*.

**Schindler and Förstner (2013)** DijkstraFPS: Graph partitioning in geometry and image processing. *Zeitschrift für Photogrammetrie, Fernerkundung und Geoinformation 2013* (4), 285–296.

## 1.3 Related work

Previous work on the topic of surface structure recognition and estimation falls into two groups: 1) mere segmentation, i.e. dividing the given surface mesh into homogeneous regions, and 2) reconstruction, i.e. determining a parametric representation of the underlying object or surface. We will discuss one group after the other in the two following sections.

### 1.3.1 Surface segmentation

Under surface segmentation we understand the complete partitioning of a given implicit or meshed surface into non-overlapping regions that fulfill a specific homogeneity criterion. We distinguish segmentations as a preprocessing step for subsequent interpretation and semantical segmentations that integrate the interpretation of surface regions. Within this thesis we combine both concepts: After a data-driven pre-segmentation we refine the partitioning using a semantical segmentation including the assignment of different surface types.

In this section we discuss four groups of surface segmentation techniques: There is a group of segmentation strategies based on *distance transforms* on surfaces. Many approaches are based on *hierarchically* simplifying a mesh, starting with one region per vertex or triangle and merging them iteratively. Another common concept is to *detect geometrical primitives*, yielding a segmentation implicitly. Finally, some approaches are based on the *watershed* image segmentation, which can be understood as region-growing followed by an iterative merging step. We will highlight relevant publications from each of these four groups and draw conclusions for our work.

An overview of more mesh segmentation algorithms is given by Attene et al. (2006) and Wilke (2002, Ch. 6). Chen et al. (2009) provide a benchmark and various metrics for comparing mesh segmentations. Cignoni et al. (1998) and Garland (1999a) give an extensive overview of the related concept of mesh *simplification*.

#### 1.3.1.1 Segmentation via distance transforms

We first look into surface segmentation algorithms based on distance transforms. The principle is to initialize small regions at seed vertices and to propagate the region boundaries to the outside, until the whole mesh is segmented. Since the selection of all seeds at once is difficult at the beginning, one starts with one random seed only and iteratively samples the farthest seed according to a predefined distance metric, motivating the term *farthest point sampling* (FPS). In this section we collect relevant publications about FPS, the foundation of the surface segmentation introduced in Chapter 4.

Originally FPS is used for progressive image sampling (Eldar et al., 1997). Moenning and Dodgson (2003a) adopt FPS for sub-sampling meshed surfaces, using the *fast marching method* (FMM, Sethian, 1996) for computing the distance map that is needed for finding the farthest point. Using an approximation from Memoli and Sapiro (2001, 2002), Moenning and Dodgson (2003b) do not rely on meshed surface, but can apply FPS to implicit surfaces and unordered point clouds as well. While FPS is usually realized using a uniform or curvature-based distance metric, Lai et al. (2008) formulate a distance metric for FPS based on

random walks, i.e. evaluating the probability of a random walk traveling along a certain path.

Schindler and Förstner (2011, 2013) recently proposed DijkstraFPS, a segmentation algorithm for meshed surfaces. It is based on FPS as well, but updates the distance map using Dijkstra’s algorithm (Dijkstra, 1959), which is a fast and simple approximation of FMM. By combining the incremental sampling with a decremental removal of surface regions, the algorithm yields more stable region boundaries even on noisy data. Additionally, Schindler and Förstner (2011) proposed an automatic stopping criterion to automatically determine the optimal number of surface regions.

A related concept from Cohen-Steiner et al. (2004) randomly initializes multiple seeds at once and gradually grows corresponding regions. In contrast to the above-mentioned approaches, it uses shape proxies, i.e. fitted primitives that can be flat or curved. The authors also suggest the incremental insertion and deletion of regions as well as the combination of both.

We will describe DijkstraFPS in Chapter 4 and use it as a fast initialization for a subsequent hierarchical surface segmentation in Chapter 5. In the following section we will discuss related work on this group of hierarchical mesh segmentation algorithms.

### 1.3.1.2 Hierarchical mesh simplification by merging primitives

A second group of surface segmentation algorithms is based on a hierarchical structure, obtained via split or merge operations. The process is guided by an error metric w.r.t. fitted primitives. While *splitting* 1D polylines is easy (Ramer, 1972; Douglas and Peucker, 1973; Pan and Förstner, 1994), finding optimal splits is much more difficult for 2D surfaces. Therefore, we concentrate on approaches based on *merging* adjacent regions. In this section we will discuss the original approach for planar regions from Garland et al. as well as the extension for curved regions from Attene et al.

The fundamental idea behind hierarchical mesh simplification is to start with a complete oversegmentation, e.g. one region for each vertex or triangle, and to merge them iteratively according to a predefined objective function. Garland and Heckbert (1997) propose a mesh simplification algorithm that iteratively contracts vertex pairs reducing the number of vertices while trying to keep the approximation error small. The dual approach is to iteratively merge triangles leading to a surface segmentation (Garland et al., 2001). Both concepts are summarized in Garland’s PhD thesis (Garland, 1999b). We will adopt the concept of hierarchical mesh segmentation in Chapter 5.

Attene et al. (2006) as well as Attene and Patanè (2010) extend the segmentation approach to different primitives such as planes, cones, spheres, cylinders and tori. Together with mesh repairing (Attene, 2010) and other low-level mesh processing operations this is part of the software ReMESH (Attene and Falcidieno, 2006). We will as well introduce planes, cones, cylinders and spheres. In contrast to Garland et al. and Attene et al., who require the number of regions to be specified by the user, we will formulate an optimization function that automatically yields an optimal number of regions.

The hierarchical segmentation approach is well suited for reconstructing man-made surface structures due to the variety of primitives that can be used. One drawback might be the large number of trivial merge operations in the

beginning: Up to 50 % of all merge steps are required to form pairs of triangles; after 25 % more steps we obtain groups of four. On such a small scale it is unnecessary to fit multiple primitives each time. Therefore, we will use a hierarchical approach based on fitting primitives only as second, refining surface segmentation (Section 5.5).

### 1.3.1.3 Primitive fitting

Many mesh segmentation methods focus on fitting primitives into the data points, often without using a triangular mesh but unordered points only. The segmentation is usually derived by assigning the points to their closest primitive. After briefly addressing a state-of-the-art method from Schnabel et al. (2007, 2009) and an extension from Li et al. (2011) involving regularities between different primitives, we collect strategies with hybrid representations: partly described with primitives, partly with the original mesh.

The popular approach from Schnabel et al. (2007) detects primitives based on random sampling consensus (RANSAC, Fischler and Bolles, 1981). The authors detect planes, spheres, cylinders, cones and tori very efficiently with three or less points per minimal set using corresponding normal vectors. In a subsequent processing step they derive a watertight surface built from primitives by solving a discrete graph optimization (Schnabel et al., 2009) based on a 3D voxel grid. Although obtaining visually pleasing results, this approach ignores the connectivity information contained in a triangular mesh and needs a post-processing step to reconnect primitives to watertight surface models. The latter involves solving a *graph cut* (Boykov and Kolmogorov, 2004) on a voxel grid, which does not scale well with increasing object size.

After fitting primitives into unordered point clouds, Li et al. (2011) detect regularities, i.e. common angles or distances, in the set of primitives. They re-estimate all surface parameters using constraints induced by the detected regularities. Since the authors analyze relations between primitive pairs, the complexity can reach  $O(L^4)$  for  $L$  primitives, becoming intractable for large models. We will detect primitive relations as well, but within a global classification procedure taking data and model uncertainties into account.

Regions without a parametric representation, i.e. so-called freeform surfaces, deserve special attention. Gallup et al. (2010) classify a given color image into planar and non-planar regions and model both classes differently. Lafarge et al. (2010) reconstruct a hybrid representation by trying to fit planes and curved primitives first; if both attempts fail, they keep the original mesh structure. Furthermore, the reconstruction is optimized w.r.t. photo consistency (Lafarge et al., 2012). Although not part of our understanding of man-made structures in a strict sense, our reconstruction framework will be able to model freeform surfaces as well.

In order to obtain an inherently watertight segmentation, we will not globally search for primitives but couple the primitive detection with the segmentation strategies previously discussed in Sections 1.3.1.1 and 1.3.1.2.

### 1.3.1.4 Watershed segmentation

Watershed segmentation is an image segmentation method, which has been successfully transferred to 3D meshes. After briefly describing the original watershed

principle for 2D domains, we discuss analogue approaches for mesh segmentation and evaluate the suitability for our application.

Watersheds, originally a term from the field of hydrology and geomorphology, denote a net of boundary lines that segment an area into multiple drainage basins. In image processing it is a common segmentation method (Beucher and Lantuejoul, 1979): By virtually flooding the image starting from local minima of a precomputed height function, e.g. the magnitude of local image gradient, one merges basins until reaching a certain height, which needs to be given as user-defined threshold. Meine and Köthe (2005) avoid the need for such a height threshold by transferring the discrete problem into a continuous form via spline interpolation. Instead, however, they require a user-defined integration scale.

Mangan and Whitaker (1999) segment a surface by adapting the watershed image segmentation to triangulated meshes in 3D space. As height function they use local curvature. Pulla et al. (2001) propose multiple alternative curvature measures for Mangan and Whitaker’s algorithm. Page et al. (2003) and Page (2003) obtain comparable results, although they define watersheds by simply applying a threshold on local curvature, detect connected components and fill unlabeled watersheds with a region-growing approach towards the region boundaries. Atkar et al. (2005) exploit the watershed segmentation method for segmenting computer aided design (CAD) models for the application of path-planning of auto-body painting robots.

On noisy data, however, watershed segmentation tends to yield poor results. It relies on a height function, which is usually the curvature and sensitive to noise. We will face the same problem within our DijkstraFPS segmentation approach presented in Chapter 4, but will introduce several strategies to nevertheless obtain robust results. Furthermore, we will use the curvature-based DijkstraFPS segmentation only as an initialization for a subsequent primitive-based segmentation (Chapter 5).

A second drawback of the watershed segmentation is its dependence on a predefined, more or less intuitive threshold. Our surface segmentation will be completely parameterless and only depending on an assumption about the underlying noise.

This concludes the collection of algorithms and concepts on pure surface segmentation, which is an important first step of our reconstruction framework.

### 1.3.2 Object reconstruction

In the previous section we discussed concepts mainly for segmenting surfaces or point clouds. Although some approaches involve fitting geometric primitives like planes or quadratic surfaces, relations across multiple regions are usually not considered. In this section we want to focus on approaches for reconstructing objects in terms of a collection of connected surface regions with certain pre-specified relations in between. We will first bring up the concepts of labeling *cell complexes* and *topological constraints* for 3D building models, before giving examples for reconstructing roofs and large-scale *city models* – mainly originated by the photogrammetric community.

### 1.3.2.1 Labeling cell complexes

A *cell complex* is a topological concept, basically dividing the space into cells (Frank and Kuhn, 1986). By labeling each cell as inside and outside, the union of all boundaries between differently labeled cells represents the object’s surface.

The complex can, e.g., be induced by a binary space partition tree: Starting with one large cell, Labatut et al. (2009) and Labatut (2009) detect a dominant primitive with a RANSAC approach and split the current cell in two parts, one of each side of the primitive. This process is iterated as long as primitives can be found. Similarly, Chauve et al. (2010) build a mixture of a binary space partition and a voxel grid via region growing, but are limited to planar primitives only.

In both cases an optimal labeling and thus the object’s surface is found using a common graph segmentation approach called *graph cut* (Boykov and Kolmogorov, 2004). Assuming the viewpoint to be known for each point, the graph cut is solved using visibility constraints: All cells between a 3D data point and its corresponding viewpoint must be empty. The remaining cells are supposed to be part of the object and, thus, their hull defines the object’s surface.

Although the authors present very promising results, such binary space partition trees can become very complex for large scenes, multiple primitive types and full 3D surfaces. Further, this approach depends on the knowledge of corresponding viewpoints, which is not possible for all acquisition methods. Therefore, we will – at the expense of requiring a triangulation – avoid the need for building cell complexes.

### 1.3.2.2 Knowledge-based 3D building reconstruction

Reconstructing building models from 3D laser scanning data and imagery gained special attention in the photogrammetric community. Most solutions are based on building-specific object templates, primitive libraries, other knowledge bases or formal grammars.

Elberink (2009) as well as Elberink and Vosselman (2009) detect roofs from airborne laser scans via graph matching against target shapes. Similarly, Huang et al. (2013) reconstruct roofs via fitting primitives from a library and estimating their parameters using a variant of an optimization technique called *Markov Chain Monte Carlo sampling*. Others exploit knowledge about buildings and facades in form of size, position, orientation and topology of different building parts (Pu and Vosselman, 2009a,b) or formulate special rules for grouping adjacent primitives (Tian et al., 2009, 2010). Although being well suitable for some kind of buildings, we will try to avoid to use such specific knowledge, since it is not transferable to other applications like reverse engineering and indoor scenarios as stated in Section 1.1.

Milde et al. (2008) detect planar patches in 3D point clouds, build a region adjacency graph and match subgraphs against templates using formal grammars. In a subsequent work they demonstrate the use of graphical models for modelling buildings using the example of roofs with dormers (Milde and Brenner, 2009). We will adopt the graph representation of building parts, extend it with curved surface types as well as more inter-surface relations and determine the optimal configuration in a graph optimization procedure.

As we will describe in detail in Section 3.2.1, we will introduce model knowledge as well. In contrast to the above-mentioned approaches specifically designed



for building reconstruction, the model proposed in this thesis is designed for general man-made surface structures and will depend on few control parameters only with clear semantics.

### 1.3.2.3 Roofs and large-scale city models

Last but not least we want to focus on large-scale city model reconstructions as they are obtained from airborne laser scans and aerial images. Due to the large area, high altitudes and low resolution the resulting models possess level of detail 2 or less, i.e. buildings are represented as cuboids with basic roof structures, and are often limited to 2.5D structures or even 2D footprints only. Two overviews are given by Brenner et al. (2001) and Brenner (2003).

Various approaches are initially based on image processing. Baillard and Zisserman (2000) compute 3D lines from Canny edges, Werner and Zisserman (2002b) detect principle directions from vanishing points, Rottensteiner (2010) projects watershed regions into a laser point cloud and Kolbe (2001) extracts building features from aerial images. These approaches are hardly transferable to our application, since they require imagery and are limited to piece-wise planar surfaces. In contrast to using radiometric images, Weidner and Förstner (1995) and Brenner (2000) require digital elevation models, limiting the approach to 2.5D reconstructions only.

Peternell and Steiner (2004), Dorninger and Nothegger (2007) and Lari et al. (2011) reconstruct building roofs and facades, respectively, combining region growing and clustering in parameter space. Pu and Vosselman (2006) also segment facade surfaces using region growing, but recognize building parts via numerous, application-specific if-then-else conditions. The main limitation of these region-growing methods is the piece-wise planarity of detected surfaces. We will overcome this problem using a subsequent primitive fitting approach (Chapter 5).

Brunn (2000) reconstructs building models from digital surface models using a graphical model on points, edges and triangles. Due to low sampling density and large noise, the results are poor and limited to simple rectangular building shapes only. Therefore, in contrast to classifying low-level elements like single points or triangles of the surface mesh, we will perform a data-driven segmentation before introducing high-level model knowledge.

Two very promising frameworks for reconstructing large-scale city models are proposed by Zhou and Neumann (2010, 2011, 2012) and Lafarge and Mallet (2012). Like many of the above-mentioned works they are conceptually limited to 2.5D surfaces, thus not suited for our needs.

In contrast to many of the above-mentioned approaches we aim at reconstructing 3D surface models with level of detail 3, i.e. a detailed architectural model. Nevertheless, our method is applicable to large-scale city models, since many well performing building detection algorithms exist (Rottensteiner et al., 2012). Buildings can be detected in low-resolution 2.5D point clouds and subsequently precisely reconstructed using our framework, possibly using 3D point clouds with higher resolution.

With this literature review about surface segmentation and reconstruction methods in mind, we will now introduce the general concept of the proposed reconstruction framework.

## 1.4 General concept

In the following we will describe the three levels of our reconstruction framework, which will be detailed in Chapter 3. Furthermore, we define the expected outcome of the experimental evaluation in Chapter 7.

Considering the large amount of related work in this field, reconstructing surface structures of complex man-made objects proves elusive. While many authors successfully obtain visually pleasing segmentations of small, almost noise-free surface meshes, approaches for real-world object reconstruction usually restrict to rough models with low resolution and often 2.5 dimensions only. We claim, this is due to the large gap between low-level data points and the high-level model knowledge we usually want to exploit. Therefore, we propose a reconstruction framework at three levels:

1. a low-level pre-segmentation into planar regions (Chapter 4),
2. a mid-level segmentation with planar, curved or freeform surface primitives (Chapter 5) and finally
3. a high-level surface classification and parameter estimation using model knowledge about man-made surface structures (Chapter 6).

We will detail this strategy in Section 3.2.2.

The result is an object surface describable as a compound of planar or quadratic surfaces. All surfaces are augmented with certain parameters depending on the underlying surface type, e.g. position and radius for a spherical surface region or orientation and distance to the origin for a plane. Pairs of adjacent surfaces hold a specific relation in between, possibly introducing a constraint on the surfaces' parameters, e.g. orthogonal normal vectors for two planes being orthogonally related to each other.

The reconstructed surface model should be robust to noisy point clouds and outliers. Given a point cloud with reasonably small noise and dense sampling, we expect the reconstruction to yield correct surface types and accurate surface parameters, independent of the respective acquisition method. Furthermore, the computing time should be small enough for practical applicability, i.e. a couple of minutes for surfaces with a few dozens of surface parts. The method should be scalable to large-scale applications like city models.

After giving an overview of the theoretical background in Chapter 2, we will describe the three parts of the proposed reconstruction framework in detail. The experimental evaluation in Chapter 7 should confirm our expectations about the resulting surface model.

## 1.5 Organization of the thesis

The thesis is organized as follows.

In Chapter 2 we briefly collect preliminaries on related topics. This involves the generation and processing of triangular meshes, the computation of shortest paths on manifolds, the minimum description length model selection strategy, inference on graphical models as well as parameter estimation with a Gauss-Helmert model.

We describe both the task and the concept of this thesis in Chapter 3. Besides a more detailed specification of the three reconstruction levels, we introduce the model for man-made surface structures with its surface and relation types.

The following three chapters represent the three reconstruction levels. A fast, data-driven pre-segmentation approach is described in Chapter 4, followed by a hierarchical, model-driven segmentation in Chapters 5. Chapter 6 describes the third level of the proposed reconstruction framework, namely the surface classification via inference on a graphical model along with the surface parameter estimation.

Chapter 7 presents various experiments to demonstrate the performance of the proposed reconstruction framework. It involves experiments on synthetic meshes as well as on real-world data sets.

We close this thesis with concluding remarks and an outlook on future work in Chapter 8.



## CHAPTER 2

---

### Theoretical background

---

This chapter collects the relevant technical concepts used in our approach. It involves reconstructing and processing triangular meshes, computing shortest paths on such surfaces, estimating surface parameters, selecting surface models by minimizing the total description length and inferring an optimal classification of surfaces and inter-surface relations using graphical models.

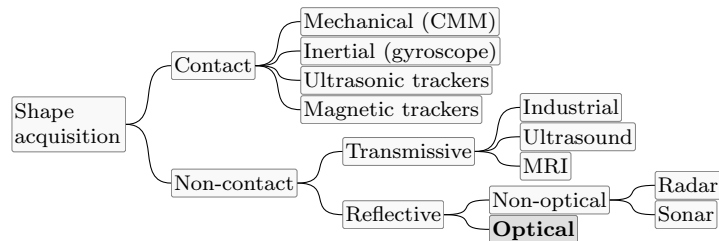
---

<b>2.1</b>	<b>Triangular meshes . . . . .</b>	<b>26</b>
2.1.1	Point cloud triangulation . . . . .	27
2.1.1.1	Triangulation as part of the Delaunay complex. . . . .	27
2.1.1.2	The crust . . . . .	28
2.1.1.3	Surface as zero-crossing of signed distance functions . . . . .	28
2.1.2	Surface normals and curvature . . . . .	29
2.1.2.1	Implicit surfaces. . . . .	29
2.1.2.2	Graph surfaces . . . . .	30
2.1.2.3	Surface meshes . . . . .	31
<b>2.2</b>	<b>Shortest paths on surfaces . . . . .</b>	<b>34</b>
2.2.1	Dijkstra's algorithm. . . . .	34
2.2.2	Fast marching method . . . . .	35
<b>2.3</b>	<b>Parameter estimation in a Gauss-Helmert model . . . . .</b>	<b>36</b>
2.3.1	Maximum-likelihood and least-squares estimation . . . . .	37
2.3.2	Homogeneous parameter vectors . . . . .	37
2.3.3	Homogeneous observation vectors . . . . .	39
2.3.4	Constraints between observations only . . . . .	39
2.3.5	Outlier re-weighting. . . . .	40
<b>2.4</b>	<b>Model selection via description length minimization. . . . .</b>	<b>40</b>
2.4.1	Points without model . . . . .	42
2.4.2	Points on one plane . . . . .	42
2.4.3	Points on multiple planes . . . . .	43
2.4.4	Outliers . . . . .	43

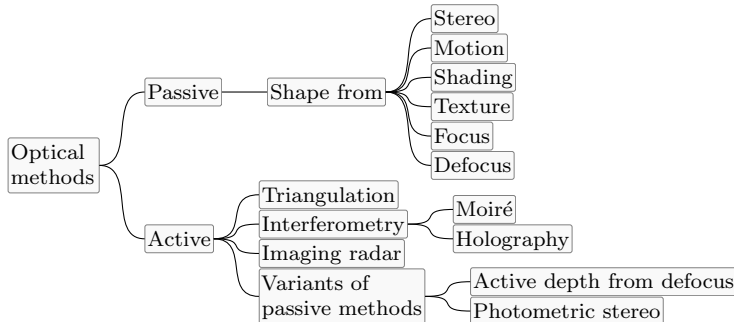
<b>2.5 Inference in graphical models . . . . .</b>	<b>44</b>
2.5.1 Factor graphs . . . . .	44
2.5.2 Marginal probability via the sum-product algorithm . . . . .	46
2.5.3 Most probable variable setting via the max-sum algorithm . . . . .	47

## 2.1 Triangular meshes

Many systems exist for capturing point clouds of 3D shapes (Mada et al., 2003). An overview is given in Fig. 2.1.



(a) Shape acquisition



(b) Optical methods

**Figure 2.1:** 3D shape acquisition techniques. The techniques are divided into contact and non-contact, and the latter into transmissive and reflective (a). Optical methods (b) are part of the last group. Some techniques like magnetic resonance imaging (MRI) yield volumetric data in form of 3D voxel images, but most techniques yield unordered 2.5D or 3D point clouds that represent a surface. Adapted from Mada et al. (2003, Fig. 1).

Since many acquisition techniques yield unordered point clouds, it has been a major research topic to reconstruct the underlying surface, either by connecting points to form triangular or quadrilateral meshes or by interpolating the original point cloud with additional, meshed surface points. In this thesis we will focus on triangular meshes only, since other polygons can be easily converted to triangles.

In the following sections we will collect relevant and most prominent methods for triangulating point clouds, before deriving surface properties like vertex normals and curvature.

### 2.1.1 Point cloud triangulation

Point cloud triangulation, also referred to as surface reconstruction or shape reconstruction, deals with the problem of deriving a meshed surface from unordered 3D points. A meshed surface consists of a set of vertices  $\{\mathcal{V}_v\}$  with indices  $v = 1, \dots, V$  and a set of polygonal faces, i.e. triangular simplices  $\{\mathcal{T}_t\}$  in our case with indices  $t = 1, \dots, T$ , that are represented as a list of three vertex indices each. The vertices together with all simplices form a simplicial complex (Spanier, 1994, pg. 108).

Since object surfaces are usually describable as a manifold dividing the space into inside and outside, we expect the triangular mesh to be free of self-intersections and orientable. Therefore, most triangular meshes encode the orientation of each triangular simplex by ordering the vertex indices clockwise as seen from outside.

Note that in the context of an observed 3D location with coordinates  $\mathbf{X}$  we use the term *point*  $\mathcal{X}$ . For triangular corner points we use the term *vertex*  $\mathcal{V}$ .

We will give a short overview of three major surface triangulation approaches. The first two, the Delaunay complex and the crust, start from a tetrahedralization of the 3D space and label tetrahedrons as inside or outside the object's surface. Since they conceptually connect the original 3D points, their resulting triangulations are called linear *interpolation*. The third concept, signed distance functions, is based on determining an implicit surface function making use of locally estimated vertex normals. The zero-crossing of this function is sampled and triangulated yielding an *approximated* surface with new, previously non-existing vertices. The Delaunay complex mainly serves as basis for the crust, which yields useful results under certain conditions. In this thesis, however, an algorithm of the last group has turned out to triangulate noisy point clouds most reliably.

#### 2.1.1.1 Triangulation as part of the Delaunay complex

The Delaunay complex of a point cloud is created via Delaunay tetrahedralization of the 3D space and consists of a set of tetrahedrons, each with four vertices from the original points. By identifying the tetrahedrons inside the object one obtains the set of boundary faces, i.e. a triangular surface mesh. In the following we differentiate two basic concepts: either classifying or removing tetrahedrons based on local properties, or within a global optimization approach.

Based on a simple rule about the topology of tetrahedrons and an ordering criterion, Boissonnat (1984) iteratively removes tetrahedrons outside the object. The hull of the remaining tetrahedrons forms the triangular surface mesh. Some authors exploit other structures derived from the Delaunay complex, like the Voronoi complex (O'Rourke et al., 1987; Glanvill and Broughan, 1997; Attali, 1997), which is dual to the Delaunay complex, or the Gabriel graph (Attene and Spagnuolo, 2000), which consists of a subset of all Delaunay edges. Edelsbrunner and Mücke (1994) introduce alpha complexes, which are subcomplexes of the Delaunay complex, basically obtained by removing simplices with a circumsphere larger than some predefined threshold. An overview of local methods for surface reconstructions based on the Delaunay complex is given by Edelsbrunner (1998). Unfortunately, all these methods are designed for noise-free samples of piece-wise smooth surfaces. Often they do not yield useful results on real point clouds with

significant noise, since the underlying assumption of the point cloud representing a manifold is violated.

Global methods usually formulate an energy function involving all tetrahedrons. By minimizing this energy a globally optimal labeling is found, separating the Delaunay complex into inside and outside. Labatut et al. (2007) combine three energy terms: visibility, photo-consistency and smoothness. Pan et al. (2009) propose an online surface reconstruction system with a similar approach achieving real-time performance. Jancosek and Pajdla (2011) formulate an energy function for point clouds from multi-view stereo reconstructions based on photo-consistency. These methods inherently require imagery to be available. For arbitrary point clouds, e.g. captured via laser scanning, they are not applicable.

All these methods utilize the Delaunay complex of the original point cloud. The following group of algorithms incorporates additional points: the centers of Voronoi cells.

### 2.1.1.2 The crust

The term *crust* is introduced by Amenta et al. (1998) and continues the concept of surfaces being part of the Delaunay complex. In contrast to the methods discussed in the previous section, the crust is derived from the Delaunay tetrahedralization of the combined point cloud: including the original points and Voronoi points, i.e. the vertices of the Voronoi diagram. While in 2D space the crust is simply the subset of edges connecting original points, the definition is more difficult in 3D space. In the following we will collect only a few approaches.

In order to find triangles being part of the surface Amenta and Bern (1998, 1999) as well as Amenta et al. (1998, 2000) exploit triangles consisting of original points only and their distance to adjacent Voronoi nodes. Extensions exist for noisy point clouds (Amenta et al., 2001; Dey and Goswami, 2006), large data sets (Dey et al., 2001), watertight surfaces (Dey and Goswami, 2003) and surfaces with sharp edges (Dey et al., 2013). The identification of boundary triangles is also formulated as a global optimization, e.g. in form of an eigenvalue problem (Kolluri et al., 2004) or energy minimization (Aganj et al., 2009). A great overview of various crust algorithms is given by Tcherniavski and Stelldinger (2008).

Despite the large variety of algorithms – software is available for all algorithms from Dey et al. – they do not always yield satisfactory results on real-world point clouds. In case of large noise and a significant amount of outliers the resulting meshes contain many non-manifold triangles, i.e. the meshes are not orientable, which complicates the subsequent processing steps. Therefore, we choose an approach of the following group of algorithms based on signed distance functions.

### 2.1.1.3 Surface as zero-crossing of signed distance functions

Signed distance functions are continuously defined over – in our case – the 3D space and indicate the Euclidean distance to the closest point on the surface. They are positive on the one side of the surface and negative on the other side. Therefore, the zero-crossings can be detected and triangulated, e.g. using the well-established marching cubes algorithm (Lorensen and Cline, 1987), which creates polygons in each voxel whose vertices are differently signed. We distinguish two



approaches, one defined over a regular voxel grid and the other one working with the more efficient  $k$ -d tree data structure.

Hoppe et al. (1992) compute oriented vertex normals from  $k$  nearest neighbors, defining gradients of the signed distance function. For each point of a regular 3D voxel grid they compute its signed distance to the nearest tangent plane. The zero-crossing is found using the marching cubes algorithm. Since the signed distance function tends to smooth the original points, it is reasonable to incorporate edge-preserving relaxation terms in order to preserve sharp surface features (Ohtake et al., 2001).

A more recent solution (Kazhdan et al., 2006), called *Poisson surface reconstruction*, exploits octrees as an efficient data structure for 3D point clouds. The name originates from solving the Poisson partial differential equation. Kazhdan and Hoppe (2013) extend this approach by incorporating points as interpolation constraints, yielding a closer approximation of the original points.

The Poisson surface reconstruction yields good results on most data sets. As long as the input point cloud roughly represents a manifold without too large holes, the resulting surface mesh is sufficiently close to the true surface.

Note that this approach requires vertex normals to be present, which is automatically the case for some acquisition techniques. In other cases they can be derived from the original point cloud.

In this section we only covered the most popular and promising methods for triangulating point clouds. A good overview on related approaches is given by Tishchenko (2010, Sect. 2) and by Wilke (2002, Ch. 4) as well as in the survey on Delaunay based surface reconstruction algorithms from Cazals and Giesen (2004).

Assuming to be given a reasonable triangulation of the original point cloud, we want to derive surface properties such as vertex normals and curvature in the following section.

### 2.1.2 Surface normals and curvature

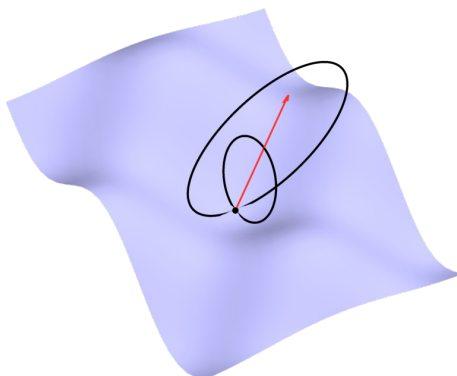
We can use the triangular mesh to compute vertex features like normals and curvature. Fig. 2.2 visualizes an intuitive interpretation of these two surface properties. At each surface point there are two principal directions with minimal and maximal curvature, respectively. The two corresponding curvatures  $\kappa_1 = \frac{1}{r_1}$  and  $\kappa_2 = \frac{1}{r_2}$  are the inverse radii of the two osculating circles. Using  $\kappa_1$  and  $\kappa_2$  we can define three common curvature measures (Pulla et al., 2001; Goldman, 2005), Gaussian curvature  $K$ , mean curvature  $H$  and mean squared curvature  $Q$ :

$$K = \kappa_1 \kappa_2, \quad H = \frac{1}{2} (\kappa_1 + \kappa_2), \quad Q = \frac{1}{2} (\kappa_1^2 + \kappa_2^2). \quad (2.1)$$

We will first define both differential geometric properties on implicit surfaces, transfer the concepts to graph surfaces and finally formulate a model for estimating normals and curvature on discrete triangular meshes.

#### 2.1.2.1 Implicit surfaces

First we assume to be given an implicit surface. In the following we describe how to derive the normal direction  $\mathbf{n}$  and the different curvature measures  $K$ ,  $H$  and  $Q$  at one point of the surface.



**Figure 2.2:** Surface normal and principal curvatures. The red arrow indicates the normal direction of an implicitly defined surface  $z = 0.5y - 0.2 \cos(3x) \cos(4.5y)$  at a point  $\mathbf{X} = [0, 0, -0.2]^T$ . The two principal curvatures at  $\mathbf{X}$  are visualized by two osculating circles.

An implicit surface is defined as the set of points  $\mathbf{X} = [X, Y, Z]^T$  fulfilling the equation  $f(X, Y, Z) = 0$ . Using a parametrization  $(u, v)$  a point on the surface is

$$\mathbf{X}(u, v) = \begin{bmatrix} X(u, v) \\ Y(u, v) \\ Z(u, v) \end{bmatrix}. \quad (2.2)$$

The normal is the normalized crossproduct of the gradients in  $u$ - and  $v$ -direction:

$$\mathbf{n} \propto \mathbf{X}_u \times \mathbf{X}_v. \quad (2.3)$$

Using the first and second fundamental forms (Strubecker, 1969; Pulla et al., 2001; Goldman, 2005)

$$F_1 = \begin{bmatrix} \mathbf{X}_u^T \mathbf{X}_u & \mathbf{X}_u^T \mathbf{X}_v \\ \mathbf{X}_u^T \mathbf{X}_v & \mathbf{X}_v^T \mathbf{X}_v \end{bmatrix} \quad \text{and} \quad F_2 = \begin{bmatrix} \mathbf{X}_{uu}^T \mathbf{n} & \mathbf{X}_{uv}^T \mathbf{n} \\ \mathbf{X}_{uv}^T \mathbf{n} & \mathbf{X}_{vv}^T \mathbf{n} \end{bmatrix} \quad (2.4)$$

we can compute the three curvatures

$$K = \frac{|F_2|}{|F_1|}, \quad H = \frac{1}{2} \text{tr}(F_1^{-1} F_2), \quad Q = \frac{1}{2} \text{tr}(F_1^{-1} F_2 F_1^{-1} F_2). \quad (2.5)$$

We will transfer this approach to the special type of graph surfaces in the following section.

### 2.1.2.2 Graph surfaces

A graph surface explicitly defines the height  $z = z(x, y)$  as a function of  $x$  and  $y$ . Therefore, the parametrization is trivial and the approach for computing vertex normals  $\mathbf{n}$  and curvatures from the previous section is easily applicable.

A point on the surface is parametrized as

$$\mathbf{X}(x, y) = \begin{bmatrix} x \\ y \\ z(x, y) \end{bmatrix}. \quad (2.6)$$

In analogy to (2.3) the surface normal is

$$\mathbf{n} = \frac{1}{\sqrt{1 + z_x^2 + z_y^2}} \begin{bmatrix} -z_x \\ -z_y \\ 1 \end{bmatrix}. \quad (2.7)$$

With the first and second fundamental forms (2.4) transferred to graph surfaces  $z = z(x, y)$ :

$$F_1 = \begin{bmatrix} 1 + z_x^2 & z_x z_y \\ z_x z_y & 1 + z_y^2 \end{bmatrix} \quad \text{and} \quad F_2 = \frac{1}{\sqrt{1 + z_x^2 + z_y^2}} \begin{bmatrix} z_{xx} & z_{xy} \\ z_{xy} & z_{yy} \end{bmatrix} \quad (2.8)$$

and the curvature measures defined in (2.5), we obtain

$$K = \frac{z_{xx}z_{yy} - z_{xy}^2}{(1 + z_x^2 + z_y^2)^2}, \quad (2.9)$$

$$H = \frac{(1 + z_x^2)z_{xx} - 2z_xz_yz_{xy} + (1 + z_y^2)z_{yy}}{2(1 + z_x^2 + z_y^2)^{3/2}}, \quad (2.10)$$

$$Q = \frac{((1 + z_y^2)z_{xx} - z_xz_yz_{xy})^2 + ((1 + z_x^2)z_{yy} - z_xz_yz_{xy})^2}{2(1 + z_x^2 + z_y^2)^2} + \frac{((1 + z_x^2)z_{xy} - z_xz_yz_{xx})((1 + z_y^2)z_{xy} - z_xz_yz_{yy})}{(1 + z_x^2 + z_y^2)^2}. \quad (2.11)$$

Later we will work in a local tangent coordinate system with  $z_x = z_y = 0$ . Then the curvature formulas drastically simplify to

$$K = z_{xx}z_{yy} - z_{xy}^2, \quad H = \frac{1}{2}(z_{xx} + z_{yy}), \quad Q = \frac{1}{2}(z_{xx}^2 + 2z_{xy}^2 + z_{yy}^2). \quad (2.12)$$

So far we derived surface features for parametrized, continuous surfaces. In the following section we will deal with discrete, densely sampled surface meshes.

### 2.1.2.3 Surface meshes

Now we assume to be given a set of points  $\{\mathcal{X}_v\}$  that sample an unknown surface. After transforming the local neighborhood of a point  $\mathcal{X}_v$  such that the mesh represents a graph surface, we fit a quadratic surface into the neighboring points and derive surface features from the resulting parametrization.

In order to compute the vertex normal  $\mathbf{n}_v$  and curvatures  $K_v$ ,  $H_v$  or  $Q_v$  at a point  $\mathcal{X}_v$ , we first transform the point  $\mathcal{X}_v$  with its neighbors  $\mathcal{X}_u = \text{ne}(\mathcal{X}_v)$  into a local tangent coordinate system. The transformation is defined as a rotation yielding the transformed normal vector pointing upwards, thus the gradients vanish:  $z_x \approx z_y \approx 0$ . We estimate the parameters  $\mathbf{a}$  of a quadratic graph surface in the rotated coordinate system and finally derive expressions for both normal and curvature.

The neighborhood  $\text{ne}(\mathcal{X}_v)$  can be defined as a sphere of fixed radius, the  $k$  nearest vertices or all vertices connected to  $\mathcal{X}_v$  by a triangular edge. The advantage of using  $k$  nearest vertices is that with  $k$  we can define the minimum

number of points required for estimating parameters of the graph surface and avoid an underdetermination.

The tangent coordinate system is defined as centered at point  $\mathcal{X}_v$  and parallel aligned with an approximate normal direction  $\mathbf{n}_{v,0}$ . This direction can, e.g., be derived from neighboring triangles: Since the vertex indices of orientable triangular meshes can be assumed to be ordered, the cross product of the two triangular edge vectors

$$\mathbf{n}_t = (\mathbf{X}_{\mathcal{T}_{t,1}} - \mathbf{X}_{\mathcal{T}_{t,2}}) \times (\mathbf{X}_{\mathcal{T}_{t,1}} - \mathbf{X}_{\mathcal{T}_{t,3}}) \quad (2.13)$$

is pointing outwards for all triangles  $t$ , each with the three vertices  $\mathcal{T}_{t,1}$ ,  $\mathcal{T}_{t,2}$  and  $\mathcal{T}_{t,3}$ . By averaging over all neighboring triangle normals

$$\mathbf{n}_{v,0} \propto \sum_{t \in \text{ne}(v)} \mathbf{n}_t \quad (2.14)$$

and normalization such that  $|\mathbf{n}_{v,0}| = 1$  we obtain consistently oriented approximations of vertex normals. In this context  $t \in \text{ne}(v)$  are the indices of triangles  $\mathcal{T}_t$  adjacent to the point  $\mathcal{X}_v$ . Note that by using the non-normalized cross product  $\mathbf{n}_t$ , whose magnitude is proportional to the area of the triangle  $t$ , we obtain a weighted average  $\mathbf{n}_{v,0}$ .

We fit a quadratic graph surface

$$z = a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 \quad (2.15)$$

through the transformed points  $u \in \text{ne}(v)$

$$\mathbf{X}_u = \begin{bmatrix} X_u \\ Y_u \\ Z_u \end{bmatrix}. \quad (2.16)$$

We estimate the parameters  $\mathbf{a} = [a_{00}, a_{10}, a_{01}, a_{20}, a_{11}, a_{02}]^\top$  solving the equation system

$$Z_u = \underbrace{\begin{bmatrix} 1 & X_u & Y_u & X_u^2 & X_u Y_u & Y_u^2 \end{bmatrix}}_{\mathbf{x}_u^\top} \underbrace{\begin{bmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{20} \\ a_{11} \\ a_{02} \end{bmatrix}}_{\mathbf{a}}. \quad (2.17)$$

With the combined coordinate matrix  $\mathbf{X}_v = [\mathbf{x}_u^\top]$  and all  $Z$ -coordinates  $\mathbf{Z}_v = [Z_u]$  the estimated parameters  $\hat{\mathbf{a}}$  are

$$\hat{\mathbf{a}} = (\mathbf{X}_v \mathbf{X}_v^\top)^{-1} \mathbf{X}_v^\top \mathbf{Z}_v. \quad (2.18)$$

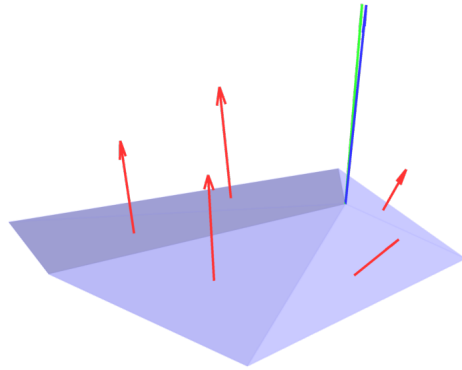
If the local coordinate system is aligned almost parallel to the surface normal, the parameters  $\hat{a}_{10} \approx \hat{a}_{01} \approx 0$  vanish and we can use the simplified curvature formulas (2.12). With the derivatives of the quadratic surface (2.15) plugged into the equations for the normal (2.7) and curvatures (2.12) of graph surfaces we obtain the normal at point  $v$

$$\mathbf{n}_v \propto \begin{bmatrix} -a_{10} - 2a_{20}X_v - a_{11}Y_v \\ -a_{01} - 2a_{02}Y_v - a_{11}X_v \\ 1 \end{bmatrix} \quad (2.19)$$

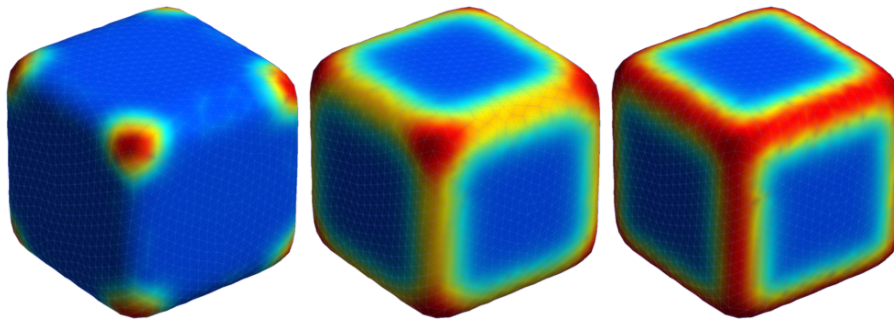
normalized to  $|\mathbf{n}_v| = 1$  and the curvatures

$$K = 4a_{20}a_{02} - a_{11}^2, \quad H = a_{20} + a_{02}, \quad Q = 2a_{20}^2 + a_{11}^2 + 2a_{02}^2. \quad (2.20)$$

Note that the normal  $\mathbf{n}_v$  needs to be rotated back into the original coordinate system considering the approximate tangent direction  $\mathbf{n}_{v,0}$ . Fig. 2.3 shows triangle normals, the derived approximation  $\mathbf{n}_{v,0}$  as well as the estimated normal  $\mathbf{n}_v$  for a vertex with five neighboring triangles. The three curvature measures are illustrated in Fig. 2.4 for a small example data set.



**Figure 2.3:** Vertex and triangle normals on a small example mesh. The vertex normal derived from the parameters of a graph surface (green) is close to the approximation (blue) from surrounding triangle normals (red).



(a) Gaussian curvature  $K$     (b) Mean curvature  $H$     (c) Mean sq. curvature  $Q$

**Figure 2.4:** Vertex curvature of a small example mesh implicitly defined as the 8-norm sphere  $x^8 + y^8 + z^8 = 1$  with  $V = 2000$  vertices. We compute the curvature from 50 nearest neighbors leading to a broad impact of object corners and edges. The individual curvature measures mainly differ in how they distinguish curvature in one principal direction at edges and curvature in two directions at corners.

Because similar vertices are more likely to belong to a common surface region, we will make use of these geometric properties in Chapter 4 to compute similarities between adjacent vertices and to control the surface pre-segmentation. In fact, we will integrate dissimilarities along paths on surfaces and derive a segmentation based on shortest paths, i.e. paths with few dissimilarity. In the

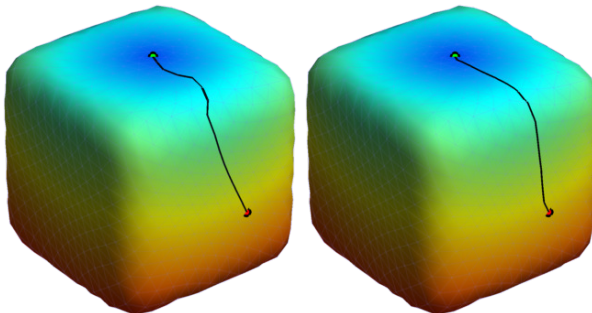
following section we will briefly discuss algorithms for determining such shortest paths.

## 2.2 Shortest paths on surfaces

As indicated in Section 1.3.1.1, we pre-segment the given surface mesh based on distance transforms with distances being defined as the length of the shortest path. Vertices with small distances in between will more likely belong to the same region, while large distances are indicative of separate regions.

A distance transform is a function that assigns distance measures to all given vertices taking into account a predefined distance metric as well as one or multiple seed vertices, from where the distances are to be computed. While in Euclidean spaces one usually is interested in Euclidean distances, distance transforms on surfaces often yield intrinsically defined, geodesic or curvature-adaptive distances.

In discrete geometry there are two standard algorithms for deriving an intrinsic *distance map*: Dijkstra’s algorithm (Dijkstra, 1959) and the fast marching method (FMM, Sethian, 1996) (Fig. 2.5). Both algorithms run in linearithmic time  $O(V \log V)$  on sparse graphs like triangular meshes. Approximations for FMM with linear complexity (Xu et al., 2010) and even parallel implementations (Weber et al., 2008) exist. In the following we will briefly describe both Dijkstra’s algorithm and FMM.



(a) Dijkstra’s algorithm    (b) Fast marching method

**Figure 2.5:** Comparison of two distance transforms. The resulting distance map  $D$  is shown as colored vertices from near (blue) to far (red). The bold black line indicates the shortest path between two vertices indicated with green and red dots. While Dijkstra’s algorithm is restricted to triangular edges, FMM can pass through triangles as well and thus yields slightly shorter distances.

### 2.2.1 Dijkstra’s algorithm

Given an attributed graph and one or more seed points, Dijkstra’s algorithm computes the shortest path along *graph edges* from each vertex to its closest seed point yielding a distance map  $D$  (Fig. 2.5a). Implicitly it yields path lengths, i.e. intrinsic distances.

The implementation is based on a marching front  $Q$ , i.e. a set of vertices that are currently in process. In every iteration one front vertex  $u := \operatorname{argmin}_{u \in Q} D_u$  with shortest distance  $D_u$  is removed  $Q := Q \setminus u$  and all its neighbors  $v \in \operatorname{ne}(u)$

updated to the new distance  $D_v := \min(D_v, D_u + d_u^v)$  with the local distance  $d_u^v$  from vertex  $u$  to vertex  $v$  and added to the front  $\mathcal{Q} := \mathcal{Q} \cup v$ . The algorithm is given in Alg. 2.1.

---

**Algorithm 2.1:** Dijkstra’s algorithm. Given pair-wise distances  $d_u^v$  between vertices  $\mathcal{V}_u$  and their neighbors  $\mathcal{V}_v \in \text{ne}(\mathcal{V}_u)$  as well as a vertex list of the current front  $\mathcal{Q}$ , the algorithm propagates the front, updating the distance map  $\mathbf{D}$  and vertex labels  $\mathbf{l}$ , until there are no more vertices to process.

---

**Input:** neighbors  $\text{ne}(u)$ , pair-wise distances  $d_u^v$ , front  $\mathcal{Q}$   
**Updates:** distance map  $\mathbf{D}$ , vertex labels  $\mathbf{l}$   
**while** front is not empty  $\mathcal{Q} \neq \{\}$  **do**  
    select closest vertex in front  $u \leftarrow \text{argmin}_{u \in \mathcal{Q}} D_u$   
    remove  $u$  from front  $\mathcal{Q} \leftarrow \mathcal{Q} \setminus u$   
    **foreach** neighbor  $v \in \text{ne}(u)$  **do**  
        evaluate new distance  $D'_v \leftarrow D_u + d_u^v$   
        **if** new distance  $D'_v < D_v$  **then**  
            update distance  $D_v \leftarrow D'_v$  and vertex label  $l_v \leftarrow l_u$   
            add this neighbor  $v$  to front  $\mathcal{Q} \leftarrow \mathcal{Q} \cup v$   
        **end**  
    **end**  
**end**

---

### 2.2.2 Fast marching method

The fast marching method (FMM), especially its formulation for meshed manifolds (Kimmel and Sethian, 1998), computes *surface intrinsic distances* on meshed surfaces (Fig. 2.5b). In contrast to Dijkstra’s algorithm, paths can pass *through* triangles, thus are not restricted to triangular edges.

Technically, both Dijkstra’s algorithm and FMM solve the so-called Eikonal equation

$$|\nabla D_v| = F_v \quad (2.21)$$

for all vertices  $\mathcal{V}_v$ ,  $v = 1, \dots, V$ , with the boundary condition  $D_{v_0} = 0$  at a seed vertex  $v_0$ . In terms of distances on meshed manifolds FMM’s interpretation is as follows: Given a seed vertex  $\mathcal{V}_{v_0}$  and values  $\mathbf{F} = [F_v]$  defining the friction at each vertex  $\mathcal{V}_v$  on the manifold, FMM yields the distance map  $\mathbf{D} = [D_v]$  such that the gradient magnitude  $|\nabla D_v|$  is identical to the local friction  $F_v$ . Then the distance  $F_v$  is proportional to the arrival time of a propagating wave front starting at vertex  $\mathcal{V}_{v_0}$ , in case the local friction is inversely proportional to the local speed of the wave front. In case of a Euclidean metric the friction  $F_v$  is constant for all vertices  $\mathcal{V}_v$  and the distances  $D_v$  represent the geodesic distances, i.e. the length of the shortest path to the seed vertex  $\mathcal{V}_{v_0}$ .

The implementation of FMM is similar to Dijkstra’s algorithm. The update formula, however, is slightly more complicated: While in Dijkstra’s algorithm distances simply are the sum of edge lengths, with FMM the front can travel *through* triangles.

Both algorithms meet all requirements for the pre-segmentation proposed in Chapter 4. Due to its simplicity and flexibility (Schindler and Förstner, 2013)

we choose Dijkstra's algorithm to compute distance transforms throughout this thesis.

## 2.3 Parameter estimation in a Gauss-Helmert model

In Chapter 6 we will need to determine maximum-likelihood estimations of surface parameters  $\boldsymbol{\theta}$  given a set of  $V$  points  $\{\mathcal{X}_v\}$ ,  $v = 1, \dots, V$  and the surface class  $s \in \{1, \dots, S\}$ , i.e. the primitive type we want to fit into the point cloud. We can formulate this task in terms of a Gauss-Helmert model (Förstner and Wrobel, 2004, pg. 81ff.), also known as mixed model (Koch, 1997, pg. 231f.). In the current section we will formulate both the maximum-likelihood and the least-squares estimation in a Gauss-Helmert model, discuss the special case of homogeneous parameter or observation vectors, derive the estimation model with constraints between observations only and explain a robust re-weighting scheme to handle outliers.

We will use the notation of adjustment computations as follows: We are given  $N = 3V$  observed coordinates  $\mathbf{y}$  of the  $V$  points  $\{\mathcal{X}_v\}$ , which are assumed to be normally distributed around the true but unknown observation vector  $\tilde{\mathbf{y}}$ :

$$\mathbf{y} \sim \mathcal{N}(\tilde{\mathbf{y}}, \boldsymbol{\Sigma}_{yy}). \quad (2.22)$$

Further, there are  $G$  constraints  $\mathbf{g} = \mathbf{g}(\mathbf{y}, \mathbf{p}) \stackrel{!}{=} \mathbf{0}$  between the observations  $\mathbf{y}$  and the  $U$  unknown parameters  $\mathbf{p}$ , the latter being the unknown surface parameters  $\boldsymbol{\theta}$  depending on the surface class  $s$ . Usually each point leads to a constraint, i.e. its distance to the surface is zero; thus there are  $G = V$  constraints  $\mathbf{g}$ . The system is underdetermined if there are less constraints than unknown parameters  $G < U$ . On the other hand it is overdetermined with redundancy  $R = G - U$  if there are more points than parameters.

The estimated entities  $\hat{\mathbf{y}} = \mathbf{y} + \hat{\mathbf{v}}$  and  $\hat{\mathbf{p}}$  need to fulfill the constraints

$$\mathbf{g}(\mathbf{y} + \hat{\mathbf{v}}, \hat{\mathbf{p}}) = \mathbf{0} \quad (2.23)$$

with the residuals  $\hat{\mathbf{v}}$ . This equation is linearized at approximate values  $\mathbf{y}_0$  and  $\mathbf{p}_0$ :

$$\mathbf{g}(\mathbf{y}_0, \mathbf{p}_0) + A\Delta\hat{\mathbf{p}} + B^T\Delta\hat{\mathbf{y}} = \mathbf{0} \quad (2.24)$$

with the derivatives

$$A_{G \times U} = \left. \frac{\partial \mathbf{g}(\mathbf{y}, \mathbf{p})}{\partial \mathbf{p}} \right|_{\mathbf{y}=\mathbf{y}_0, \mathbf{p}=\mathbf{p}_0}, \quad (2.25)$$

$$B^T_{G \times N} = \left. \frac{\partial \mathbf{g}(\mathbf{y}, \mathbf{p})}{\partial \mathbf{y}} \right|_{\mathbf{y}=\mathbf{y}_0, \mathbf{p}=\mathbf{p}_0} \quad (2.26)$$

and the differences between approximation and estimation

$$\Delta\hat{\mathbf{p}} = \hat{\mathbf{p}} - \mathbf{p}_0, \quad (2.27)$$

$$\Delta\hat{\mathbf{y}} = \hat{\mathbf{y}} - \mathbf{y}_0. \quad (2.28)$$

In the following we will present the maximum-likelihood and the least-squares solution for this linearized model. Afterwards we discuss derivative approaches for homogeneous entities, missing parameters and observations with outliers.



### 2.3.1 Maximum-likelihood and least-squares estimation

There are different concepts for parameter estimation. We will briefly present the objective function and the solution for both the maximum-likelihood and the least-squares estimation within the Gauss-Helmert model. A more detailed derivation can be found in the literature, e.g. by Förstner and Wrobel (2004, pg. 81ff.) or by Koch (1997, pg. 231f.).

The objective of the maximum-likelihood estimation is to find an estimation  $\hat{\mathbf{v}}$  and  $\hat{\mathbf{p}}$  that fulfills the constraints (2.24) and maximizes the likelihood of the observations  $\mathbf{y}$  given the estimated parameters  $\hat{\mathbf{p}}$ :

$$\begin{aligned} & \text{maximize} && p(\mathbf{y} \mid \mathbf{p}, \sigma_0^2) \\ & \text{subject to} && \mathbf{g}(\mathbf{y}, \mathbf{p}) = \mathbf{0}. \end{aligned} \quad (2.29)$$

In case of normally distributed observations  $\mathbf{y} \sim \mathcal{N}(\tilde{\mathbf{y}}, \Sigma_{yy})$  the maximum-likelihood estimation is equivalent to the least-squares estimation

$$\begin{aligned} & \text{minimize} && \hat{\mathbf{v}}^\top \Sigma_{yy}^{-1} \hat{\mathbf{v}} \\ & \text{subject to} && \mathbf{g}(\mathbf{y}, \mathbf{p}) = \mathbf{0}, \end{aligned} \quad (2.30)$$

which minimizes the weighted sum of squared residuals  $\hat{\mathbf{v}}$  under the constraints  $\mathbf{g}$ . After setting the derivative of the objective function to zero and solving the equation system, both approaches lead to the following iterative solution for the parameters  $\hat{\mathbf{p}}$ , the observations  $\hat{\mathbf{y}}$ , the variance coefficient  $\hat{\sigma}_0^2$  and the covariance matrix of the estimated parameters  $\Sigma_{\hat{p}\hat{p}}$  (Förstner and Wrobel, 2004, pg. 84ff.):

$$\hat{\mathbf{p}} = \mathbf{p}_0 + (A^\top \Sigma_{gg}^{-1} A)^{-1} A^\top \Sigma_{gg}^{-1} \mathbf{c}_g, \quad (2.31)$$

$$\hat{\mathbf{y}} = \mathbf{y} + \Sigma_{yy} B \Sigma_{gg}^{-1} (\mathbf{c}_g - A \Delta \hat{\mathbf{p}}), \quad (2.32)$$

$$\hat{\sigma}_0^2 = \frac{(\hat{\mathbf{y}} - \mathbf{y})^\top \Sigma_{yy}^{-1} (\hat{\mathbf{y}} - \mathbf{y})}{R}, \quad (2.33)$$

$$\Sigma_{\hat{p}\hat{p}} = (A^\top \Sigma_{gg}^{-1} A)^{-1} \quad (2.34)$$

with the redundancy  $R = G - U$ . The residuals of the constraints  $\mathbf{c}_g$  as well as their covariance matrix  $\Sigma_{gg}$  are

$$\mathbf{c}_g = -\mathbf{g}(\mathbf{y}_0, \mathbf{p}_0) + B^\top (\mathbf{y}_0 - \mathbf{y}), \quad (2.35)$$

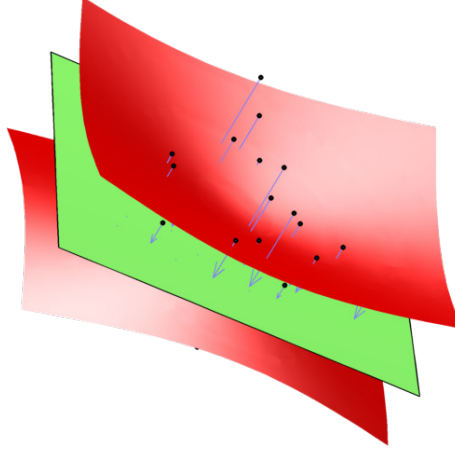
$$\Sigma_{gg} = B^\top \Sigma_{yy} B. \quad (2.36)$$

Fig. 2.6 shows the estimated plane through a set of  $V = 50$  points together with the residuals  $\hat{\mathbf{v}}$  and the  $3\sigma$ -confidence interval derived from  $\Sigma_{\hat{p}\hat{p}}$ .

This is the basic model for all parameter estimation problems within this thesis. In the following sections we will adapt this solution to work with homogeneous entities, constraints between observations only and outliers.

### 2.3.2 Homogeneous parameter vectors

Homogeneous entities deserve special attention, like a 3D plane  $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_n \\ -\theta_d \end{bmatrix}$  with normal  $\boldsymbol{\theta}_n$  and distance to the origin  $\theta_d$ . Their variable scale leads to a singular covariance matrix and thus a singular equation system. In contrast to addressing



**Figure 2.6:** Estimated plane through a set of 3D points. The  $V = 50$  points (black) are synthetically generated and perturbed with  $\sigma = 25\%$  Gaussian noise w.r.t. maximum coordinate range. The parameters of a plane (green) have been estimated within the Gauss-Helmert model. The residuals  $\hat{\mathbf{v}}$  are shown with blue arrows and point to the closest location  $\hat{\mathbf{y}}$  on the plane, i.e. fulfilling the constraint  $\mathbf{g}(\hat{\mathbf{y}}, \hat{\mathbf{p}})$  and minimizing  $\hat{\mathbf{v}}^\top \Sigma_{yy} \hat{\mathbf{v}}$ . The  $3\sigma$ -confidence interval is shown as red, bivalve surface.

this problem with additional constraints (Meidow et al., 2009), we exploit a strategy from Förstner (2012) which projects the estimation problem into the lower-dimensional tangent space, where the covariance matrix is regular.

In case of a homogeneous parameter vector  $\mathbf{p}$  its scale is variable. Therefore, the covariance matrix  $\Sigma_{\hat{\mathbf{p}}\hat{\mathbf{p}}}$  is singular and cannot be inverted as required in (2.34). Since the scale  $|\mathbf{p}|$  does not matter, we project the estimation problem into the vector's tangent space  $\mathbf{p}_r = J_p \mathbf{p}$  with the Jacobian  $J_p$  as listed in Tab. 3.1 for each primitive class and only solve for the update  $\Delta \hat{\mathbf{p}}_r$  in this reduced space:

$$\mathbf{g}(\mathbf{y}_0, \mathbf{p}_0) + A_r \Delta \hat{\mathbf{p}}_r + B^\top \Delta \hat{\mathbf{y}} = \mathbf{0} \quad (2.37)$$

with the derivative

$$A_r = \frac{\partial \mathbf{g}(\mathbf{y}, \mathbf{p})}{\partial \mathbf{p}_r} = A J_p^\top. \quad (2.38)$$

The solution is obtained by estimating the parameter update  $\Delta \hat{\mathbf{p}}_r$  in the reduced tangent space and transforming back to the original space  $\hat{\mathbf{p}}$  when needed:

$$\Delta \hat{\mathbf{p}}_r = (A_r^\top \Sigma_{gg}^{-1} A_r)^{-1} A_r^\top \Sigma_{gg}^{-1} \mathbf{c}_g, \quad (2.39)$$

$$\hat{\mathbf{p}} = \mathbf{N}(\mathbf{p}_0) + J_p^\top \Delta \hat{\mathbf{p}}_r, \quad (2.40)$$

$$\hat{\mathbf{y}} = \mathbf{y} + \Sigma_{yy} B \Sigma_{gg}^{-1} (\mathbf{c}_g - A_r \Delta \hat{\mathbf{p}}_r), \quad (2.41)$$

$$\Sigma_{\hat{\mathbf{p}}\hat{\mathbf{p}}} = J_p^\top (A_r^\top \Sigma_{gg}^{-1} A_r)^{-1} J_p \quad (2.42)$$

and the estimated variance coefficient  $\hat{\sigma}_0^2$  as in (2.33). Note that we need to normalize the parameter vector  $\hat{\mathbf{p}}$  such that it correctly corresponds to its variance  $\Sigma_{\hat{\mathbf{p}}\hat{\mathbf{p}}}$ .

This way we can estimate homogeneous parameter vectors in a Gauss-Helmert model. In the following section we will apply the very same principle to homogeneous observation vectors, which we will deal with in Section 6.5.

### 2.3.3 Homogeneous observation vectors

In case of a homogeneous observation vector  $\mathbf{y}$  normalized to unit length we proceed similarly. We project the vector  $\mathbf{y}$  and its covariance matrix  $\Sigma_{yy}$  into its tangent space  $\mathbf{y}_r = J_y \mathbf{y}$  and  $\Sigma_{y_r y_r} = J_y \Sigma_{yy} J_y^T$  with  $J_y$  depending on the primitive type as listed in Tab. 3.1 and solve for the residuals  $\hat{\mathbf{v}}_r = \Delta \hat{\mathbf{y}}_r - (\mathbf{y}_r - \mathbf{y}_{r,0})$  in this reduced space:

$$\mathbf{g}(\mathbf{y}_0, \mathbf{p}_0) + A \Delta \hat{\mathbf{p}} + B_r^T \Delta \hat{\mathbf{y}}_r = \mathbf{0} \quad (2.43)$$

with the derivative

$$B_r^T = \frac{\partial \mathbf{g}(\mathbf{y}, \mathbf{p})}{\partial \mathbf{y}_r} = B^T J_y^T. \quad (2.44)$$

The solution is

$$\hat{\mathbf{v}}_r = \Sigma_{y_r y_r} B_r \Sigma_{gg}^{-1} (\mathbf{c}_g - A \Delta \hat{\mathbf{p}}), \quad (2.45)$$

$$\hat{\mathbf{y}} = \mathbf{y} + J_y^T \hat{\mathbf{v}}_r, \quad (2.46)$$

$$\hat{\sigma}_0^2 = \frac{(\hat{\mathbf{y}}_r - \mathbf{y}_r)^T \Sigma_{y_r y_r}^{-1} (\hat{\mathbf{y}}_r - \mathbf{y}_r)}{R} \quad (2.47)$$

with the residuals of the constraints  $\mathbf{c}_g$  and their covariance matrix  $\Sigma_{gg}$ :

$$\mathbf{c}_g = -\mathbf{g}(\mathbf{y}_{r,0}, \mathbf{p}_0) + B_r^T (\mathbf{y}_{r,0} - \mathbf{y}_r), \quad (2.48)$$

$$\Sigma_{gg} = B_r^T \Sigma_{y_r y_r} B_r. \quad (2.49)$$

Again, we need to normalize the updated observation vector  $\hat{\mathbf{y}}$ .

This yields an optimal estimation in case of a homogeneous observation vector  $\mathbf{y}$ . Similarly, we can treat multiple concatenated homogeneous observation vectors  $\mathbf{y} = [\mathbf{y}_i]$  by projecting and normalizing each part individually. Although not needed within this thesis, homogeneous parameter vectors  $\mathbf{p}$  and observations  $\mathbf{y}$  can be used both in one estimation.

### 2.3.4 Constraints between observations only

A special case of the Gauss-Helmert model is to have no unknown parameters at all and constraints between observations  $\mathbf{y}$  only. The objective is to find observations  $\hat{\mathbf{y}}$  with as small corrections  $\hat{\mathbf{v}}$  as possible that satisfy the constraints.

The corresponding functional model is

$$\mathbf{g}(\mathbf{y}) = \mathbf{0} \quad (2.50)$$

with the linearization at approximate observations  $\mathbf{y}_0$ :

$$\mathbf{g}(\mathbf{y}_0) + B^T \Delta \hat{\mathbf{y}} = \mathbf{0}. \quad (2.51)$$

Both the maximum-likelihood and the least-squares estimation lead to the solution

$$\hat{\mathbf{y}} = \mathbf{y} + \Sigma_{yy} B \Sigma_{gg}^{-1} \mathbf{c}_g \quad (2.52)$$

with  $\mathbf{c}_g = -\mathbf{g}(\mathbf{y}_0) + B^T (\mathbf{y}_0 - \mathbf{y})$  similarly to (2.35),  $\Sigma_{gg}$  as in (2.36) and the estimated variance coefficient  $\hat{\sigma}_0^2$  as in (2.33).

We will apply this model in Section 6.5 for jointly estimating surface parameter vectors considering constraints between pairs of adjacent surfaces.

### 2.3.5 Outlier re-weighting

So far we assumed the observations to be normally distributed. Often they are, however, perturbed by a significant amount of outliers, which would disturb the estimation if not treated appropriately. We will use an iterative re-weighting scheme to downweight points with large residuals, i.e. alleged outliers.

In order to reduce the impact of outliers among the observations  $\mathbf{y}$  we do not minimize  $\hat{\mathbf{v}}^\top \Sigma_{yy}^{-1} \hat{\mathbf{v}} = \sum_v \hat{\mathbf{v}}_v^\top \Sigma_{y_v y_v}^{-1} \hat{\mathbf{v}}_v$  as in (2.31) for uncorrelated points  $\mathcal{X}_v$ , but

$$\text{minimize } \sum_v \rho \left( \sqrt{\frac{X_v}{E(X_v)}} \right) \quad (2.53)$$

with a re-weighting function  $\rho$  (Förstner and Wrobel, 2004, pg. 108ff.) applied to the chi-squared distributed sum of squared residuals  $X_v$  per point  $\mathcal{X}_v$

$$X_v = \hat{\mathbf{v}}_v^\top \Sigma_{y_v y_v}^{-1} \hat{\mathbf{v}}_v \sim \chi_3^2 \quad (2.54)$$

with three degrees of freedom. Thus, the expectation  $E(X_v) = 3$  is the number of dimensions in 3D space. This is equivalent to

$$\text{minimize } \hat{\mathbf{v}}^\top W \hat{\mathbf{v}} \quad (2.55)$$

with a block diagonal weight matrix  $W$  built from weight matrices  $W_v$  for each point  $\mathcal{X}_v$

$$W_v = w_v \Sigma_{y_v y_v}^{-1}. \quad (2.56)$$

The covariance matrices  $\Sigma_{y_v y_v}$  of each point  $\mathcal{X}_v$  will be re-weighted for the next iteration using the weight function

$$w_v = w(X_v) = \frac{\rho' \left( \sqrt{X_v/3} \right)}{\sqrt{X_v/3}} \quad (2.57)$$

depending on the sum of squared residuals  $X_v$  of all coordinates of a point  $\mathcal{X}_v$ .

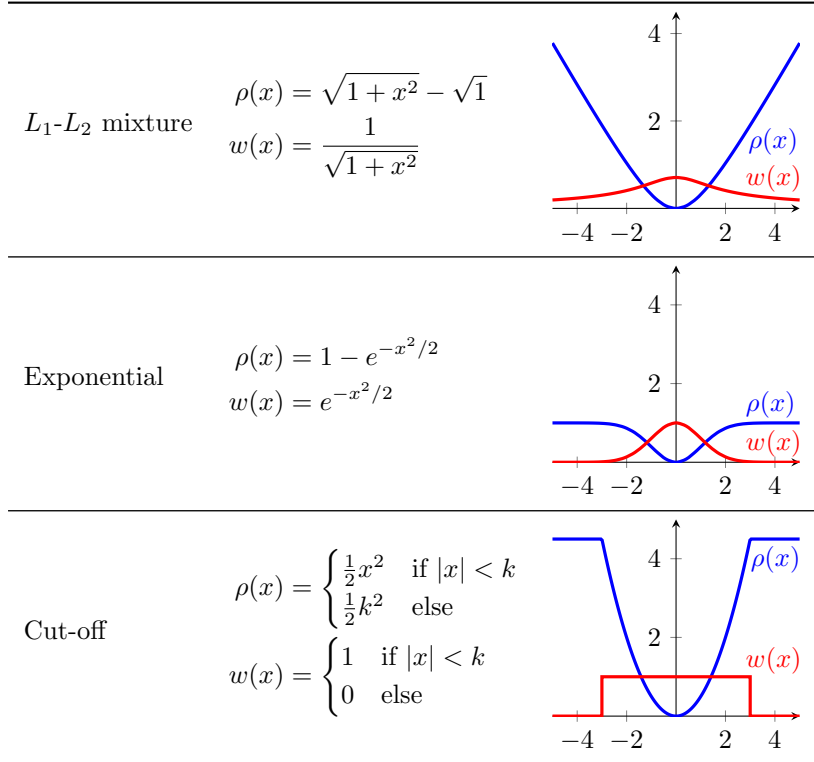
During the iteration sequence we exchange the weight function such that outliers retain some weight during the first iterations but are completely ignored in the final iterations (Tab. 2.1): The first iterations use the  $L_1$ - $L_2$  mixture, followed by a few iterations with the exponential re-weighting function. Finally, the cut-off function ignores outliers completely.

With the Gauss-Helmert model and its derivatives we have a powerful tool for the estimation problems to be solved within this thesis. Besides its applicability for both Euclidean and homogeneous entities and its robustness to noise, it yields empirical variance and covariance information, which we will use to evaluate different models for classification purposes in Section 6.2. Another computationally less expensive model selection strategy is based on minimizing a description length, which is addressed in the following section.

## 2.4 Model selection via description length minimization

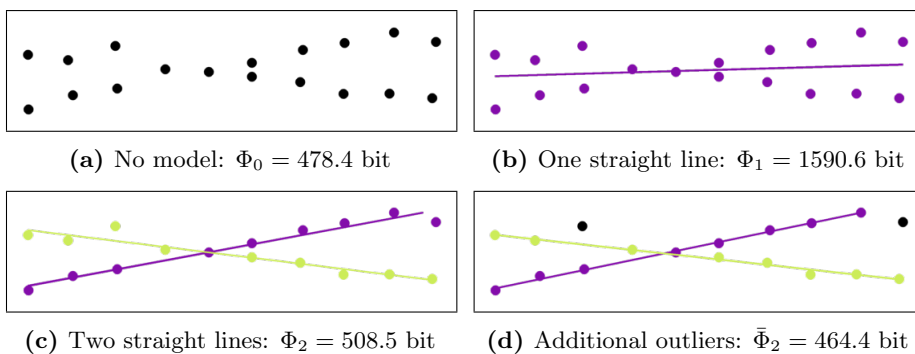
When modelling different primitive types, a model selection problem occurs. To describe the given data, one needs to find a trade-off between approximation

**Table 2.1:** Three outlier re-weighting schemes with the re-weighting function  $\rho(x)$  and the weight function  $w(x)$ . While the  $L_1$ - $L_2$  mixture still yields moderate weights  $w$  for vertices with a large sum of squared residuals, the exponential re-weighting scheme downweights them almost completely and the cut-off scheme distinguishes between constant weight for inliers and zero weight for outliers. The cut-off point  $k$  is usually set to  $k = 3$  as shown in the figure, which corresponds to the 99.7 % quantile of the standard normal distribution. The three re-weighting functions  $\rho$  are constructed to have equal curvature at  $x = 0$ .



error and model complexity: While an arbitrarily complex model would perfectly approximate each data point, its complexity would be unnecessarily high. Both aspects are covered by the concept of minimizing the description length (MDL), which aims at finding an as short as possible description for the given data without losing information. We will use this concept for low- and mid-level decisions in Chapters 4 and 5.

Following the MDL principle described by Georgeff and Wallace (1984), Förstner (1989) applies MDL for detecting straight lines in noisy 2D point clouds with outliers, as illustrated in Fig. 2.7. As shown by Leonardis et al. (1995) and Yang and Förstner (2010), MDL can be applied to primitives in 2.5D and 3D space as well. In the following we will derive formulas for computing the description length for various scenarios in 3D space.



**Figure 2.7:** Model selection on a 2D data set with 18 points spread across a coordinate range of  $r = 1$ . The desired resolution is  $\varepsilon = 10^{-4}$  and a coordinate uncertainty of  $\sigma = 0.005 = 0.5\%$  was used to generate synthetic, Gaussian noise. The residuals w.r.t. one straight line are much larger than the noise  $\sigma$ , thus the description length  $\Phi_1$  (b) is larger than without any line  $\Phi_0$  (a). Two straight lines are still worse than none,  $\Phi_2 > \Phi_0$  (c), since two outliers lead to bad line parameters and consequently to large residuals. Only modelling those outliers explicitly leads to the minimum description length  $\bar{\Phi}_2$  (d). Note that the improvement of  $\bar{\Phi}_2$  compared to  $\Phi_0$  or  $\Phi_2$  will be much more significant for a larger data sets, since the overhead of a complex model will be compensated by many points whose residuals will decrease.

### 2.4.1 Points without model

The most trivial scenario is to have a set of 3D points without any underlying model. Thus, the description only contains the coordinates of all points.

Following Förstner (1989), the description length for one uniformly distributed coordinate is  $\text{lb} \frac{r}{\varepsilon}$  with the coordinate range  $r$  and the resolution  $\varepsilon$ , up to which we want to encode the points' coordinates. Thus, for  $V$  points in 3D space without any model assumption we obtain a description length

$$\Phi_0 = 3V \text{lb} \frac{r}{\varepsilon}. \quad (2.58)$$

In the next sections we will extend this description. First we assume the points to approximately lie on one or multiple planes. Finally, we allow some points to be outliers, i.e. not corresponding to any plane.

### 2.4.2 Points on one plane

In case all  $V$  points lie on a plane, the description length decreases. After the plane parameters have once been encoded, for each point only its in-plane location and its off-plane residual need to be encoded, which is usually less information than all three coordinates separately.

When encoding the 2D in-plane locations for all points, the 1D off-plane residuals and the three independent plane parameters individually, we obtain the description length

$$\Phi_1 = \underbrace{2V \text{lb} \frac{r}{\varepsilon}}_{\text{in-plane locations}} + \underbrace{\sum_{v=1}^V \left( \frac{1}{2 \ln 2} \cdot \frac{v_v^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right)}_{\text{off-plane residuals}} + \underbrace{3 \text{lb} \frac{r}{\varepsilon}}_{\text{plane parameters}}. \quad (2.59)$$

If the model assumption is correct, the residuals  $v_v$  are normally distributed with a small, given uncertainty  $\sigma$ . Then the overall description length with a planar model  $\Phi_1$  is usually smaller than without any model  $\Phi_0$ , although we need to additionally store plane parameters with same range  $r$  and resolution  $\varepsilon$  like for the point coordinates.

So far we introduced a model assumption to reduce the description length. In the following section we add alternative models, i.e. additional planes, yielding an only slightly modified description.

### 2.4.3 Points on multiple planes

When considering multiple planes, the description length only slightly changes. We need to encode point labels, i.e. which plane each point corresponds to. If the model assumption of multiple planes is correct, the additional description induced by point labels is compensated by smaller residuals, since  $L > 1$  planes approximate the points better than a single plane.

Now we need to store three independent parameters for each of the  $L$  planes, point labels with a description length of  $\text{lb } L$  per point and residuals  $v_v$ . This leads to an overall description length of

$$\Phi_L = \underbrace{2V \text{lb} \frac{r}{\varepsilon}}_{\text{in-plane locations}} + \underbrace{\sum_{v=1}^V \left( \frac{1}{2 \ln 2} \cdot \frac{v_v^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right)}_{\text{off-plane residuals}} + \underbrace{3L \text{lb} \frac{r}{\varepsilon}}_{\text{plane parameters}} + \underbrace{V \text{lb} L}_{\text{labeling}}. \quad (2.60)$$

We will apply this description length formulation in Chapter 4 to find an optimal number of planes representing the original surface mesh. In Chapter 5 we will introduce additional surface types, one of which will be a freeform surface without any surface parameters. Thus, all corresponding points are encoded as outliers, which we will address in the following section.

### 2.4.4 Outliers

Finally, we might want to encode  $\bar{V}$  outliers, i.e. points not belonging to any of the  $L$  planes. This is relevant if only a few points are far off the plane and thus are better encoded independently. But we will apply this encoding to whole surface parts as well that can not be described with one of the surface types available, i.e. so-called freeform surfaces.

We still need to encode parameters for  $L$  planes, but the point labeling has to consider  $L + 1$  states, e.g. with  $L = 0$  for outliers. Further, we encode the  $\bar{V}$  outliers as uniform coordinates like in (2.58), thus sum the residuals  $v_v$  only over inliers  $v$  with  $l_v > 0$  and encode residuals for outliers uniformly:

$$\bar{\Phi}_L = \underbrace{2V \text{lb} \frac{r}{\varepsilon}}_{\text{in-plane locations}} + \underbrace{\bar{V} \text{lb} \frac{r}{\varepsilon}}_{\text{outlier residuals}} + \underbrace{\sum_{\{v|l_v>0\}} \left( \frac{1}{2 \ln 2} \cdot \frac{v_v^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right)}_{\text{off-plane residuals}} + \underbrace{3L \text{lb} \frac{r}{\varepsilon}}_{\text{plane parameters}} + \underbrace{V \text{lb}(L+1)}_{\text{labeling}}. \quad (2.61)$$

This formulation comprises all scenarios for describing a set of 3D points with multiple planes. In this section we restricted ourselves to planar surfaces only, although the concept is easily transferable to curved surfaces, as we will see in Section 5.4.

The MDL strategy proves to be a suitable model selection approach: By minimizing the description length for encoding a given point cloud, we find the model that best represents the given points. The selected model not only contains an optimal number of surfaces, since each additional surface increases the description length; but it also yields optimal surface parameters, since only the least sum of squared residuals leads to the minimal description length.

We will use the MDL principle twice throughout our reconstruction framework: It will serve as optimization criterion for the planar pre-segmentation in Chapter 4 and will guide the refinement operations in Chapter 5 with curved and freeform surface parts.

## 2.5 Inference in graphical models

In Chapter 6 we will classify mutually dependent surfaces and inter-surface relations. Those discrete variables together with their dependencies can be represented using a graphical model. This section gives a short introduction to inference with graphical models. After introducing the factor graph representation, we will present two common inference algorithms.

### 2.5.1 Factor graphs

Factor graphs are a special representation for a set of stochastic variables and their mutual dependencies. This section briefly describes the graph structure and a solution for two common inference problems via so-called message passing.

A graphical model contains nodes, which represent stochastic variables  $x_i$ , and edges, representing conditional dependencies between adjacent nodes. The joint probability density  $p(\mathbf{x})$  for all variables  $\mathbf{x} = [x_i]$

$$p(\mathbf{x}) = \frac{1}{Z} \prod_j f_j(\mathbf{x}_j) \quad (2.62)$$

is the product of multiple factors  $f_j = f_j(\mathbf{x}_j) > 0$  depending on variable subsets  $\mathbf{x}_j$  with a normalization  $Z$  such that  $\int p(\mathbf{x}) = 1$ , i.e.  $p(\mathbf{x})$  is a probability density. Depending on the cardinality  $|\mathbf{x}_j|$  the factors  $f_j$  are called unary, binary or – as in our case in Chapter 6 – ternary factors. The factors can be explicitly modeled as factor nodes in a so-called *factor graph* (Kschischang et al., 2001), which is bipartite: All edges connect one variable node with one factor node, like in Fig. 2.8b.

There are two common problems related to graphical models: (1) to find the marginal probability density  $p(x_i)$  of a single variable  $x_i$  or (2) to find a variable setting  $\hat{\mathbf{x}} = \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x})$  that maximizes the joint probability density  $p(\mathbf{x})$ .

The factors  $f_j$  can be chosen to be conditional probability densities  $f_j = p(\mathbf{x}_j | \mathcal{D})$  depending on data  $\mathcal{D}$ , which allows them to be learned from training data. Alternatively, they can be interpreted as potentials  $\phi$ , using the notion



from the concept of Markov random fields (Kindermann and Snell, 1980). Then the function  $f_j(\mathbf{x}_j)$  can be chosen such that likely configurations of the variable subset  $\mathbf{x}_j$  yield a large value  $f_j$ ; otherwise there are no restrictions on the form of the function  $f_j$ . Although we will formulate factors on the basis of probability densities, we will neglect the normalization, thus obtain potentials. Since we will be interested in the variable setting  $\hat{\mathbf{x}}$  maximizing the joint probability density  $p(\mathbf{x})$ , the normalization is not needed.

Exact solutions to the above-mentioned problems are computationally prohibitive for general graphs (Bishop, 2006, Ch. 8.4). There are, however, algorithms with polynomial complexity for chains and trees that make use of a principle called *message passing* (Pearl, 1982): Each node receives messages from its neighbors and computes updated messages that are sent back to the neighbors in turn. Following Bishop (2006), we will show for both problems, finding the marginal probability density as well as finding the most probable variable setting, that with the right choice of messages an optimal solution can be guaranteed for chains and trees after traversing the graph twice.

Effective approximations are possible for general graphs as discussed in (Bishop, 2006, p. 417): One class of approximation schemes is based on stochastic sampling and – given infinite computing time – yields exact results. In practice they are tractable for small-scale problems only. Another class are deterministic approximation schemes based on analytically approximating the posterior density function, e.g. assuming Gaussian probability distributions. Finally, a simple and widely used approximation is known as *loopy belief propagation* (Frey and MacKay, 1998). One iteratively applies the message passing algorithm as it is formulated for loop-less graphs and observes convergence in many but not all cases. In our implementation in Chapter 6 we will work with this approach.

For the sake of clarity we will derive both algorithms, the *sum-product* and the *max-sum* algorithm, for Markov chains and only mention the corresponding messages for the general case. Fig. 2.8a shows the structure of a Markov chain with directed edges: Linearly connected variable nodes  $x_i$  depend only on its predecessors  $\text{ne}(x_i) = \{x_{i-1}\}$ . The joint probability is

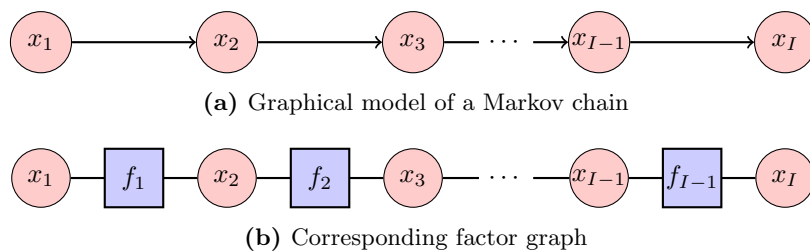
$$p(\mathbf{x}) = p(x_1) \cdot p(x_2 | x_1) \cdot p(x_3 | x_2) \cdots p(x_I | x_{I-1}). \quad (2.63)$$

The corresponding factor graph in Fig. 2.8b indicates the factors of the joint probability density

$$p(\mathbf{x}) = f_1(x_1, x_2) f_2(x_2, x_3) \cdots f_{I-1}(x_{I-1}, x_I), \quad (2.64)$$

where the unary factor  $p(x_1)$  is subsumed in the binary factor  $f(x_1, x_2)$ . In the following we will neglect the indices of the factors  $f$ , since they are uniquely identifiable by their list of arguments.

We will use a factor graph in Section 6.1 to represent the man-made surface structure with variables for both the surface types and inter-surface relations. In general this will be a cyclic graph, i.e. containing loops, thus requiring inference via loopy belief propagation or a similar strategy. In the following two sections we will start describing the sum-product and the max-sum algorithm, respectively, for Markov chains and provide formulas for the general case afterwards.



**Figure 2.8:** Graphical model and factor graph of a Markov chain. The graphical model (a) consists of  $I$  variable nodes  $x$  with directed edges between each neighboring pair of them. In the factor graph (b) these edges are replaced by factor nodes  $f$  representing the mutual dependencies.

## 2.5.2 Marginal probability via the sum-product algorithm

The sum-product algorithm is a common method for inferring marginal probabilities in a graphical model. We will first derive it for Markov chains, before generalizing the formulas to general graphs.

**Markov chains.** The marginal probability density of a variable  $x_i$  is obtained via summation of  $p(\mathbf{x})$  over all other variables  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_I$ :

$$p(x_i) = \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_I} p(\mathbf{x}) \quad (2.65)$$

$$\propto \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_I} f(x_1, x_2) \cdots f(x_{I-1}, x_I). \quad (2.66)$$

This can be reordered as

$$\begin{aligned}
 p(x_i) &\propto \underbrace{\sum_{x_{i-1}} \underbrace{f(x_{i-1}, x_i) \sum_{x_{i-2}} \underbrace{f(x_{i-2}, x_{i-1}) \cdots \sum_{x_2} \underbrace{f(x_2, x_3) \sum_{x_1} \underbrace{f(x_1, x_2)}_{\mu_2(x_2)}}_{\mu_3(x_3)}}_{\mu_{i-1}(x_{i-1})}}_{\mu_i(x_i)}}_{\pi_{i+1}(x_{i+1})} \cdot \\
 &\quad \underbrace{\sum_{x_{i+1}} \underbrace{f(x_i, x_{i+1}) \sum_{x_{i+2}} \underbrace{f(x_{i+1}, x_{i+2}) \cdots \sum_{x_{I-1}} \underbrace{f(x_{I-2}, x_{I-1}) \sum_{x_I} \underbrace{f(x_{I-1}, x_I)}_{\pi_{I-1}(x_{I-1})}}_{\pi_{I-2}(x_{I-2})}}_{\pi_{i+1}(x_{i+1})}}_{\pi_i(x_i)}}_{\pi_i(x_i)} \\
 &= \mu_i(x_i) \pi_i(x_i) \quad (2.67) \\
 &= \mu_i(x_i) \pi_i(x_i) \quad (2.68)
 \end{aligned}$$

with recursively computed forward and backward messages

$$\mu_i(x_i) = \sum_{x_{i-1}} f(x_{i-1}, x_i) \mu_{i-1}(x_{i-1}), \quad (2.69)$$

$$\pi_i(x_i) = \sum_{x_{i+1}} f(x_i, x_{i+1}) \pi_{i+1}(x_{i+1}). \quad (2.70)$$

**General graphs.** In general graphs, where variable nodes  $x_i$  may have more than two neighboring factor nodes  $f_j \in \text{ne}(x_i)$ , according to Bishop (2006, p. 404ff.) the marginal probability density  $p(x_i)$  is

$$p(x_i) \propto \prod_{f_j \in \text{ne}(x_i)} \mu_{f_j \rightarrow x_i}(x_i), \quad (2.71)$$

$$\mu_{f_j \rightarrow x_i}(x_i) = \sum_{\text{ne}(f_j) \setminus x_i} f_j(\text{ne}(f_j)) \prod_{x'_i \in \text{ne}(f_j)} \mu_{x'_i \rightarrow f_j}(x'_i), \quad (2.72)$$

$$\mu_{x_i \rightarrow f_j}(x_i) = \prod_{f'_j \in \text{ne}(x_i) \setminus f_j} \mu_{f'_j \rightarrow x_i}(x_i). \quad (2.73)$$

It depends on messages  $\mu$  that are initialized at leaf nodes with

$$\mu_{x_i \rightarrow f_j}(x_i) = 1, \quad (2.74)$$

$$\mu_{f_j \rightarrow x_i}(x_i) = f(x_i) \quad (2.75)$$

and repeatedly passed to neighboring nodes according to a predefined schedule until convergence.

The resulting marginal probability density  $p(x_i)$  describes the probability distribution of the variable  $x_i$  with all other variables not being fixed. In Chapter 6 we will, however, be interested in the most probable variable *setting*  $\mathbf{x}$ , i.e. maximizing the joint probability  $p(\mathbf{x})$  by fixing the values of all variables. This inference problem is addressed by the max-sum algorithm as described by the following section.

### 2.5.3 Most probable variable setting via the max-sum algorithm

In contrast to the previous problem, where we were interested in the local probability density  $p(x_i)$  of a variable  $x_i$ , not knowing the values of surrounding variables, we are now trying to find the most probable variable setting  $\mathbf{x}$ , i.e. globally optimizing the joint probability density  $p(\mathbf{x})$ . This problem is solved by the max-sum algorithm, which is very similar to the previously described sum-product algorithm. Instead of summing over all variable states, it chooses the maximum. Furthermore, it usually works in the log-space, where products become sums, thus motivating the name max-sum algorithm.

**Markov chains.** The setting  $\mathbf{x}$  that yields the maximum probability is found by maximizing the joint probability density  $p(\mathbf{x})$  over all variables  $x_i$  with indices  $i = 1, \dots, I$ :

$$\max_{\mathbf{x}} p(\mathbf{x}) \propto \max_{x_1} \cdots \max_{x_{I-1}} \max_{x_I} \left( f(x_1, x_2) \cdots f(x_{I-2}, x_{I-1}) f(x_{I-1}, x_I) \right) \quad (2.76)$$

$$\propto \max_{x_1} \left( f(x_1, x_2) \cdots \max_{x_{I-1}} \left( f(x_{I-2}, x_{I-1}) \max_{x_I} f(x_{I-1}, x_I) \right) \right). \quad (2.77)$$

In order to avoid numerical problems it is convenient to maximize the logarithm of the probability and introduce messages  $\pi$  that are recursively evaluated:

$$\max_{\mathbf{x}} \log p(\mathbf{x}) \propto \max_{x_1} \left( \underbrace{\log f(x_1, x_2) + \cdots + \max_{x_{I-1}} \left( \underbrace{\log f(x_{I-2}, x_{I-1}) + \max_{x_I} \log f(x_{I-1}, x_I)}_{\pi_{I-1}(x_{I-1})} \right)}_{\pi_{I-2}(x_{I-2})} \right)_{\pi_1(x_1)}. \quad (2.78)$$

Note that here the messages  $\pi$  are differently defined as for the sum-product algorithm presented in the previous section.

**General graphs.** In general graphs the variable nodes  $x_i$  may have more than two neighboring factor nodes  $f_j \in \text{ne}(x_i)$ . According to Bishop (2006, p. 413) the joint probability of the most probable setting  $\mathbf{x}$  is obtained via recursively summing over all variables  $\mathbf{x}$  starting at any root node  $x_i$ :

$$\max_{\mathbf{x}} \log p(\mathbf{x}) = \max_{x_i} \sum_{f_j \in \text{ne}(x_i)} \mu_{f_j \rightarrow x_i}(x_i), \quad (2.79)$$

$$\mu_{f_j \rightarrow x_i}(x_i) = \max_{\text{ne}(f_j) \setminus x_i} \log f_j(\text{ne}(f_j)) + \sum_{x'_i \in \text{ne}(f_j)} \mu_{x'_i \rightarrow f_j}(x'_i), \quad (2.80)$$

$$\mu_{x_i \rightarrow f_j}(x_i) = \sum_{f'_j \in \text{ne}(x_i) \setminus f_j} \mu_{f'_j \rightarrow x_i}(x_i). \quad (2.81)$$

The messages  $\mu$  are initialized at leaf nodes with

$$\mu_{x_i \rightarrow f_j}(x_i) = 0, \quad (2.82)$$

$$\mu_{f_j \rightarrow x_i}(x_i) = \log f(x_i) \quad (2.83)$$

and repeatedly passed to neighboring nodes until convergence.

So far we obtain the highest possible joint probability  $\max_{\mathbf{x}} \log p(\mathbf{x})$ . The most probable setting  $\mathbf{x}$  itself is obtained by storing pointers back to the values of  $x'_i$  that achieved the maximum and backtracing these values from the root node  $x_i$  to all other nodes  $\mathbf{x}$ .

Although neither sum-product nor max-sum guarantee exact results on general graphs, they usually yield satisfactory results. Within this thesis we use the Probabilistic Graphical Model Library from Andres et al. (2008). We will apply the max-sum algorithm with loopy belief propagation in Chapter 6 to find the most probable setting of surface types and inter-surface relations for reconstructing man-made surface structures.

All these preliminaries from the fields of mesh processing, parameter estimation, model selection and graphical models form the theoretical background we need for our surface structure reconstruction framework. In the following chapter we will specify the task of this thesis in more detail and precisely formulate the concept of the proposed reconstruction approach.



## CHAPTER 3

---

### Concept for reconstructing models of man-made surfaces

---

This chapter specifies the task to be solved in this thesis in more detail. We characterize the given data and necessary assumptions, formulate the objective of this work and define criteria for the later evaluation. Afterwards we detail the concept of the proposed reconstruction framework. This includes the two parts of the model for man-made surface structures, namely surface classes and inter-surface relations, as well as the three levels, at which the reconstruction operates.

---

<b>3.1 Task specification</b>	<b>51</b>
3.1.1 Type of input data	52
3.1.2 Assumptions about the surface triangulation	52
3.1.3 Objective of the proposed reconstruction framework	54
3.1.4 Criteria for the experimental evaluation	55
<b>3.2 Concept</b>	<b>56</b>
3.2.1 Model for man-made surface structures	56
3.2.1.1 Surface classes	57
3.2.1.2 Inter-surface relations	58
3.2.2 Reconstruction framework at three levels	60

---

### 3.1 Task specification

In Section 1.2 we already formulated the principle goal of this work: Given a triangulation of a densely sampled surface, we want to automatically recognize and parametrize the underlying surface structure. The following sections will detail certain aspects of this task.

### 3.1.1 Type of input data

This section specifies the given data more precisely: Besides a triangulated point cloud we only require some uncertainty information about the points' coordinates.

We expect to be given a dense point cloud of a man-made object captured using common 3D acquisition techniques and triangulated using one of the methods mentioned in Section 2.1.1. The data consists of a list of  $V$  3D points  $\{\mathcal{X}_v\}$  with indices  $v = 1, \dots, V$  and a set of  $T$  triangles  $\{\mathcal{T}_t\}$  with indices  $t = 1, \dots, T$ . Each triangle  $\mathcal{T}_t$  contains three indices referring to the point list. Each point  $\mathcal{X}_v$  contains a coordinate vector  $\mathbf{X}_v$  and information about its uncertainty, e.g. a covariance matrix  $\Sigma_{\mathbf{X}_v \mathbf{X}_v}$  or only a variance  $\sigma^2$ .

Many acquisition methods yield additional point attributes, e.g. reflectivity, color or waveforms profiles. Although our approach could conceivably use such information within the surface segmentation in Chapter 4, we will not rely on any of them and only expect triangulated points with variance.

For successfully reconstructing a surface model, a triangulated point cloud with known uncertainty does not suffice. We need to make certain assumptions which we will discuss in the following section.

### 3.1.2 Assumptions about the surface triangulation

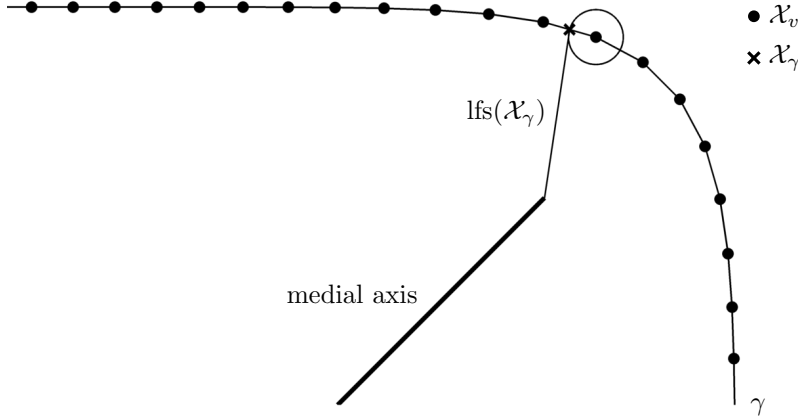
For the proposed reconstruction framework to function properly we need to make a few, mostly weak assumptions on the given data.

**Uncorrelated, isotropic, normally distributed uncertainty.** The data points are usually disturbed by random noise due to the acquisition process. Although this noise depends on the acquisition system and configuration, we assume it to be unbiased and normally distributed as well as small w.r.t. the object size. While the proposed reconstruction framework could be formulated for arbitrary covariance matrices  $\Sigma_{\mathbf{X}\mathbf{X}}$ , we assume mutually independent noise for all coordinates of a point; i.e. the covariance matrix of a point  $\Sigma_{\mathbf{X}_v \mathbf{X}_v} = \sigma^2 I_3$  is diagonal with common variance  $\sigma^2$  for each coordinate. Furthermore, we assume uniform mutually independent points, i.e. zero correlations  $\Sigma_{\mathbf{X}_u \mathbf{X}_v}$  between points  $u \neq v$ , which simplifies most processing steps. These assumptions also facilitate the practical application, since precise noise models with full covariance matrices and correlation between the points are often not available or intractable in terms of memory and computing time.

**Sampling density.** We assume the surface to be densely sampled. A common measure for characterizing the sampling density is the  $\epsilon$ -sampling, which is originally defined for a set of points  $\{\mathcal{X}_v\}$  on a 2D curve: It depends on the *local feature size* of a point  $\mathcal{X}$  on a curve  $\gamma$ . Given the *medial axis* as the set of points with more than one closest point on the curve  $\gamma$ , the local feature size  $\text{lfs}(\mathcal{X})$  is the Euclidean distance of  $\mathcal{X}$  to the medial axis. Then a set of points  $\{\mathcal{X}_v\}$  is an  $\epsilon$ -sample of the curve  $\gamma$  if every point  $\mathcal{X}_\gamma \in \gamma$  on the curve  $\gamma$  is within distance  $\epsilon \cdot \text{lfs}(\mathcal{X}_v)$  of some point  $\mathcal{X}_v$  (Edelsbrunner, 1998). This concept is visualized in Fig. 3.1. We will refer to local feature size and  $\epsilon$ -sampling in Section 7.2.2.4 to characterize the sensitivity of a segmentation procedure w.r.t. sampling density, as



larger values of  $\epsilon$  correspond to lower sampling density. According to Tcherniavski and Stellingner (2008), 3D surface triangulation algorithms require an  $\epsilon$ -sampling of  $\epsilon < 0.5$ , most of them even  $\epsilon < 0.2$  or less.

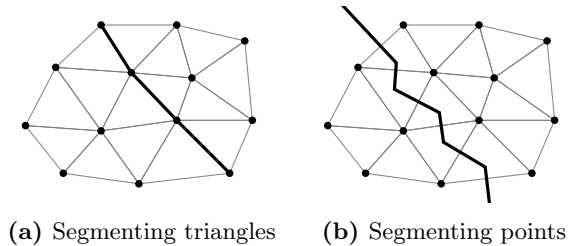


**Figure 3.1:** Local feature size and  $\epsilon$ -sampling. A 1D curve  $\gamma$  is sampled with a set of points  $\{\mathcal{X}_v\}$  shown as black dots. The bold straight line represents the medial axis of the curve, i.e. all points of the 2D plane with two or more closest points in  $\gamma$ . The local feature size  $\text{lfs}(\mathcal{X}_\gamma)$  is the distance of the point  $\mathcal{X}_\gamma$  to the medial axis. The point set  $\{\mathcal{X}_v\}$  is an  $\epsilon$ -sample of the curve  $\gamma$  if every point  $\mathcal{X}_\gamma \in \gamma$  is within radius  $\epsilon \cdot \text{lfs}(\mathcal{X}_v)$  of some sample point  $\mathcal{X}_v$ . In this example  $\epsilon$  is around 0.25.

**Piece-wise quadratic surface.** Since we focus on man-made surface structures, we expect the triangulated point cloud to be intuitively dividable into planar and quadratic parts, i.e. quadrics of special kinds, as these groups of surfaces are most common in architecture and computer aided design. In some cases we will not be able to exclude freeform surfaces completely. Therefore, our reconstruction framework will be able to handle them as well.

**Semantics of a data point.** In the field of surface segmentation there exist two major concepts: Surface parts either consist of a set of triangles, i.e. the region boundaries will be comprised of triangular edges, or a set of points, i.e. the region boundaries will pass *through* triangles and will be comprised of Voronoi edges. We assume the acquisition process to capture points on locally planar surface patches, thus associate a point with a surface region and not with an edge. Consequently, we will segment points, not triangles. Fig. 3.2 illustrates both concepts.

**Manifold surface.** The surface does not need to be closed as often required by meshing algorithms. The proposed reconstruction framework will be able to handle open surfaces like every 2.5D mesh or meshes with holes. We do, however, require the surface to be a manifold, i.e. locally homeomorphic to either a disk or a half-disk. Although usually the case, the surface does not have to be orientable: Conceptually, a surface homeomorphic to a Möbius strip or a Klein bottle can be reconstructed with the proposed framework. For technical reasons and due to the limited practical relevance of such



**Figure 3.2:** Two possible concepts for segmenting a meshed 2D manifold. One possibility is to assign triangles to either one region or the other yielding a segmentation boundary (bold line) following edges of the original triangulation (a). Alternatively, one assigns vertices to its corresponding region yielding a segmentation boundary following edges of the Voronoi diagram (b). Throughout this thesis we will follow the second concept: We segment points.

non-orientable manifolds, we will focus on orientable surfaces within this thesis.

**Data structure with efficient queries.** Some parts of the algorithm will require fast access to neighboring vertices  $\mathcal{V}_u \mid u \in \text{ne}(v)$  given one vertex  $\mathcal{V}_v$ . Therefore, it is advisable to precompute those neighborhood relations or to use data structures that support such queries in constant or logarithmic time. Neighboring vertices connected via triangular edges can be obtained in linear time  $O(V)$  for the whole mesh with  $V$  vertices by traversing all edges of the triangulation. Other possible neighborhoods are  $k$ -nearest neighbors or vertices in a certain radius that can be efficiently queried in logarithmic complexity  $O(\log V)$  per vertex using  $k$ -d trees or octrees, both of which are created in linearithmic time  $O(V \log V)$ .

**Knowledge of the vertical axis.** One determining factor, especially for constructing buildings, is gravity. Therefore, most architectural surfaces contain many elements that are parallel or perpendicular w.r.t. the vertical direction. We will assume this direction to be known or the given mesh to be aligned such that the Z-axis is parallel to the vertical direction. Although this assumption stabilizes the reconstruction process, we do not necessarily *require* the vertical direction to be known. In case it is not accessible, the proposed framework still works – only with less specific surface classes.

After having clarified both structure and properties of the given data, we will detail the objective of the proposed reconstruction framework.

### 3.1.3 Objective of the proposed reconstruction framework

The objective of this work is to automatically derive a surface structure from a triangulated point cloud. The surface structure contains both a segmentation into multiple non-overlapping regions as well as a surface parametrization for each of those regions. In the following we will detail the objective for both reconstruction parts and align ourselves w.r.t. the large number of specially tailored methods for reconstructing buildings and building parts.

The objective of this work is to develop an automatic segmentation and reconstruction approach for triangulated point clouds of man-made surface structures. The segmentation involves labels  $\mathbf{l} = [l_v]$  for each point  $\mathcal{X}_v$ , since we decided to segment points, not triangles, in the previous section. Each label is an integer  $l_v \in \{1, \dots, L\}$ , thus the segmentation is complete with  $L$  non-overlapping regions. Each region should contain exactly one connected component of the triangulation, whose vertices can be approximated with a geometrically simple primitive or – in exceptional cases – described as freeform surface. We will not group regions which are not neighbored, e.g. when analyzing a point cloud with multiple road parallel building facades.

For each surface region  $l$  we want to determine the most likely primitive type  $\hat{s}_l \in \{1, \dots, S\}$  and find parameter estimates  $\hat{\theta}_l$  that minimize the sum of squared residuals between corresponding points  $\{\mathcal{X}_v \mid l_v = l\}$  and the surface defined by the region’s primitive. Furthermore, we want to guide the parameter estimation with constraints  $\mathbf{g}$  induced by inter-surface relations  $\hat{r}_k \in \{1, \dots, R\}$  that are most likely given the two involved surface regions  $\theta_l$  and  $\theta_{l'}$ . We will define the set of  $S$  possible surface classes and  $R$  inter-surface relations shortly in Section 3.2.1.

For reconstructing buildings, many highly specialized methods exist for preprocessing the data or for solving very application-specific tasks: Small parts of a building like windows (e.g. Pu and Vosselman, 2007; Tuttas and Stilla, 2013), doors (e.g. Kang et al., 2010) or even window cornice (e.g. Brandenburger et al., 2013) are often detected with machine learning approaches in point clouds and imagery. Many recent approaches for facade reconstruction exploit images only (e.g. Werner and Zisserman, 2002a; Müller et al., 2007; Gool et al., 2007; Ripperda, 2008) or the combination with terrestrial laser scans (Brenner and Ripperda, 2006; Becker, 2009). These approaches explicitly model repetitive structures of windows and doors using formal grammars. This way they are able to detect structures that are not recognizable in a pure 3D mesh, but limited to the special problem of facade reconstruction.

Besides that, many approaches exist for detecting vegetation and excluding those parts of the point cloud for further processing when interested in man-made objects (Sedlacek and Zara, 2009; Gallup et al., 2010). Such specific aspects are not within scope of this work. We will assume the data to be preprocessed where necessary and that points which violate the sampling density assumption have been removed.

Therewith we clearly defined the objective of this thesis. For evaluating the performance of the proposed reconstruction framework later on, we will declare specific evaluation criteria in the following section.

### 3.1.4 Criteria for the experimental evaluation

The results of the segmentation and the reconstruction need to be evaluated w.r.t. prespecified criteria. We expect the reconstruction framework to meet the following requirements.

**Correctness of surface structure.** The reconstruction should yield a result that matches the truly underlying surface structure. This comprises accurate region boundaries without under- or oversegmentation as well as correct surface and relation classes. For synthetically generated meshes

this can be precisely evaluated – in contrast to real-world data sets, where ground-truth information is rarely available and the evaluation needs to be done per visual inspection.

**Accuracy of surface parameters.** The estimated surfaces should approximate the original data well. The estimation should be unbiased and minimize the point residuals.

**Sensitivity w.r.t. noise and outliers.** The system should be able to handle point clouds with coordinates disturbed by random noise, i.e. small deviations from their true position. Furthermore, the reconstruction should not be misled by outliers, i.e. points with large deviations from the underlying object surface.

**Sensitivity w.r.t. control parameters.** Control parameters required for the reconstruction process should be semantically perceivable by a user, such that they can be adjusted to control the result. Furthermore, they should be small in number and the result should not sensitively depend on their precise value.

**Computational complexity and scalability.** For the sake of practical relevance the computing time should be acceptable and the system should be scalable for large data sets. Within this thesis we do not focus on optimizing computing times, but pay attention to computational complexity and scalability.

**Flexibility.** The system should be designed such that it can be transferred to many different fields of applications, e.g. from aerial laser scans to desktop scenes, from laser scans to structured light reconstructions or from piecewise planar objects to more complex objects involving additional surface types like arbitrary quadrics or tori.

In Chapter 7 we will take up again these criteria and analyze the proposed framework accordingly.

With the detailed task of this thesis defined, we will introduce the concept of the proposed reconstruction framework. The following section will describe the model for man-made surfaces as well as the three processing levels.

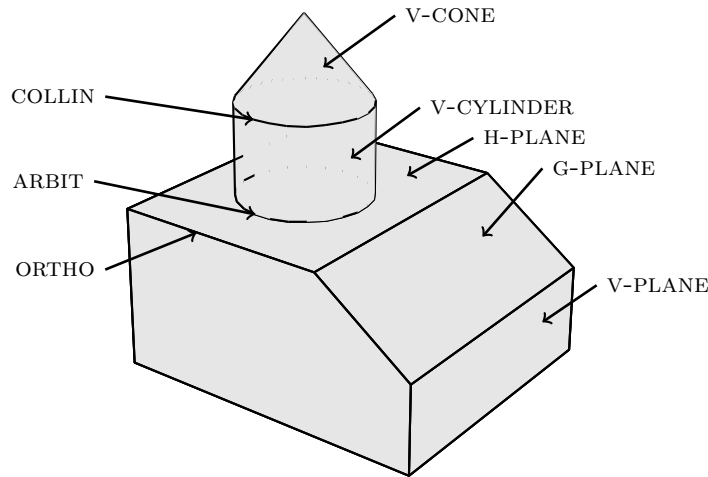
## 3.2 Concept

In this section we want to describe the concept of the proposed approach in more detail. Besides defining a concrete model for man-made surface structures comprising different types of surface and inter-surface relations, we will outline the three levels for segmenting the given mesh and reconstructing the surface structure.

### 3.2.1 Model for man-made surface structures

As previously indicated, we aim at deriving a surface segmentation with each region being describable with a planar or curved, but geometrically simple

primitive. To incorporate pair-wise relations between adjacent surface regions, we model them explicitly and consider constraints on corresponding surface parameters depending on their most likely relation. An example for the surface of a small object – as we would model it – is given in Fig. 3.3.



**Figure 3.3:** Model for man-made surface structures. A man-made surface consists of multiple planar or curved surface regions, which are classified into several types of surfaces as indicated by the annotations on the right-hand side. Adjacent regions are connected via explicit relations as shown on the left-hand side. The abbreviations are explained in Sections 3.2.1.1 and 3.2.1.2.

The surface and relation classes will be determined with a Bayesian approach incorporating the likelihood derived from given data points as well as prior information about class probabilities and distributions of surface parameters. In the following we will describe the model components for man-made surface structures in detail.

### 3.2.1.1 Surface classes

In this section we define particular surface classes that we intend to recognize and to reconstruct from given triangular meshes. For each surface class, also referred to as surface type or primitive type, we choose a parametrization together with a constraint which corresponding points need to fulfill. Additionally we will need to restrict the parameters of certain surface classes to a subspace in order to avoid degenerated primitives. Later in Chapter 6 we will use different prior probabilities in order to steer the model selection in the presence of more or less restrictive classes.

The simplest and most intuitive surface classes that occur in man-made scenes are planes. Since horizontal and vertical planes are contained very frequently, especially on buildings, we will model them as individual surface classes. These two special types of planes have less degrees of freedom and therefore a lower model complexity. We will design our reconstruction framework such that small approximation errors are accepted in favor of low model complexity, i.e. horizontal and vertical planes are chosen in place of general planes wherever possible. For

the sake of convenience and readability we will use the abbreviations H-PLANE, V-PLANE and G-PLANE for these three surface classes. The parameter vector  $\theta$  of a plane contains its normal vector  $\theta_n$  and its negative distance to the origin  $-\theta_d$ . While the normal of a V-PLANE is a 2D-vector in the horizontal plane, the normal of a H-PLANE is always perfectly vertical, thus vanishes in the parameter vector. The constraint for a point on a planar surface is usually  $\mathbf{X}^T \theta_n = \theta_d$ . For a V-PLANE this equation is computed for  $X$ - and  $Y$ -components only and for a H-PLANE only  $X_Z = \theta_d$  remains. In order to favor H-PLANE and V-PLANE, we penalize the G-PLANE class by prohibiting horizontal or vertical normal vectors, i.e. by restricting the zenith distance  $z = \arccos(\theta_{nz})$  to a range  $t_z < z < 90^\circ - t_z$  or  $90^\circ + t_z < z < 180^\circ - t_z$ .

Other common surface types are vertical and general cylinders (V-CYLINDER, G-CYLINDER), vertical cones (V-CONE) and spheres (SPHERE). As for planes, we model specially oriented cylinders and cones, since iterative and direct solutions for determining their parameters are simpler and computationally cheaper than for the general counterparts, and general cylinders and cones occur very rarely. Both SPHERE and V-CYLINDER are represented by a radius  $\theta_r$  and their center in 3D  $\theta_c$  or in 2D  $\theta_{c'}$ . Besides its 3D center  $\theta_c$  a V-CONE has a parameter  $\theta_A$  representing its squared slope, being related to the opening angle  $\alpha = \alpha(\theta_A)$ . Last but not least we represent the G-CYLINDER with a radius  $\theta_r$  and six Plücker coordinates  $\theta_L^T = [\theta_{L_n}^T \quad \theta_{L_0}^T]$  of the 3D line (Förstner and Wrobel, 2004, pg. 117ff.) representing the cylinder axis. The constraint for incident points usually involves the distance between the point  $\mathbf{X}$  and the primitive center  $\theta_c$ , that is equal to the radius  $\theta_r$  (SPHERE), equal to the radius  $\theta_r$  when projected into the horizontal (V-CYLINDER) or a slanted plane (G-CYLINDER) or linearly depending on the slope  $\sqrt{\theta_A}$  and the vertical coordinate difference (V-CONE). To prevent those curved surfaces from degenerating into planes, we penalize radii of spheres and cylinders larger than a threshold  $t_r$  and restrict the opening angle  $\alpha$  of a V-CONE to a range  $t_\alpha < \alpha < 180^\circ - t_\alpha$ . Further we control the orientation of the G-CYLINDER via its zenith angle, like we do for the G-PLANE.









Additionally we model FREEFORM surfaces: meshed surface regions without parametrization and without incidence constraint. To prevent every surface region to be classified as a FREEFORM surface – the approximation error will be zero –, we will use a smaller prior probability than for other surface classes. Thus, only if the approximation error with all other primitives is large and leads to small likelihoods for planar or curved primitives, the FREEFORM class will be chosen despite their low prior probability.

All surface classes described above are listed in Tab. 3.1. Besides the respective parametrization, the incidence constraint and the parameter restriction, a Jacobi matrix  $\mathbf{J}$  is given for the parameter reduction into the tangent space of homogeneous vectors  $\theta_r = \mathbf{J}\theta$  as previously described in Section 2.3.

### 3.2.1.2 Inter-surface relations

Additionally to the primitive types described in the previous section, we model five different relations between adjacent surface regions (Tab. 3.2). For two planar surfaces we allow orthogonality (ORTHO), identity (IDENT) and the arbitrary relation (ARBIT). For curved surfaces we additionally introduce collinearity (COLLIN), i.e. two axes are identical or one center is incident with an axis, and centering (CENTER), i.e. one center or one axis is incident with the other surface.

**Table 3.1:** Surface primitives for man-made surface structures. We define eight types of primitives: three different planes, four curved quadrics and a non-parametric FREEFORM surface. Besides the Euclidean or homogeneous parameter vector  $\theta$  or  $\boldsymbol{\theta}$ , respectively, this table contains the incidence constraint of corresponding 3D points  $\mathbf{X}$ , possibly a restriction of the parameter space depending on parameters  $t_z$ ,  $t_r$  and  $t_\alpha$  as well as the Jacobian matrix  $J$  for the transformation into the tangent space introduced in Section 2.3.2. 2D entities are marked with a prime; i.e.  $\mathbf{X}'$ ,  $\boldsymbol{\theta}_{n'}$  and  $\boldsymbol{\theta}_{c'}$  are the point coordinates, normal vector and center projected into the horizontal coordinate plane. Normal vectors  $\boldsymbol{\theta}_n$  as well as both Plücker elements  $\boldsymbol{\theta}_{L_h}$  and  $\boldsymbol{\theta}_{L_0}$  are normalized to unit length:  $|\boldsymbol{\theta}_n| = |\boldsymbol{\theta}_{L_h}| = |\boldsymbol{\theta}_{L_0}| = 1$ . The latter two elements need to fulfill the Plücker constraint  $\boldsymbol{\theta}_{L_h}^\top \boldsymbol{\theta}_{L_0} = 0$ .

Surface	Parametrization	Incidence constraint	Restriction	Reduction
 H-PLANE	$\theta = [\theta_d]$	$X_z = \theta_d$	—	$J = 1$
 V-PLANE	$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_{n'} \\ -\theta_d \end{bmatrix}$	$\mathbf{X}'^\top \boldsymbol{\theta}_{n'} = \theta_d$	—	$J = \begin{bmatrix} \text{null}^\top(\boldsymbol{\theta}_{n'}^\top) & 0 \\ \mathbf{0}^\top & 1 \end{bmatrix}$
 G-PLANE	$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_n \\ -\theta_d \end{bmatrix}$	$\mathbf{X}^\top \boldsymbol{\theta}_n = \theta_d$	$t_z < z(\boldsymbol{\theta}_n) < 90^\circ - t_z$ or $90^\circ + t_z < z(\boldsymbol{\theta}_n) < 180^\circ - t_z$	$J = \begin{bmatrix} \text{null}^\top(\boldsymbol{\theta}_n^\top) & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
 V-CYLINDER	$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_{c'} \\ \theta_r \end{bmatrix}$	$ \mathbf{X}' - \boldsymbol{\theta}_{c'}  = \theta_r$	$\theta_r < t_r$	$J = I_3$
 G-CYLINDER	$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_{L_h} \\ \boldsymbol{\theta}_{L_0} \\ \theta_r \end{bmatrix}$	$\frac{ \boldsymbol{\theta}_{L_h} \times \mathbf{X} + \boldsymbol{\theta}_{L_0} }{ \boldsymbol{\theta}_{L_h} } = \theta_r$	$t_z < z(\boldsymbol{\theta}_n) < 90^\circ - t_z$ or $90^\circ + t_z < z(\boldsymbol{\theta}_n) < 180^\circ - t_z$ , $\theta_r < t_r$	$J = \begin{bmatrix} \text{null}^\top \left( \begin{bmatrix} \boldsymbol{\theta}_{L_h} & \boldsymbol{\theta}_{L_0} \\ \boldsymbol{\theta}_{L_0} & \boldsymbol{\theta}_{L_h} \end{bmatrix}^\top \right) & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
 V-CONE	$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_c \\ \theta_A \end{bmatrix}$	$\frac{ \mathbf{X}' - \boldsymbol{\theta}_{c'} ^2}{ X_z - \theta_{cz} ^2} = \theta_A$	$t_\alpha < \alpha(\theta_A) < 180^\circ - t_\alpha$	$J = I_4$
 SPHERE	$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_c \\ \theta_r \end{bmatrix}$	$ \mathbf{X} - \boldsymbol{\theta}_c  = \theta_r$	$\theta_r < t_r$	$J = I_4$
 FREEFORM	$\boldsymbol{\theta} = []$	—	—	—

For each relation we define a number of possible unordered pairs of surface classes. Consequently, all combinations of two adjacent surface classes and one relation not listed in Tab. 3.2 will not be recognized. Note that this list of combinations is not necessarily complete and could be supplemented with, e.g., touching or parallel primitives. Other topologically possible combinations are excluded, since they do not occur naturally, partly due to very complicated 3D intersection lines that can be fourth degree curves.

The constraints on the two involved parameter vectors  $\theta$  and  $\theta'$  are either orthogonalities  $\perp$ , equalities  $=$  or parallelisms  $\parallel$ . The latter is the case for homogeneous vectors that are equal up to scale. Some relations do not introduce additional constraints, since they are either implicitly fulfilled, e.g. H-PLANE and V-PLANE are ORTHO anyway, or the relation is designed as such: the ARBIT relation. In order to avoid all pairs of adjacent regions being classified as arbitrarily related, we will use a smaller prior probability for ARBIT than for other relations later in Chapter 6.

This concludes the proposed model for man-made surface structures. It is taken as a basis for the surface classification and constraint parametrization in Chapter 6.

### 3.2.2 Reconstruction framework at three levels

Before describing the individual steps of the proposed reconstruction framework in detail, we want to summarize the three processing levels. They combine a low-level data-driven pre-segmentation of single vertices, a mid-level segmentation refinement focusing on surface regions as well as a high-level model-driven reconstruction working on the global surface structure and involving model knowledge (Fig. 3.4):

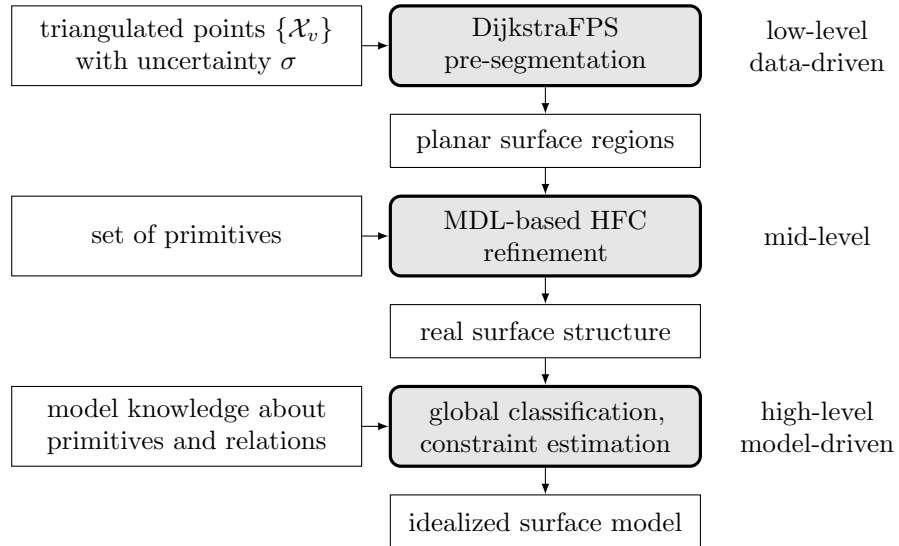
1. **Low-level pre-segmentation (Chapter 4).** On the lowest level we look at vertices of the triangular mesh, their local neighborhood and derived surface properties like normals and curvature as described in Section 2.1.2. We perform a pre-segmentation based on shortest paths on surfaces (Section 2.2). The pre-segmentation assumes local planarity only and cannot well represent curved regions or even inter-regional relations. The computational complexity, however, is small; thus we quickly reduce the amount of data for the next reconstruction level.
2. **Mid-level segmentation (Chapter 5).** We then introduce different surface primitives and hierarchically merge adjacent regions that can be well approximated with a single primitive. The decisions, when to merge two regions, are made based on their description length (Section 2.4). Due to the previous pre-segmentation this step quickly converges to an optimal segmentation given a predefined set of possible primitive types. Although a model selection already takes place, we do not model inter-regional relations yet and obtain a *real*, data-driven surface structure.
3. **High-level reconstruction (Chapter 6).** Assuming an almost perfect segmentation from the previous levels, we finally classify surface regions and relations between adjacent pairs of them using inference on a graphical model (Section 2.5). This classification makes use of available model



**Table 3.2:** Surface relations for man-made surface structures. We define five relations for adjacent pairs of surfaces. For each relation this table contains a list of unordered pairs of surfaces that are topologically possible and likely to occur in combination with the respective relation. The last column displays the constraint induced on the two involved parameter vectors  $\theta$  and  $\theta'$ . As in Tab. 3.1 the prime denotes 2D projections of a center  $\theta_{c'}$  or the normal vector  $\theta_{n'}$  into the horizontal coordinate plane. All constraints are either formulated as orthogonalities  $\perp$ , equalities  $=$  or parallelisms  $\parallel$ . Some combinations induce no constraint at all.

Relation	Surface pairs	Constraint
ORTHO	G-PLANE, G-PLANE	$\theta_n \perp \theta'_n$
	G-PLANE, V-PLANE	$\theta_{n'} \perp \theta'_{n'}$
	V-PLANE, V-PLANE	
	G-CYLINDER, G-PLANE	$\theta_{L_h} \parallel \theta'_n$
	G-CYLINDER, G-CYLINDER	$\theta_{L_h} \perp \theta'_{L_h}$
	H-PLANE, V-PLANE	—
	H-PLANE, V-CYLINDER	
	H-PLANE, V-CONE	
IDENT	H-PLANE, H-PLANE	$\theta = \theta'$
	SPHERE, SPHERE	
	V-CONE, V-CONE	
	V-CYLINDER, V-CYLINDER	$\theta \parallel \theta'$
	V-PLANE, V-PLANE	
COLLIN	V-CONE, V-CYLINDER	$\theta_{c'} = \theta'_{c'}$
	SPHERE, V-CYLINDER	$\begin{bmatrix} \theta_{c'} \\ \theta_r \end{bmatrix} \perp \theta'$
CENTER	H-PLANE, SPHERE	$\theta_d = \theta'_{c_z}$
	SPHERE, V-PLANE	$\begin{bmatrix} \theta_{c'} \\ 1 \end{bmatrix} \perp \theta'$
	V-CONE, V-PLANE	
	V-CYLINDER, V-PLANE	$\theta \perp \begin{bmatrix} \theta'_c \\ 1 \end{bmatrix}$
G-PLANE, SPHERE		
ARBIT	H-PLANE, G-PLANE	—
	V-PLANE, V-PLANE	
	G-PLANE, V-PLANE	
	G-PLANE, G-PLANE	
	G-PLANE, G-CYLINDER	
	H-PLANE, G-CYLINDER	
	H-PLANE, SPHERE	
	G-PLANE, V-CYLINDER	
	FREEFORM, *	

knowledge and yields constraints on surface parameters, which are estimated in a concluding parameter estimation using the Gauss-Helmert model (Section 2.3). In contrast to the model obtained after the mid-level reconstruction step, the incorporation of model knowledge now yields an *idealized* model-driven surface structure.



**Figure 3.4:** Three levels of the proposed reconstruction framework. During a data-driven pre-segmentation we obtain a compound of planar surface regions. It is refined by introducing a set of different primitives. Finally, the surface structure is idealized with a constraint parameter estimation based on a global classification using further model knowledge.

Therewith we defined the model for man-made surface structures and outlined the three reconstruction levels. In the following chapters we will describe each of them in more detail.

---

## Segmentation via Dijkstra farthest point sampling

---

The surface segmentation algorithm proposed in this chapter aims at partitioning a given surface mesh into planar regions. It is inspired by a surface sampling strategy called farthest point sampling (FPS): Given a current sampling with  $L$  seed points of an underlying surface we add the *farthest* point, i.e. the one with largest distance to the closest seed point w.r.t. a carefully chosen metric, yielding a new sampling. This strategy implicitly yields a surface segmentation in form of the corresponding Voronoi diagram. Due to the distance transform based on Dijkstra's algorithm we use the term *DijkstraFPS*.

The following sections describe the core algorithm and the distance metric in more detail. Multiple advanced sampling strategies as well as distance metrics are presented to improve the robustness to noisy data.

---

<b>4.1 Surface segmentation via farthest point sampling . . . . .</b>	<b>63</b>
<b>4.2 Advanced sampling strategies . . . . .</b>	<b>66</b>
4.2.1 Incremental-decremental sampling . . . . .	67
4.2.2 Automatic stopping criterion . . . . .	69
4.2.3 Refinement with Lloyd iterations . . . . .	72
<b>4.3 Advanced distance metrics . . . . .</b>	<b>74</b>
4.3.1 Non-planar regions . . . . .	75
4.3.2 Extrinsic distance metrics . . . . .	76

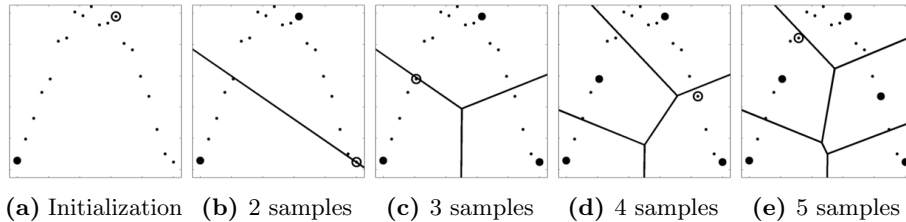
---

### 4.1 Surface segmentation via farthest point sampling

The proposed surface segmentation approach relies on the equivalence of a surface sampling with  $L$  sampling points and the  $L$  corresponding Voronoi cells that are formed by assigning each vertex to its closest sampling point w.r.t. a predefined

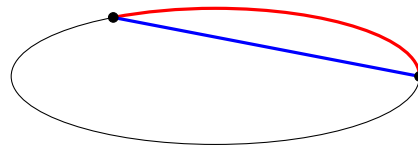
distance metric. The crucial point is how to find reasonably distributed sample points that represent the surface well and, likewise, yield a proper segmentation. After explaining the concept of segmentation via point sampling in 2D space and introducing an appropriate metric for 3D surfaces, we will present the DijkstraFPS surface segmentation algorithm.

Moening and Dodgson (2003a,b) propose a strategy for surface sampling called farthest point sampling (FPS, Fig. 4.1). The FPS strategy suggests to start with one random sample point and the corresponding, trivial Voronoi diagram consisting of one region only. Now the surface is differently well represented: Close to the one sample point the sampling might be sufficient since the distance of surrounding points to the next sample is small; other areas are not sampled at all, i.e. the points are far apart from the next sample. To improve the current sampling the FPS strategy suggests to add the *farthest point* as a new sampling point. This step is repeated  $L - 1$  times until a suitable sampling or segmentation with  $L$  regions is reached.



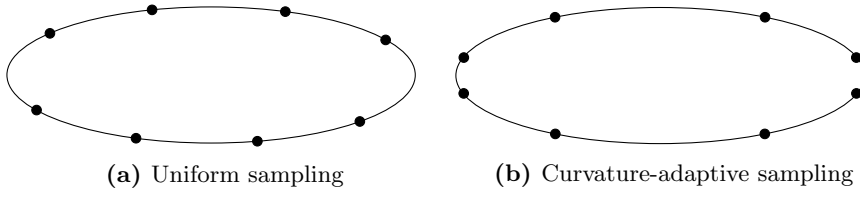
**Figure 4.1:** Farthest point sampling. After initializing with one random point we repeatedly sample the farthest point (white circles) to all previously sampled points (bold dots). Meanwhile the space is divided into Voronoi cells.

Peyré and Cohen (2004, 2006) apply a similar farthest point strategy on meshed surfaces for segmentation, re-meshing and surface flattening. In terms of surface segmentation we do not sample an affine space but a 2D manifold embedded in the 3D space. Thus, we need to distinguish between extrinsic and intrinsic distance metrics: The distance between two points is defined as the length of the shortest path connecting them. When using an extrinsic Euclidean metric, this path is a straight line, thus the distance is the Euclidean distance. In contrast, an intrinsic metric is only defined in a subspace, e.g. the 2D surface manifold. Therefore, the shortest path is completely embedded in the surface and – in case of a uniform and isotropic metric – yields the geodesic distance. Fig 4.2 illustrates the difference on a 1D curve embedded in the 2D plane.



**Figure 4.2:** Intrinsic and extrinsic distances. The extrinsic distance (blue) is the length of the shortest path between two points (bold dots), which is a straight line for the case of a Euclidean metric. In contrast, the intrinsic distance (red) is the length of the shortest path *within* a given manifold (black line).

Note that intrinsic distances need not necessarily be geodesic distances in terms of Euclidean lengths, but the integral over the local distance metrics along the path. With the FPS strategy a Euclidean metric would yield evenly distributed sample points, independent of the underlying surface shape (Fig. 4.3a). Within the context of segmenting surfaces for reconstructing objects we are, however, interested in planar or at least geometrically uniform regions. While planar regions can remain untouched, heavily undulated regions need further refinement, i.e. more sample points (Fig. 4.3b). Thus, the chosen distance measure should reflect the amount of undulation between a vertex and its closest sample point.



**Figure 4.3:** Uniform and curvature-adaptive sampling. Both 1D curves (black lines) embedded in the 2D space are sampled with  $V = 8$  points (bold dots). A uniform sampling (a) yields the same geodesic distance between each pair of adjacent sample points. In contrast, a curvature-adaptive sampling (b) yields smaller geodesic distances at areas with larger curvature and vice versa.

In Section 2.2 we discussed two algorithms for computing intrinsic distances on meshed surfaces based on propagating wave fronts. The Eikonal equation (2.21) relates distance or path lengths  $D$  with local distances  $d$ , also referred to as friction or inverse speed  $F$ . Integrating over the Euclidean length  $s$  of a continuous path yields the distance

$$D = \int_s F ds. \quad (4.1)$$

On discrete triangular meshes this becomes the sum over all visited vertices  $u$

$$D = \sum_u \underbrace{F_u \cdot s_u^v}_{d_u^v} \quad (4.2)$$

with the local friction  $F_u$  multiplied with the Euclidean distance  $s_u^v$  between vertices  $\mathcal{V}_u$  and  $\mathcal{V}_v$ .

To achieve a curvature-adaptive sampling as in Fig. 4.3b, we define the friction  $F_u$  proportional to the curvature  $\kappa_u$  along the path at point  $\mathcal{X}_u$  or – in terms of propagation speed – the speed needs to be inversely proportional to the curvature  $\kappa_u$ . The curvature is the inverse radius of the osculating circle  $\kappa_u = \frac{1}{r_u}$  and the radius  $r_u$  relates to the arc length  $s_u$  and the angle between both vertex normals  $\mathbf{n}_u$  and  $\mathbf{n}_v$  (Section 2.1.2) as

$$s_u^v = r_u^v \cdot \angle(\mathbf{n}_u, \mathbf{n}_v). \quad (4.3)$$

Therefore, the curvature-adaptive pair-wise distances  $d_u^v$  are

$$d_u^v = \kappa_u^v \cdot s_u^v = \frac{s_u^v}{r_u^v} = \angle(\mathbf{n}_u, \mathbf{n}_v) \quad (4.4)$$

yielding distances

$$D = \sum_u \angle(\mathbf{n}_u, \mathbf{n}_v) \quad (4.5)$$

being the sum over pair-wise normal angles of all vertices  $u$  along the respective path. This metric supports planar regions with constant normal direction, but yields large distances on curved regions. Later in Section 4.3.1 we will introduce advanced metrics, e.g. difference of curvature, that might be better suited for surface regions with varying normals but constant curvature.

In Section 2.2 we discussed two common distance transforms: Dijkstra’s algorithm (Dijkstra, 1959) and the fast marching method (FMM, Sethian, 1996). Moenning and Dodgson (2003a,b) use the more precise distance transform FMM, suggesting the name FastFPS for the overall sampling algorithm. Dijkstra’s algorithm, however, is slightly faster, applicable to more general graph topologies and yields sufficiently accurate results (Fig. 2.5). Therefore, we choose Dijkstra’s algorithm in favor of FMM (Schindler and Förstner, 2011) and refer to the segmentation algorithm with the term *DijkstraFPS* throughout this thesis.

Alg. 4.1 and Fig. 4.4 demonstrate the DijkstraFPS surface segmentation algorithm. After each new seed vertex both the distance map  $\mathbf{D}$  and the vertex labeling  $\mathbf{l}$  are updated using Dijkstra’s algorithm (Section 2.2).

---

**Algorithm 4.1:** Incremental DijkstraFPS surface segmentation. New seed vertices  $u_0$  are added iteratively by choosing the farthest vertex w.r.t. distances  $\mathbf{D}$  that are constantly updated using Dijkstra’s algorithm (Alg. 2.1) with pair-wise distances  $d_u^v$ , yielding a vertex labeling  $\mathbf{l}$ .

---

**Input:** neighbors  $\text{ne}(u)$ , pair-wise distances  $d_u^v$ , number of regions  $L_{\text{inc}}$

**Output:** distance map  $\mathbf{D}$ , vertex labels  $\mathbf{l}$

initialize distances  $\mathbf{D} \leftarrow \infty$  and labels  $\mathbf{l} \leftarrow \text{NaN}$

**for**  $l^+ \leftarrow 1$  **to**  $L_{\text{inc}}$  **do**

    pick farthest point as new seed vertex  $u_0 \leftarrow \text{argmax}_u D_u$

    set distance  $D_{u_0} \leftarrow 0$  and label  $l_{u_0} \leftarrow l^+$

    initialize front  $\mathcal{Q} \leftarrow \{u_0\}$

    update distances  $\mathbf{D}$  and labels  $\mathbf{l}$  via Dijkstra’s algorithm (Alg. 2.1)

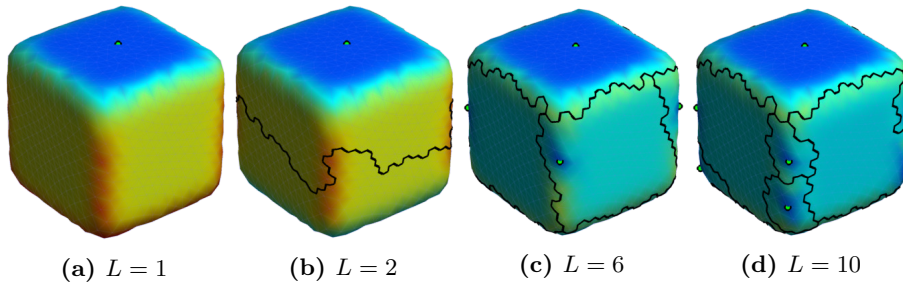
**end**

---

Apparently this algorithm is already sufficient for segmenting simple, piece-wise planar surfaces. Given the correct number of regions  $L$ , all region boundaries approximately correspond to object edges. Generating more regions yields an oversegmentation. As we will see in the following section, we can improve the segmentation and automatically detect the correct number of regions with advanced sampling strategies.

## 4.2 Advanced sampling strategies

In the previous section we demonstrated the incremental DijkstraFPS surface segmentation algorithm: We applied an incremental distance transform on triangulated manifolds to generate an intrinsic distance map  $\mathbf{D}$  and its corresponding Voronoi segmentation  $\mathbf{l}$ . There is, however, the possibility to also decrementally remove regions. Beyond the immediate use for improving the incremental segmentation during an incremental-decremental sampling scheme we will present



**Figure 4.4:** Surface segmentation with DijkstraFPS on a synthetically generated 8-norm sphere with  $V = 1000$  vertices. Starting from a random seed point (green dot on top) the intrinsic distance to each other surface point is computed with Dijkstra’s algorithm (a). The distance is color-coded from near (blue) to far (red). Using a curvature-adaptive metric, implemented as pair-wise distances  $d_u^v$  between two vertices  $u$  and  $v$ , we obtain small gradients along planar regions and large gradients at sharp edges. Choosing the farthest point as additional seed point (on bottom) and repeating Dijkstra’s algorithm we obtain a segmentation into two regions (b). The second wave front stops somewhere in the middle when newly computed distances exceed distances from the first iteration. After six iterations the cube is roughly segmented into its six faces (c). Another four iterations yield a clear oversegmentation (d).

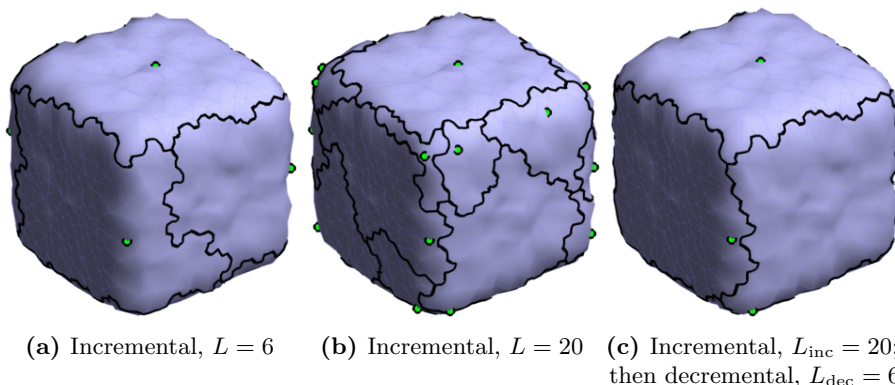
two advanced concepts making use of decremental sampling: determining the number of regions for an optimal segmentation as well as refining a segmentation via Lloyd iterations.

### 4.2.1 Incremental-decremental sampling

Due to the random initialization and possibly noisy edge attributes along the paths, the Voronoi segmentation boundaries might not be perfectly aligned with the object edges (Fig. 4.5a), although the supposedly correct number of regions  $L$  is reached. This is also because in terms of undulation a vertex on a rounded edge is far away from both adjacent object faces. Therefore, it is likely to be chosen as farthest point, leading to a Voronoi region centered on an edge, displacing Voronoi boundaries towards the object’s planar faces. This effect cannot be corrected without removing this seed vertex.

Experiments showed that we can overcome this problem by 1) adding more seeds in order to increase the chance of having at least *some* Voronoi boundaries coinciding with true object edges and 2) removing seeds again until the desired number of regions  $L$  is reached (Schindler and Förstner, 2011). The first step is identical to the incremental DijkstraFPS algorithm yielding a clear oversegmentation (Fig. 4.5b). But the second step is very similar as well: Instead of adding the farthest point as new seed, we remove one region  $l^-$  by setting the distance of its vertices  $\{\mathcal{V}_v \mid l_v = l^-\}$  to infinity  $D_v := \infty$  and removing their labels, e.g. assigning the value “not a number”:  $l_v := \text{NaN}$ . Then we update the distance map  $\mathbf{D}$  using Dijkstra’s algorithm starting at vertices adjacent to the removed region  $l^-$ .

Usually the regions to be removed are located on object edges. They are small, since their boundary is stopped very early when approaching planar object faces. We therefore obtain reasonable results by simply choosing the smallest



**Figure 4.5:** Incremental-decremental surface segmentation on a synthetically generated 8-norm sphere with  $N = 1000$  vertices and  $\sigma = 1\%$  Gaussian noise w.r.t. the coordinate range. The rounded edges and the large noise lead to an incorrect segmentation after  $L = 6$  incremental DijkstraFPS iterations (a). Continuing the incremental segmentation creates segmentation boundaries on all object edges, but introduces an oversegmentation as well (b). Decrementally removing 14 regions eliminates the oversegmentation (c) and yields more accurate segmentation boundaries than with the purely incremental approach in (a).

region in terms of the number of corresponding vertices, whenever a region is to be removed (Fig. 4.5c). The decremental algorithm is given in Alg. 4.2.

---

**Algorithm 4.2:** Decremental DijkstraFPS surface segmentation. Until reaching the prespecified number of regions  $L_{\text{dec}}$  we choose the smallest region  $l^-$  for removal. The distance map  $\mathbf{D}$  and vertex labels  $\mathbf{l}$  are updated by propagating a front  $\mathcal{Q}$  inwards starting from the boundary of region  $l^-$ .

---

**Input:** neighbors  $\text{ne}(u)$ , pair-wise distances  $d_u^v$ , number of regions  $L_{\text{dec}}$

**Updates:** distance map  $\mathbf{D}$ , vertex labels  $\mathbf{l}$

**while**  $L > L_{\text{dec}}$  **do**

    find smallest region  $l^- \leftarrow \text{argmin}_l |\{u \mid l_u = l\}|$

    set distances  $D_u \leftarrow \infty$  and labels  $l_u \leftarrow \text{NaN}$  for vertices  $\{u \mid l_u = l^-\}$

    fill front  $\mathcal{Q} \leftarrow \{v\}$  with neighbors  $v$  of region  $l^-$ :  $v \in \text{ne}(u)$ ,  $l_u = l^-$ ,  $l_v \neq l^-$

    update distances  $\mathbf{D}$  and labels  $\mathbf{l}$  via Dijkstra's algorithm (Alg. 2.1)

**end**

---

Our overall incremental-decremental segmentation strategy is as follows:

1. We incrementally segment the surface until a prespecified number of regions  $L_{\text{inc}}$  is reached or a stopping criterion is fulfilled. Now all object edges should be covered with Voronoi boundaries or with small regions, respectively. Note that only the first iterations affect many vertices (Reem, 2011) and thus oversegmenting the surface only marginally increases the computing time.
2. We decrementally segment the surface until a prespecified number of regions  $L_{\text{dec}}$  is reached or a stopping criterion is fulfilled.



The strategy allows the user to define the expected number of regions  $L_{\text{inc}}$  and  $L_{\text{dec}}$ , respectively. In the following section we will formulate an *automatic* stopping criterion based on the description length  $\Phi_L$ .

### 4.2.2 Automatic stopping criterion

When searching for the optimal number of planar regions  $L$  we need to find a trade-off between fitting error – i.e. how well the points in a region  $l$  are approximated by a best fitting plane – and model complexity – i.e. the number of regions and plane parameters. Both aspects can be jointly formulated using the concept of minimum description length (MDL) introduced in Section 2.4. Evaluating the description length after each incremental and decremental segmentation step and detecting local minima will allow us to automatically stop at an optimal number of regions.

In analogy with Förstner (1989) and in accordance with (2.60) the description length for  $V$  vertices on  $L$  planes in 3D space is

$$\Phi_L = \underbrace{2V \text{lb} \frac{r}{\varepsilon}}_{\text{in-plane locations}} + \underbrace{\sum_{v=1}^V \left( \frac{1}{2 \ln 2} \cdot \frac{v_v^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right)}_{\text{off-plane residuals}} + \underbrace{3L \text{lb} \frac{r}{\varepsilon}}_{\text{plane parameters}} + \underbrace{V \text{lb} L}_{\text{labeling}} \quad (4.6)$$

with the following components:

**In-plane locations.** The description length for storing one coordinate that is uniformly distributed across a range  $r$  and rounded to a precision  $\varepsilon$  requires  $\text{lb} \frac{r}{\varepsilon}$  bits. Since each of the  $V$  points has two uniformly distributed coordinates along the plane, we need  $2V \text{lb} \frac{r}{\varepsilon}$  bits for storing all of them.

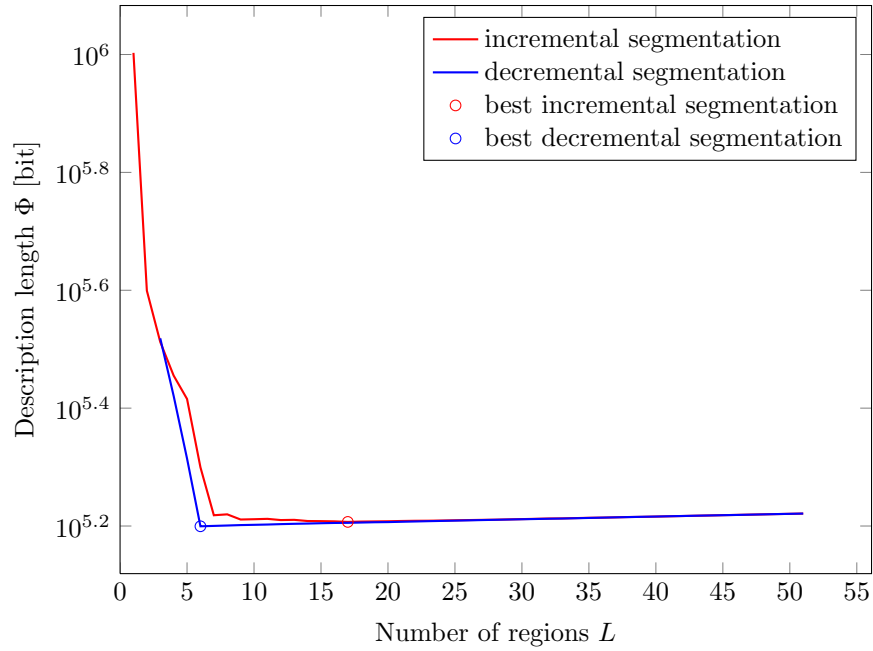
**Off-plane residuals.** Additionally we need to store the third degree of freedom of each point, namely the residual perpendicularly to the surface. This residual is Gaussianly distributed and rounded to precision  $\varepsilon$ , thus requires  $\frac{1}{2 \ln 2} \cdot \frac{v_v^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon}$  bits for each point  $v = 1, \dots, V$ .

**Planes.** We need to store the uniformly distributed plane parameters as well. Regardless of the chosen representation, we have three independent parameters for each plane  $l = 1, \dots, L$ . Assuming the same range  $r$  and precision  $\varepsilon$  as for the coordinates above, we obtain  $3L \text{lb} \frac{r}{\varepsilon}$  bits for all parameters.

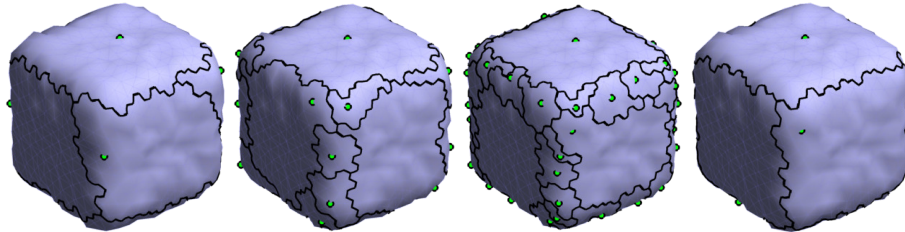
**Labeling.** For each point we need to store a label  $l \in \{1, \dots, L\}$ , each of which requires  $\text{lb} L$  bits. Having  $V$  vertices in total we need  $V \text{lb} L$  bits for storing the whole labeling  $l$ .

We claim that by minimizing this description length we obtain a visually pleasing segmentation.

As depicted in Fig. 4.6a we can compute the description length for the current segmentation after each incremental or decremental iteration. As expected, the description length decreases rapidly during the first iterations (red line). Adding a seventh seed point still improves the result significantly – there is no chance to recognize the true number of object faces at this point. The description length reaches a minimum at  $L^* = 17$  regions before slowly increasing again.



(a) Description length during incremental (red) and decremental (blue) segmentation



(b) 6 regions

(c) 17 regions

(d) 51 regions

(e) 6 regions again

**Figure 4.6:** Incremental-decremental segmentation with an automatic stopping criterion on a synthetically generated 6-norm sphere with  $V = 1000$  vertices and  $\sigma = 1\%$  Gaussian noise w.r.t. the coordinate range. After reaching  $L = 51$  regions (d) – three times the number of regions of the currently best segmentation with  $L^* = 17$  regions (c) –, we decrementally segment until again reaching an optimal segmentation with  $L^* = 6$  regions (e). The segmentation is visually and quantitatively better than after an incremental segmentation only (b).

As mentioned above, adding more regions is not expensive. Thus, we keep on segmenting the surface incrementally to make sure that we found a global minimum and that the oversegmentation is large enough. Whenever we find a segmentation with  $L$  regions yielding a smaller description length  $\Phi_L < \Phi_{L^*}$  than the currently best segmentation with  $L^*$  regions, we will adjust the upper bound for the number of regions  $L_{\text{inc}} := 3L$ . We stop as soon as reaching the upper bound  $L = L_{\text{inc}}$ , which is the case at  $L = 51$  in Fig. 4.6a.

We now start to decrementally remove regions (Fig. 4.6a, blue line). Interestingly the description length reaches a minimum at 6 rather than 17 regions with a much lower description length than during the incremental segmentation. To make sure to have reached a global minimum we store the result  $\mathbf{l}^*$  with  $L^* = 6$  regions and continue the decremental segmentation. Whenever we find a segmentation with smaller description length  $\Phi_L < \Phi_{L^*}$ , we will adjust the lower bound for the number of regions  $L_{\text{dec}} := \lfloor \frac{L}{2} \rfloor$ . We stop as soon as we reach the lower bound  $L = L_{\text{dec}}$ , which is the case at  $L = 3$  in Fig. 4.6a. Then the segmentation  $\mathbf{l}^*$  with  $L^* = 6$  regions is returned as the final result. The incremental-decremental segmentation strategy is presented in Alg. 4.3.

---

**Algorithm 4.3:** Incremental-decremental DijkstraFPS segmentation strategy. The description length is used as a stopping criterion for both the incremental and the decremental segmentation step. In order to avoid local minima, the upper and lower bounds  $L_{\text{inc}}$  and  $L_{\text{dec}}$  are loosely set to  $3L$  and  $\lfloor \frac{L}{2} \rfloor$ , respectively.

---

**Input:** neighbors  $\text{ne}(u)$ , pair-wise distances  $d_u^v$

**Output:** vertex labels  $\mathbf{l}^*$

initialize labeling  $L \leftarrow 0$ ,  $\mathbf{l} \leftarrow \mathbf{0}$  and compute  $\Phi_L$

initialize currently best segmentation  $\mathbf{l}^* \leftarrow \mathbf{l}$ ,  $L^* \leftarrow L$  and  $\Phi_{L^*} \leftarrow \Phi_L$

initialize upper bound for the number of regions  $L_{\text{inc}} \leftarrow \infty$

**while** upper bound not reached  $L < L_{\text{inc}}$  **do**

    add a region  $L \leftarrow L + 1$  using incremental DijkstraFPS (Alg. 4.1)

    update description length  $\Phi_L$

**if**  $\Phi_L < \Phi_{L^*}$  **then**

        update best segmentation  $\mathbf{l}^* \leftarrow \mathbf{l}$ ,  $L^* \leftarrow L$  and  $\Phi_{L^*} \leftarrow \Phi_L$

        update upper bound  $L_{\text{inc}} \leftarrow 3L$

**end**

**end**

initialize lower bound for the number of regions  $L_{\text{dec}} \leftarrow 1$

**while** lower bound not reached  $L > L_{\text{dec}}$  **do**

    remove a region  $L \leftarrow L - 1$  using decremental DijkstraFPS (Alg. 4.2)

    update description length  $\Phi_L$

**if**  $\Phi_L < \Phi_{L^*}$  **then**

        update best segmentation  $\mathbf{l}^* \leftarrow \mathbf{l}$ ,  $L^* \leftarrow L$  and  $\Phi_{L^*} \leftarrow \Phi_L$

        update lower bound  $L_{\text{dec}} \leftarrow \lfloor \frac{L}{2} \rfloor$

**end**

**end**

---

In Fig. 4.6b–4.6e we see the corresponding segmentation results: During the incremental segmentation,  $L = 17$  regions (Fig. 4.6c) yield the minimum description length. But during the decremental segmentation starting with  $L_{\text{inc}} = 51$  regions (Fig. 4.6d) the description length is minimized at  $L^* = 6$

regions (Fig. 4.6e), yielding a visually much more pleasing result than at  $L = 6$  regions during the incremental segmentation (Fig. 4.6b).

The combination of subsequent incremental and decremental segmentations can be repeated in order to find an even better result with smaller description length. Experiments show that one or two repetitions are sufficient in most cases. Nevertheless, we will anyway apply a second segmentation approach in Chapter 5 to improve the result by further reducing a possible oversegmentation and better aligning the region boundaries with the object's edges.

Fig. 4.7 illustrates the final result using the incremental-decremental DijkstraFPS surface segmentation with description length minimization. While the roundness of the cube's edges decreases from left to right, the synthetically added Gaussian noise increases from top to bottom. The roundness is controlled by normalizing all point vectors  $\mathbf{X}_v$  accordingly with 4-, 6-, 8-, 10- and 12-norm. With an underlying standard deviation of  $\sigma = 0.05$  % w.r.t. the coordinate range, even a 12-norm sphere does not result in a segmentation with six regions only. The points are too certain, thus their residuals w.r.t. only six planes would be too expensive in terms of description length. More noise, however, can explain the round edges. With  $\sigma = 0.20$  % the 12-norm sphere is segmented as a cube. With  $\sigma = 2.00$  % even the 4-norm sphere yields six regions only.

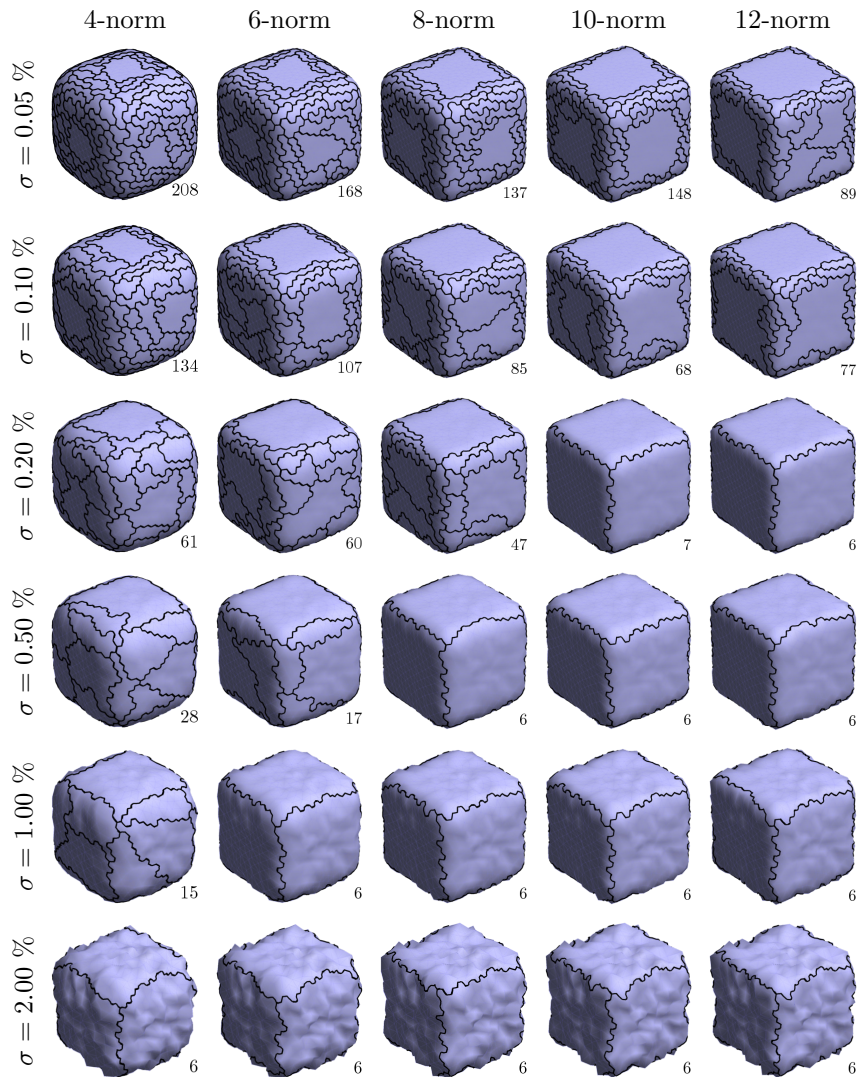
As demonstrated in Fig. 4.7 and also previously shown by Schindler and Förstner (2011), the incremental-decremental DijkstraFPS strategy yields robust surface segmentations even on noisy data sets. The following section will present another strategy to exploit a combination of incremental and decremental segmentation steps to improve an existing segmentation.

### 4.2.3 Refinement with Lloyd iterations

Another strategy combining incremental and decremental segmentation is to apply Lloyd iterations (Lloyd, 1982), also known as  $k$ -means clustering, as proposed by Cohen-Steiner et al. (2004) as well as by Peyré and Cohen (2004, 2006). The idea is to relocate the seed vertices as the intrinsic centroids of each Voronoi region. By updating the seed vertices the Voronoi boundaries change as well, stabilizing the overall segmentation and reducing effects of the random initialization and the incremental segmentation process.

During one Lloyd iteration we process all regions sequentially: We remove one region  $l$  by setting the distances of its vertices  $\{\mathcal{V}_v \mid l_v = l\}$  to infinity  $D_v := \infty$  and initialize a Dijkstra front  $\mathcal{Q} = \{\mathcal{V}_u\}$  with zero distances  $D_u := 0$  on all vertices  $u$  that are adjacent to region  $l$ , i.e.  $\exists v \in \text{ne}(u) \mid l_v = l$  and  $l_v \neq l_u$ . Then we propagate the front towards the empty region using a Euclidean metric  $s_u^v$  and Dijkstra's algorithm (Alg. 2.1). Using the curvature-adaptive metric  $d_u^v$  for this step might be reasonable as well, but empirically the Euclidean metric yields more stable results. Afterwards we store the farthest point as new seed point  $v_{0,l}$  of this region  $l$ . After obtaining new seed points for all regions, we derive a new Voronoi diagram using all updated seeds, Dijkstra's algorithm and the curvature-adaptive distance metric  $d_u^v$ . This process is shown in Alg. 4.4.

We can perform one Lloyd iteration only, repeat it multiple times or repeat it until the segmentation remains unchanged after another iteration. Usually – when starting with a reasonably accurate initialization – we observe convergence after very few iterations (Fig. 4.8).



**Figure 4.7:** Number of regions  $L$  (number in the lower right corner) automatically determined by DijkstraFPS for cubes with  $V = 1000$  vertices, different roundness and different Gaussian noise  $\sigma$ . Given a standard deviation of only  $\sigma = 0.05\%$  w.r.t. the coordinate range, all deviations from a piece-wise planar object like the rounded corners need to get described with additional regions, thus we obtain  $L > 6$  regions. Large uncertainty, however, can explain such model violations. Therefore, the minimum description length for  $\sigma = 2.00\%$  noise is achieved with six regions – no matter how sharp the cube’s edges actually are.

---

**Algorithm 4.4:** DijkstraFPS Lloyd iteration. After decrementally removing each region  $l$  and storing the farthest point  $v_{0,l}$  within region  $l$  according to a Euclidean metric  $s_u^v$ , an improved segmentation is obtained using Dijkstra’s algorithm with a front starting at the seeds  $v_{0,l}$ . This process can be iterated.

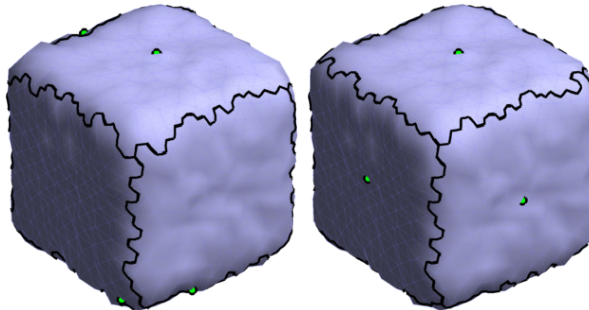
---

```

foreach region  $l$  do
  set distances  $D_v \leftarrow \infty$  for vertices  $\{v \mid l_v = l\}$ 
  set front  $\mathcal{Q} \leftarrow \{\mathcal{V}_u\}$  and distances  $D_u \leftarrow 0$  for adjacent vertices  $u$ 
  apply Dijkstra’s algorithm (Alg. 2.1) with Euclidean metric  $s_u^v$ 
  store farthest vertex as new seed  $v_{0,l}$ 
end
re-initialize all labels  $l_v \leftarrow NaN$  and distances  $\mathbf{D} \leftarrow \infty$ 
apply Dijkstra’s algorithm (Alg. 2.1) with front  $\mathcal{Q} \leftarrow \{v_{0,l}\}$ 

```

---



(a) No Lloyd iterations      (b) Two Lloyd iterations

**Figure 4.8:** DijkstraFPS surface segmentation with Lloyd iterations on a synthetically generated 8-norm sphere with  $V = 1000$  vertices and  $\sigma = 1\%$  Gaussian noise w.r.t. the coordinate range. All regions are sequentially removed and added again, moving their seed point (red dots) to the region’s intrinsic centroid. In this example this process is repeated only twice until the resulting segmentation  $l$  converges, i.e. remains unchanged after another iteration.

With the DijkstraFPS surface segmentation, its automatic MDL-based stopping criterion and Lloyd iterations for aligning the segmentation boundaries more precisely we have a useful tool for segmenting a meshed surface. It yields planar regions and is based on an intrinsic curvature-adaptive distance metric. In the following section we will briefly discuss alternative metrics that are either defined extrinsically or designed for non-planar surface regions.

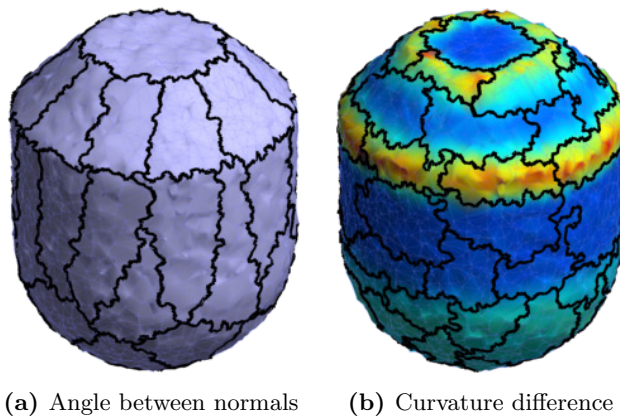
### 4.3 Advanced distance metrics

So far we used an intrinsic, curvature-adaptive distance measure  $d_u^v = \angle(\mathbf{n}_u, \mathbf{n}_v)$  for the DijkstraFPS surface segmentation. Therefore, we computed the angle between adjacent vertex normals and were able to successfully segment piece-wise planar surfaces. In this section we will, first, transfer this approach to objects with non-planar faces. Second, we discuss the possibility to incorporate extrinsic metrics into DijkstraFPS.

### 4.3.1 Non-planar regions

In Fig. 4.9 we see a synthetic mesh consisting of a horizontal plane, a truncated cone, a cylinder and a hemisphere (from top to bottom). A segmentation with our distance metric based on angles between adjacent normals yields boundaries at sharp edges, i.e. between plane and cone as well as between cone and cylinder. The smooth edge between cylinder and sphere, however, is not detected. Marching from cylinder to sphere does not lead to a sudden increase of the intrinsic distance. Thus, even though there are region boundaries around, they are not fixed to the truly underlying edge of the two object parts.

Computing differences of surface curvature  $d_u^v = |K_v - K_u|$  is an alternative to differences of surface normals. Fig. 4.9b shows the Gaussian curvature  $K$  computed from a certain neighborhood of each vertex. Since the Gaussian curvature of a noisy cylinder – in contrast to a sphere – is close to zero, DijkstraFPS successfully segments these two object parts using this new metric. The sharp edges, e.g., between cylinder and cone, however, are segmented as a separate region, since their local curvature is larger than in the area around.



**Figure 4.9:** Different curvature-adaptive metrics on a synthetically generated mesh with  $V = 5000$  vertices and  $\sigma = 0.5\%$  Gaussian noise w.r.t. the coordinate range. While considering angles between adjacent surface normals yields good segmentation results on top of this object, the smooth edge between cylinder and hemisphere is not correctly reconstructed. Considering differences of local curvature allows to improve the segmentation between these two object parts, while sharp edges on top introduce new edge regions.

Of course, the choice of using the Gaussian curvature and the size of the local neighborhood affects the distance metric and thus the final segmentation. A smaller neighborhood, e.g., shrinks the edge regions between cylinder and cone, but makes it harder to locally distinguish the sphere from a plane. Alternative measures for curvature differences exist, e.g. comparing curvature tensors (Seong et al., 2008) or covariance matrices (Belton and Lichti, 2006), which yield similar results.

The general problem of exploiting local surface features to derive a globally optimal segmentation remains. Therefore, we propose a variant of DijkstraFPS with extrinsically defined distances in the following section.

### 4.3.2 Extrinsic distance metrics

Until now we defined the metric to be used for the distance transform via local distances  $d_u^v$  between adjacent vertices  $\mathcal{V}_u$  and  $\mathcal{V}_v$ . Using the angle between surface normals allows to sum up the undulation along a path. As shown above this yields reasonable results. When segmenting a surface for reconstructing piece-wise planar objects, however, we might want to minimize distances to best fitting planes. In this section we demonstrate how to use such a criterion within our DijkstraFPS surface segmentation.

In contrast to the previously proposed distance metrics we do not define pair-wise distances  $d_u^v$  a priori, but on demand depending on the current segmentation. When evaluating the new distance  $D_v$  of a vertex  $\mathcal{V}_v$  neighboring to a vertex  $\mathcal{V}_u$  with label  $l_u$ , we add the squared distance to the best fitting plane of region  $l_u$ . Thus, the distance  $d_u^v$  indicates the increase of the sum of squared residuals if including this vertex into the current Voronoi region. Thus, it is also reasonable to sample the *farthest point* for refining the segmentation, since it is the one leading to the worst-fitting plane.

Fig. 4.10 shows resulting segmentations on a synthetic data set. As can be clearly seen, the extrinsic metric yields globally more consistent results with less curvy boundaries.

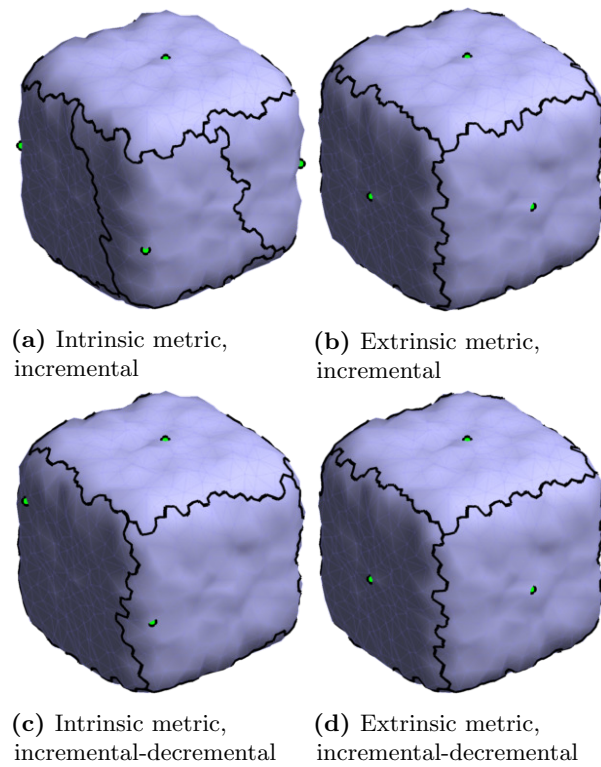
Although this approach yields promising results and might even support more complex surface parts like quadratic surfaces, its drawback is a significantly larger computing time. For each single distance  $d_u^v$  a plane has to be fit through vertices of the region  $l_u$ . This can be implemented more efficiently using incremental updates of quadric metrics (Garland et al., 2001), but is still computationally more expensive than pre-computing pair-wise distances once.

In the remainder of this thesis we will continue using intrinsic metrics. The incremental-decremental DijkstraFPS segmentation yields a sufficient approximation of the surface topology. We will improve the accuracy of the region boundaries in the following chapter using a primitive fitting approach.

Besides angles between normal vectors (4.4) we discussed two advanced distance metrics: curvature distances and extrinsic distances based on fitting planes. Curvature distances yield a segmentation with object edges segmented as separate regions, which is reasonable, but not particularly useful for our application. The extrinsic metric is very robust due to globally fitting primitives rather than computing distances locally; unfortunately this is at the expense of higher computational complexity.

Sections 4.1 and 4.2 introduced DijkstraFPS as a robust surface segmentation strategy. Although a combination with alternative metrics is possible (Section 4.3), it does not yield an efficient approach for segmenting surfaces with non-planar regions. Therefore, we will use DijkstraFPS for quickly pre-segmenting a surface into planar regions and proceed with a different strategy discussed in the following chapter.





**Figure 4.10:** DijkstraFPS with intrinsic and extrinsic distance metrics on an 6-norm sphere with  $V = 1000$  vertices and  $\sigma = 1\%$  Gaussian noise w.r.t. the coordinate range. In the top row the mesh is segmented using  $L = 6$  incremental DijkstraFPS iterations. In the bottom row the automatic incremental-decremental iteration scheme is applied. In both cases the extrinsic metric (right) yields better results than the intrinsic curvature-adaptive metric (left).



## CHAPTER 5

---

### Description length minimizing hierarchical face clustering

---

The segmentation from the previous chapter is based on a local, curvature-adaptive distance metric and is restricted to planar regions only. We will refine the segmentation using a hierarchical approach based on fitting primitives, which is called *hierarchical face clustering* (HFC, Garland et al., 2001), combined with the minimum description length (MDL) model selection strategy to obtain one superior surface segmentation algorithm. After briefly describing the HFC approach, we will introduce two kinds of operations, merging adjacent regions and diffusing boundary vertices, with the corresponding description length reduction. A decision strategy to efficiently derive an optimal segmentation concludes the MDL-based HFC surface segmentation. We extend the MDL formulation with a boundary regularization term and include more complex surface classes like quadratic and freeform surfaces. Finally, we combine both segmentation concepts: DijkstraFPS as a fast initialization and MDL-based HFC for its robustness and more complex surface types.

---

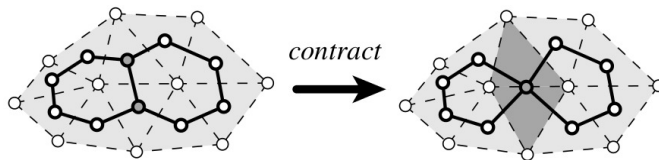
<b>5.1 Hierarchical face clustering (HFC)</b>	<b>80</b>
<b>5.2 Minimizing the description length</b>	<b>81</b>
5.2.1 Description length reduction when merging two regions	82
5.2.2 Description length reduction when diffusing vertices	82
5.2.3 Decision strategy	83
<b>5.3 Regularizing the boundary length</b>	<b>85</b>
<b>5.4 Different surface types</b>	<b>86</b>
5.4.1 Planar surfaces	86
5.4.2 Quadratic surfaces	86
5.4.3 Freeform surfaces	88
<b>5.5 Combination of MDL-based HFC and DijkstraFPS</b>	<b>89</b>

---

## 5.1 Hierarchical face clustering

Hierarchical face clustering (HFC, Garland et al., 2001) is a surface segmentation approach that merges adjacent regions based on their combined *fit error*. In this section we will briefly describe the original approach for piece-wise planar surfaces before augmenting it with MDL model selection and alternative surface types in the following sections.

Garland et al. (2001) start with all triangles separated, i.e. one region per triangle. Then they create a list of all pairs of adjacent regions. For each list entry they compute the fit error as the sum of squared residuals of all corresponding points w.r.t. the respective best fitting plane. Now they iteratively merge the two adjacent regions with the minimum fit error, until a user-defined number of regions is reached (Fig. 5.1, Alg. 5.1).



**Figure 5.1:** Merging adjacent triangles (Garland et al., 2001). The fit error for merging two adjacent faces is evaluated and the two faces with lowest fit error – here the two dark triangles – are merged. This operation is repeated until a user-defined number of faces is reached or the fit-error exceeds a predefined threshold. From Garland et al. (2001, Fig. 2).

---

**Algorithm 5.1:** Hierarchical face clustering (HFC, Garland et al., 2001). The clustering algorithm iteratively merges the pair of adjacent regions yielding the smallest fit error according to the sum of squared residuals w.r.t. the best fitting plane. It stops as soon as the predefined number of regions  $L$  is reached.

---

**Input:** triangles  $\{\mathcal{T}_t\}$ , vertices  $\{\mathcal{V}_v\}$ , number of regions  $L$

**Output:** labels  $l$  for all triangles

initialize labels  $l$  with one region per triangle:  $l_t \leftarrow t$  with  $t = 1, \dots, T$

collect pairs of adjacent regions, i.e. triangles  $(\mathcal{T}_t, \mathcal{T}_{t'})$  sharing two vertices

**for** each pair **do**

  | compute the fit error of a plane through all vertices involved by  $\mathcal{T}_t$  and  $\mathcal{T}_{t'}$

**end**

**while** number of regions  $L$  is not reached **do**

  | merge the pair of adjacent regions with the smallest fit error

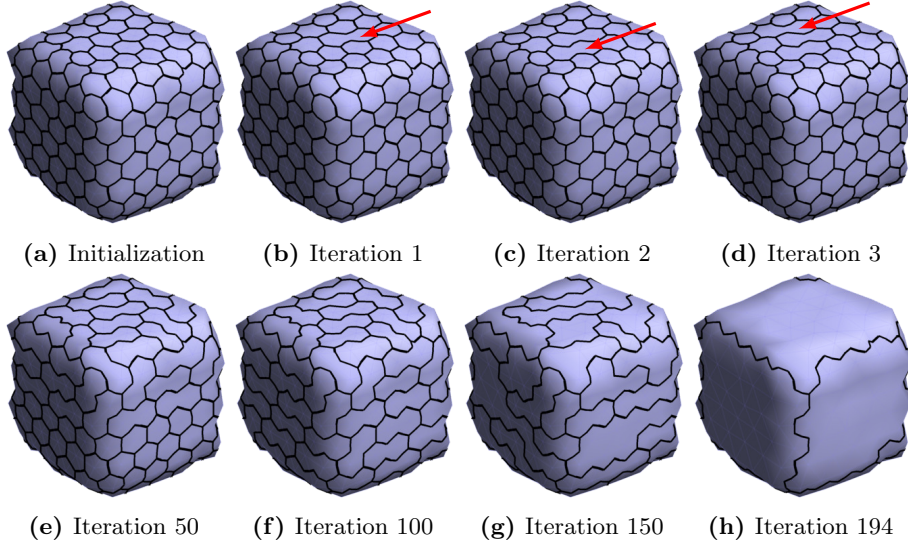
  | update the list of region pairs and corresponding fit errors

**end**

---

In contrast to Garland et al. (2001) our objective is to segment *vertices* rather than triangles. Therefore, we easily modify the original approach to start with one region per vertex and iteratively merge adjacent pairs of them. Fig. 5.2 demonstrates the HFC segmentation of a synthetic cube. Since the best fitting plane of any two vertices yields zero residuals, the first merge operations happen at random vertices (Fig. 5.2a–5.2d). The remaining process is shown for a few

iterations only, until it stops at a user-defined number of regions  $L = 6$  after 194 iterations (Fig. 5.2h).



**Figure 5.2:** Hierarchical face clustering on a synthetically generated 20-norm sphere with  $V = 200$  vertices and  $\sigma = 1$  % Gaussian noise w.r.t. the coordinate range. In contrast to Garland et al. (2001) we segment vertices rather than triangles. During the first iterations (a)–(d) random pairs of vertices are merged (red arrows), since two vertices always perfectly lie on a common plane. After a user-defined number of iterations, however, the segmentation represents the  $L = 6$  faces of a cube (h).

In the following section we will adopt this idea of hierarchically merging regions and introduce an error measure based on the description length  $\Phi_L$ . Besides the classical merge operation, we will allow to move boundary vertices from one region to another.

## 5.2 Minimizing the description length

As described in (2.60) and again in (4.6), the description length  $\Phi_L$  of a labeling  $l$  with  $L$  planes and a label  $l_v \in \{1, \dots, L\}$  for each vertex  $v$  is

$$\Phi_L = \underbrace{2V \text{lb} \frac{r}{\varepsilon}}_{\text{in-plane locations}} + \underbrace{\sum_{v=1}^V \left( \frac{1}{2 \ln 2} \cdot \frac{v_v^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right)}_{\text{off-plane residuals}} + \underbrace{3L \text{lb} \frac{r}{\varepsilon}}_{\text{plane parameters}} + \underbrace{V \text{lb} L}_{\text{labeling}}. \quad (5.1)$$

In the following we will analyze the description length reduction when merging two regions as shown in Fig. 5.2 and when moving boundary vertices from one region to another. Afterwards we discuss a fast and almost optimal decision strategy that combines both operations.

### 5.2.1 Description length reduction when merging two regions

As in the original HFC surface segmentation approach from Garland et al. (2001), the fundamental strategy of our MDL-based HFC is to iteratively merge adjacent regions. In contrast to selecting pairs of regions solely based on their joint fit error, we exploit the description length reduction  $\Delta\Phi_k$  induced by the respective merge operation.

When *merging* a pair  $k$  of adjacent regions we obtain a new description length  $\Phi_{k,L-1} = \Phi_L + \Delta\Phi_k$ . The difference  $\Delta\Phi_k$ , caused by the merge operation, is the difference of the previous description length  $\Phi_L$  and the resulting description length  $\Phi_{k,L-1}$ :

$$\Delta\Phi_k = \Phi_{k,L-1} - \Phi_L \quad (5.2)$$

$$\begin{aligned} &= 2V \text{lb} \frac{r}{\varepsilon} + \sum_{v=1}^V \left( \frac{1}{2 \ln 2} \cdot \frac{v_v'^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right) + 3(L-1) \text{lb} \frac{r}{\varepsilon} + V \text{lb}(L-1) \\ &- \left( 2V \text{lb} \frac{r}{\varepsilon} + \sum_{v=1}^V \left( \frac{1}{2 \ln 2} \cdot \frac{v_v^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right) + 3L \text{lb} \frac{r}{\varepsilon} + V \text{lb} L \right) \end{aligned} \quad (5.3)$$

$$= \sum_{v=1}^V \left( \frac{1}{2 \ln 2} \cdot \frac{v_v'^2 - v_v^2}{\sigma^2} \right) - 3 \text{lb} \frac{r}{\varepsilon} + V (\text{lb}(L-1) - \text{lb} L) \quad (5.4)$$

depending on the residuals  $\mathbf{v}$  before and  $\mathbf{v}'$  after the merge operation. Thus, less description length  $\text{lb}(L-1) - \text{lb} L < 0$  is needed to store the label of each vertex; three less plane parameters  $3 \text{lb} \frac{r}{\varepsilon}$  need to be stored; but the sum of squared residuals will increase depending on the geometry of the two merged regions.

In each iteration of the MDL-based HFC procedure we evaluate  $\Delta\Phi_k$  for each pair  $k$  of adjacent regions and merge the pair  $k^*$  with largest decrease of description length  $k^* = \text{argmin}_k \Delta\Phi_k$ . Furthermore, we automatically stop if no description length reduction is possible, i.e. if  $\Delta\Phi_{k^*} \geq 0$ . After each iteration the description length reduction needs to be re-computed only for pairs containing a region that has been affected by the previous merge operation.

So far the MDL-based selection strategy yields an automatic stopping criterion. In combination with another operation, i.e. moving or *diffusing* boundary vertices to an adjacent region, MDL allows us to automatically select the best type of operation as well.

### 5.2.2 Description length reduction when diffusing boundary vertices

A number of subsequent merge operations might yield large regions with stable parametrizations, but with tattered boundaries. Unfortunately, the boundaries cannot be straightened via merge operations. A possibility to improve the labeling  $\mathbf{l}$  and to reduce the overall description length is to *diffuse* a vertex from one side of the boundary to the other one.

We obtain the new description length  $\Phi_{v,L} = \Phi_L + \Delta\Phi_v$  with the description length reduction for diffusing a single vertex  $v$

$$\Delta\Phi_v = \Phi_{v,L} - \Phi_L \quad (5.5)$$

$$\begin{aligned} &= 2V \text{lb} \frac{r}{\varepsilon} + \sum_{v=1}^V \left( \frac{1}{2 \ln 2} \cdot \frac{v'_v{}^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right) + 3L \text{lb} \frac{r}{\varepsilon} + V \text{lb} L \\ &- \left( 2V \text{lb} \frac{r}{\varepsilon} + \sum_{v=1}^V \left( \frac{1}{2 \ln 2} \cdot \frac{v_v^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right) + 3L \text{lb} \frac{r}{\varepsilon} + V \text{lb} L \right) \end{aligned} \quad (5.6)$$

$$= \frac{1}{2 \ln 2} \cdot \frac{v'_v{}^2 - v_v^2}{\sigma^2} \quad (5.7)$$

only depending on the residual  $v_v$  before and the residual  $v'_v$  after the diffusion. Thus, whenever one vertex has smaller distance to the neighboring plane, we can diffuse it yielding a description length reduction.

Note that, to be precise, we would actually need to re-compute plane parameters for the new and old region of vertex  $v$ , which affects all residuals  $\mathbf{v}$  of both regions. The impact, however, is usually negligible. Thus, we assume the planes to be fixed and only consider the residuals of vertex  $v$  for evaluating  $\Delta\Phi_v$ .

Having formulated the description length reduction for merge and diffusion operations we could iteratively perform the best operation and re-compute all affected operations each time. This is, however, similarly computationally expensive like the extrinsic distance metric for the DijkstraFPS approach in Section 4.3.2. In the following section we will propose an effective strategy to speed up the process significantly, but obtaining almost identical results.

### 5.2.3 Decision strategy

In order to do operations with large description length reduction first and to postpone vague ones, we should evaluate all  $\Delta\Phi_k$  and  $\Delta\Phi_v$  and perform the best operation first. Afterwards we would re-evaluate all operations involving vertices affected by the first one and again perform the best operation. This strategy is optimal in the sense that it always performs the currently optimal operation, but is prohibitively expensive for large data sets. One diffused vertex  $v$  affects two regions. Thus, all merge operations with one of these regions and all diffusions from or to one of these regions need to be re-evaluated, involving the computation of one or two best fitting planes for each candidate operation.

A simplification and a huge time saving is the following strategy: We merge pairs of regions and re-evaluate only affected merge operations after each iteration, until there is no possible description length reduction anymore  $\Delta\Phi_{k^*} \geq 0$ . Then we compute residuals  $\mathbf{v}$  and  $\mathbf{v}'$  of all boundary vertices to the two adjacent planes and diffuse all vertices  $v$  with  $\Delta\Phi_v < 0$  at once. Afterwards we re-compute planes for all affected regions, re-evaluate all affected merge operations and try to merge regions again. As soon as there is no merge operation  $\Delta\Phi_{k^*} < 0$  and no diffusion  $\Delta\Phi_v < 0$ , the algorithm stops. This decision strategy is summarized in Alg. 5.2.

Fig. 5.3 shows the result of the MDL-based HFC for a synthetic data set with and without boundary vertex diffusion. In contrast to only merging adjacent regions, our combined merge and diffusion strategy yields better segmentation

---

**Algorithm 5.2:** MDL-based hierarchical face clustering. Pairs  $k$  of adjacent regions are merged and boundary vertices  $v$  diffused according to the induced description length reduction  $\Delta\Phi_k$  and  $\Delta\Phi_v$ , respectively. The clustering algorithm automatically stops when no further operations would reduce the total description length.

---

**Input:** triangles  $\{\mathcal{T}_t\}$ , vertices  $\{\mathcal{V}_v\}$ , noise variance  $\sigma^2$

**Output:** vertex labels  $l$

initialize labels  $l$  with one region per vertex  $l_v \leftarrow v$  with  $v = 1, \dots, V$

determine best fitting planes for each region  $l$

collect pairs of adjacent vertices, i.e. vertices  $(\mathcal{V}_u, \mathcal{V}_v)$  sharing a triangular edge

compute the description length reduction  $\Delta\Phi_k$  for each pair  $k$

**while** any description length reduction possible  $\min \Delta\Phi < 0$  **do**

    determine best pair of regions for merging  $k^* \leftarrow \operatorname{argmin}_k \Delta\Phi_k$

**if** merge operation would reduce the description length  $\Delta\Phi_{k^*} < 0$  **then**

        merge pair  $k^*$

        determine best fitting plane of the resulting region

**else**

        update description length reduction  $\Delta\Phi_v$  for boundary vertices  $v$

        diffuse all boundary vertices  $v$  with  $\Delta\Phi_v < 0$

        determine best fitting planes of affected regions

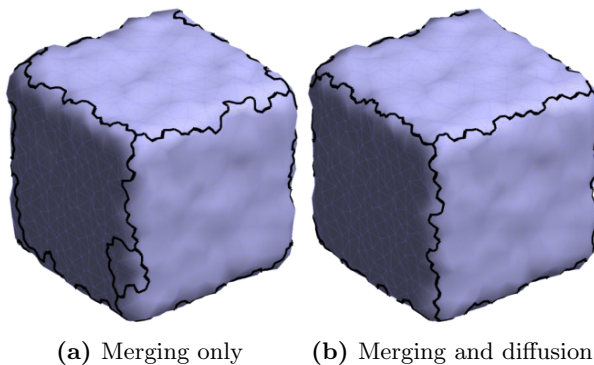
**end**

    update description length reduction  $\Delta\Phi_k$  for affected pairs  $k$

**end**

---

results: The vertex diffusion operations significantly straighten the segmentation boundaries.



**Figure 5.3:** MDL-based HFC with and without boundary vertex diffusion on a synthetically generated 20-norm sphere with  $V = 1000$  vertices and  $\sigma = 1\%$  Gaussian noise w.r.t. the coordinate range. In (a) we only allow merge operations and obtain a rather rough boundary and small regions that cannot be merged without increasing the description length. With boundary vertex diffusions (b) all boundary vertices are individually evaluated yielding a visually pleasing segmentation.

After describing the fundamental MDL-based HFC surface segmentation, we will discuss an alternative formulation for regularizing the boundary length. Later on we will introduce more complex surface types and apply the DijkstraFPS pre-segmentation from Chapter 4 as a fast initialization.



### 5.3 Regularizing the boundary length

Usually the MDL-based HFC, as described so far, already yields satisfactory results. On surfaces with obtuse-angled edges, however, we might obtain boundaries with a meandering pattern. Vertices far on one side of the boundary might have a smaller distance to the other plane by chance (Fig. 5.4a). By regularizing the boundary length we will suppress this behavior. Furthermore, we will provide a theoretical motivation for the proposed regularization term.

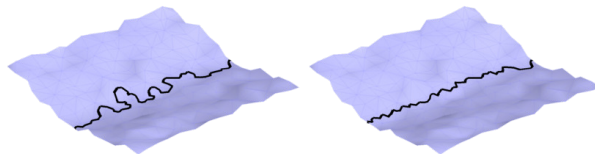
Although not necessary in terms of encoding the labeling  $l$ , we obtain good results when introducing an additional bit for each piece of boundary. This prevents huge boundary undulations at obtuse crease edges as in Fig. 5.4. I.e. whenever two neighboring vertices  $u$  and  $v$  have different labels  $l_u \neq l_v$ , we increment the description length by one bit.

Using exactly one bit per boundary piece is reasonable, since at each triangle the boundary has two possible directions to continue, leading to  $\text{lb } 2 = 1$  bit description length. Leclerc (1989, pg. 81) and Pan (1994, eq. 27) derive a similar description length of  $\approx \text{lb } 3$  bits per boundary pixel on a 2D image grid. Note that simply adding a bit per piece of boundary introduces some redundancy, since the term  $V \text{ lb } L$  already covers the complete labeling. With additional modifications, like a start vertex and the number of vertices per boundary (Pan, 1994), we might be able to drop the term  $V \text{ lb } L$  completely. In this work, however, we will treat the boundary bit as additional regularization and keep the other terms unchanged. In most cases the difference is marginal.

Thus, the regularized description length  $\Phi'_L$  is

$$\Phi'_L = 2V \text{ lb } \frac{r}{\varepsilon} + \sum_{v=1}^V \left( \frac{1}{2 \ln 2} \cdot \frac{v_v^2}{\sigma^2} + \text{lb } \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right) + 3L \text{ lb } \frac{r}{\varepsilon} + V \text{ lb } L + B \quad (5.8)$$

with the boundary length  $B$ . This definition affects both  $\Delta\Phi_k$  and  $\Delta\Phi_v$  with additional terms  $\Delta B_k$  in (5.4) and  $\Delta B_v$  in (5.7), respectively. The incremental boundary  $\Delta B_k$  for merging the region pair  $k$  is the number of triangular edges that connect vertices of merged regions. For computing the difference of the boundary length  $\Delta B_v$  when diffusing a vertex  $v$ , it is sufficient to evaluate the labels of the local neighbors  $\text{ne}(v)$ .



(a) Without boundary bits    (b) With boundary bits

**Figure 5.4:** MDL-based HFC with and without boundary regularization on a synthetically generated valley with  $V = 255$  vertices, a cutting angle of about  $157^\circ$  and  $\sigma = 1\%$  Gaussian noise w.r.t. the maximum coordinate range. Some vertices have smaller distance to the opposite plane, which is mainly caused by random noise and the obtuse intersection angle. In order to reduce their distances they are labeled with the closer region in terms of distances to the corresponding plane, regardless of the resulting boundary undulation (a). When penalizing each piece of boundary, its length is minimized at the small expense of larger vertex distances (b).

With the proposed regularization we obtain significantly straightened segmentation boundaries at obtuse edges between adjacent planes. This is even more relevant if considering non-planar surface regions with smooth edges, like a sphere adjacent to a cylinder with equal radius. We will introduce such surface types in the following section.

## 5.4 Different surface types

The concept of MDL allows to incorporate surface models of different complexity, i.e. with a different number of independent parameters. We will formulate the description length  $\Phi_L$  for  $L$  regions of arbitrary surface type, before discussing the HFC segmentation for different groups of surface types.

The description length  $\Phi_L$  for  $V$  vertices and  $L$  regions is

$$\Phi_L = 2V \text{lb} \frac{r}{\varepsilon} + \sum_{v=1}^V \left( \frac{1}{2 \ln 2} \cdot \frac{v_v^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right) + \sum_{l=1}^L \nu_{\hat{s}_l} \text{lb} \frac{r}{\varepsilon} + V \text{lb} L. \quad (5.9)$$

Depending on the surface class  $\hat{s}_l$  of a region  $l$ , the parameters  $\theta_l$  have  $\nu_{\hat{s}_l}$  degrees of freedom. Whenever re-computing surface parameters after merge or diffusion operations, we evaluate the description length for each surface class  $s_l$  and choose the class  $\hat{s}_l$  with minimum description length. The search over all classes  $s_l$  can be done independently for each surface  $l$ , thus the computational complexity only multiplies with the number of possible surface classes  $S$ .

### 5.4.1 Planar surfaces

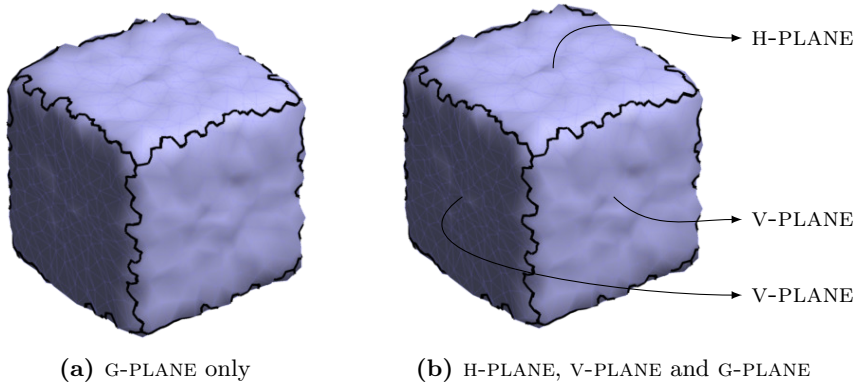
The simplest extension of our MDL-based HFC is to not only evaluate planes, but to distinguish the classes G-PLANE, H-PLANE and V-PLANE. If a plane can be parametrized as V-PLANE, one less degree of freedom needs to be stored. And if we only need to store the height of a H-PLANE, we save two degrees of freedom.

Choosing a more specific model usually results in larger residuals, since a more general plane approximates the corresponding points better. The description length considers both parts, residuals and model complexity, yielding a trade-off depending on the assumed point uncertainty  $\sigma$ . As indicated for a small example in Fig. 5.5 the minimum description length is achieved with a H-PLANE on top and bottom and a V-PLANE for each of the four vertical regions.

Consequently, we not only obtain a surface segmentation, but locally optimal surface classes as well. The difference of the resulting segmentation, however, is marginal compared to using G-PLANE only. For large data sets one might restrict oneself to G-PLANE in favor of lower computing times. Applying MDL-based HFC with different surface types becomes practically relevant with quadratic surfaces, as we will show in the following section.

### 5.4.2 Quadratic surfaces

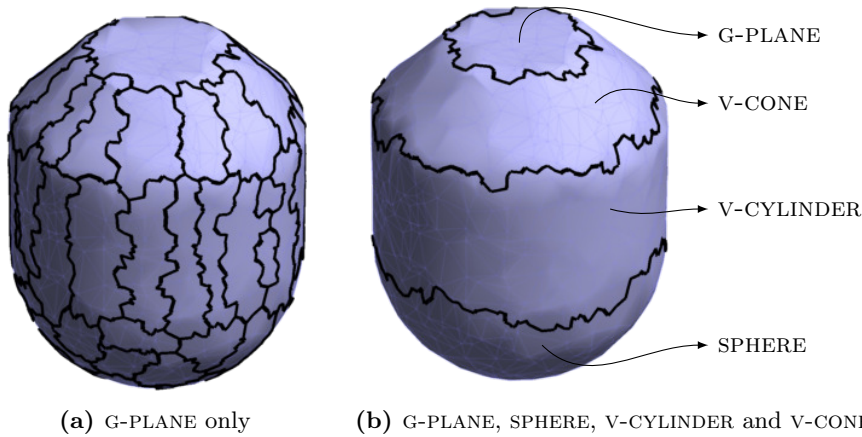
For objects that consist of piece-wise quadratic surfaces, also known as quadrics, we obtain a large benefit when not only using the G-PLANE surface class but also SPHERE, G-CYLINDER, V-CYLINDER and V-CONE. The implementation in the proposed MDL-based HFC is trivial.



**Figure 5.5:** MDL-based HFC with different classes of planes on a synthetically generated 20-sphere with  $V = 1000$  vertices and  $\sigma = 1\%$  Gaussian noise w.r.t. the coordinate range. While the MDL-based HFC already correctly segments the surface into planes in (a), it automatically chooses more specific primitive types like V-PLANE or H-PLANE in (b). The resulting segmentation not only contains a labeling  $l$ , but surface classes  $s_l$  for each region  $l$  as well.

As for G-PLANE, H-PLANE and V-PLANE we only need to implement routines for each primitive that yield parameters  $\theta_l$ , residuals  $\mathbf{v}$  and degrees of freedom  $\nu_{s_l}$  of a best fitting primitive of class  $s_l$ . Appendix A.1 lists direct solutions for determining parameters for all primitive types introduced in Tab. 3.1.

Fig. 5.6 shows the example of a synthetic object being segmented into the four parts it was constructed from. In contrast to the previous results in Fig. 4.9 obtained with DijkstraFPS, an oversegmentation is avoided. Instead of relying on local distances between adjacent vertices we now consider residuals to primitives fitted through many points.



**Figure 5.6:** MDL-based HFC with quadratic surfaces on a synthetically generated mesh with  $V = 1000$  vertices and  $\sigma = 0.1\%$  Gaussian noise w.r.t. the object diameter. In contrast to the DijkstraFPS surface segmentation (Chapter 4) the MDL-based HFC can explicitly model more complex surfaces like quadrics, which yields the very intuitive segmentation in (b), in contrast to the oversegmentation in (a).

With the current formulation we can treat all parametrized surfaces listed in Tab. 3.1. The following section will introduce one last type of surfaces relevant to this thesis, namely FREEFORM surfaces.

### 5.4.3 Freeform surfaces

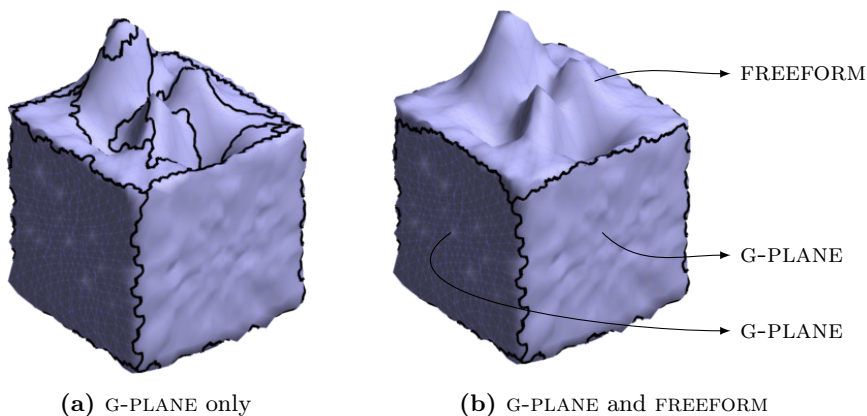
As described in Section 2.4.1, we can formulate the description length for a set of points even without any model. Therefore, it is possible to model FREEFORM surfaces, i.e. a parameterless surface represented by the original surface mesh.

All  $\bar{V}$  vertices on such a surface will be treated like outliers (Förstner, 1989), suggesting a description length of

$$\Phi_L = 2V \text{lb} \frac{r}{\varepsilon} + \bar{V} \text{lb} \frac{r}{\varepsilon} + \sum_v \left( \frac{1}{2 \ln 2} \cdot \frac{v_v^2}{\sigma^2} + \text{lb} \sqrt{2\pi} \frac{\sigma}{\varepsilon} \right) + \sum_{l=1}^L \nu_{\hat{s}_l} \text{lb} \frac{r}{\varepsilon} + V \text{lb} L \quad (5.10)$$

with  $\nu_{\text{FREEFORM}} = 0$  degrees of freedom for FREEFORM surfaces and the sum over residuals  $\sum_v$  only for inliers  $v$ . I.e. the off-plane location for a vertex on a FREEFORM surface is encoded like its in-plane location with  $\frac{r}{\varepsilon}$  bits per coordinate.

An example for a segmentation involving a FREEFORM surface is shown in Fig. 5.7. Since the area on top cannot be efficiently described with planes, it is automatically recognized as FREEFORM surface. Note that the MDL-based HFC approach always merges adjacent FREEFORM surfaces, since there are no residuals that could increase, but the boundary length is reduced.



**Figure 5.7:** MDL-based HFC with FREEFORM surfaces on a synthetically generated mesh with  $V \approx 2000$  vertices and  $\sigma = 1\%$  Gaussian noise w.r.t. the cube's edge length. It is possible to include surfaces without any parameters treating all their vertices as outliers. While the MDL-based HFC segmentation without FREEFORM surfaces tries to find an optimal segmentation into planar regions (a), it yields one large region on top when allowing FREEFORM surfaces (b).

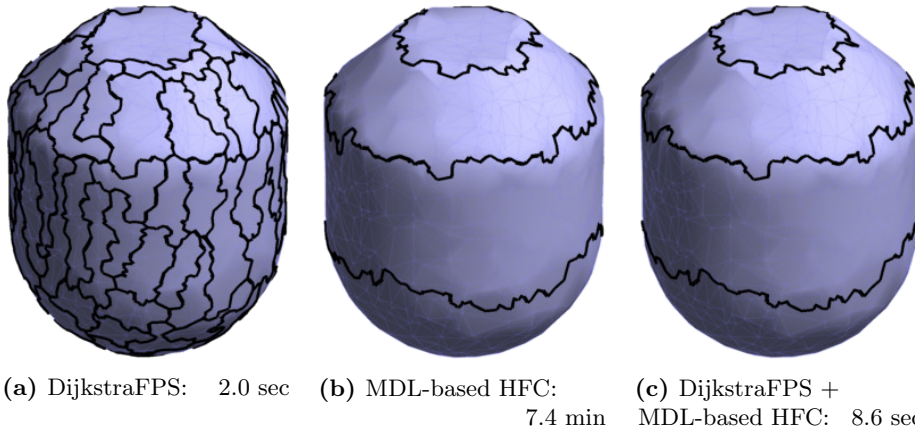
With the merge and diffusion operations and the variety of surface types we can precisely segment man-made surface structures, even with non-planar regions. The MDL model selection strategy does not only guide the different operations within the segmentation process, but yields an automatic stopping criterion as

well. The only drawback, the large computational costs in the beginning for aggregating vertices to small regions, make the approach intractable for large data sets. Therefore, we will use DijkstraFPS as an initialization in the following section.

## 5.5 Combination of MDL-based HFC and DijkstraFPS

In comparison to the previously discussed DijkstraFPS surface segmentation (Chapter 4) the MDL-based HFC segmentation is more flexible in terms of other surface classes than planes. But it is rather expensive to compute for large data sets, especially at the beginning when starting with a large number of regions  $L$  all containing one vertex only (Fig. 5.2). Therefore, we propose to combine the less powerful DijkstraFPS segmentation as an initialization with the MDL-based refinement.

As shown in Fig. 5.8a DijkstraFPS with its automatic stopping criterion yields a strong oversegmentation, since it assumes all regions to be describable with planes. The MDL-based HFC approach yields visually pleasing results, but takes several minutes and 1007 iterations, most time spent in the beginning merging tiny regions (Fig. 5.8b). The combination of DijkstraFPS and MDL-based HFC yields almost the same result, but is computed much faster with only 104 HFC iterations (Fig. 5.8c).



**Figure 5.8:** Combination of DijkstraFPS and MDL-based HFC on a synthetically generated mesh with  $V = 1000$  vertices and  $\sigma = 0.1$  % Gaussian noise w.r.t. the object diameter. DijkstraFPS is designed for piece-wise planar surfaces, thus yields 103 planar regions rather than one of the four quadratic surfaces (a). It is, however, well suited as an initialization for the MDL-based HFC segmentation (c), which itself is rather expensive to compute when initialized with one region per vertex (b).

This concludes the second step of our reconstruction framework for man-made surfaces. Together with the low-level pre-segmentation from the previous chapter we obtain a fast, accurate and flexible surface segmentation. Until now we only introduced model knowledge in form of available surface types. In the following

chapter about the high-level reconstruction step we will derive a model-driven classification as well as a constraint parameter estimation.

# CHAPTER 6

---

## Surface model reconstruction

---

In this chapter we will classify both the type of each surface region as well as each relation between adjacent pairs of regions. This global classification will take into account data-driven residuals of fitted primitives as well as model-driven parameter distributions and prior probabilities. E.g. an *almost* horizontal plane might be classified as H-PLANE if the slope is within a certain range, despite its normal vector significantly deviating from the vertical direction. Later we will make use of the determined surface and relation classes within a constraint parameter estimation. At the expense of slightly larger residuals we will introduce constraints to obtain an idealized, model-driven surface structure.

First we will represent the surface structure as a factor graph. After defining all factors involved, we will infer the most probable configuration and conclude the reconstruction with a constraint parameter estimation.

---

<b>6.1</b>	<b>Factor graph of surface structures</b>	<b>91</b>
<b>6.2</b>	<b>Unary factors</b>	<b>93</b>
6.2.1	Likelihood of points given the parametrization	94
6.2.2	Parameter distribution given a surface class	95
6.2.3	Surface priors	97
<b>6.3</b>	<b>Ternary factors</b>	<b>98</b>
6.3.1	Joint parameter distribution given a relation class	99
6.3.2	Relation priors	101
<b>6.4</b>	<b>Classification using the max-sum algorithm</b>	<b>101</b>
<b>6.5</b>	<b>Surface parameter estimation</b>	<b>103</b>

---

### 6.1 Factor graph of surface structures

The surface segmentation obtained in the previous chapters contains an underlying topological structure, i.e. a number of surface regions with pair-wise

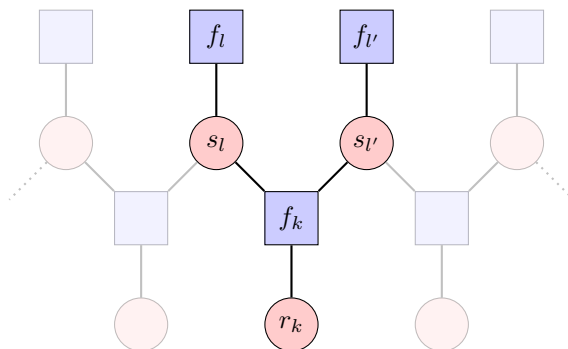
adjacencies, which we will explicitly represent using a special representation for graphical models, namely factor graphs, as discussed in Section 2.5. While variable nodes represent regions and inter-regional relations, factor nodes represent dependencies in between. This section will present the factor graph used for man-made surface structures and divide the joint probability into factors assigned to each factor node.

As shown in the factor graph in Fig. 6.1, we model two types of variables:

**Surface regions** Each surface region  $l$  is represented by one variable node. Possible surface classes are given in Tab. 3.1. The probability of a surface class being assigned to a region depends on the residuals of all corresponding points to the best fitting primitive of that type, a restricted parameter distribution and a prior probability.

**Relations** Each relation  $k$ , i.e. a pair of adjacent surface regions, is represented by a node. Possible relation classes are given in Tab. 3.2. The probability of a relation between two surface regions depends on the fulfillment of certain constraints and a prior probability.

Both variables, surfaces and relations, depend on unary factors  $f_l$  and ternary factors  $f_k$ .



**Figure 6.1:** Factor graph of adjacent surfaces  $l$  and  $l'$  with relation  $k$ . Variable nodes are indicated by red circles; factor nodes are depicted as blue squares. Unary factors  $f_l$  and  $f_{l'}$  as well as ternary factors  $f_k$  influence the joint probability  $P(\mathbf{s}, \mathbf{r})$  of one configuration of surfaces and relations  $(\mathbf{s}, \mathbf{r})$ . As indicated with the faded variable and factor nodes, the graph is usually larger than only two surfaces and one relation. In fact, its topology corresponds to the adjacency relations of all  $L$  surface regions.

The joint probability of one configuration with all surfaces  $\mathbf{s}$  and relations  $\mathbf{r}$  is proportional to the product over all factors. According to the factor graph defined in Fig. 6.1, we obtain

$$P(\mathbf{s}, \mathbf{r}) \propto \prod_{l=1 \dots L} f_l(s_l, \mathcal{X}_l) \cdot \prod_{k=1 \dots K} f_k(r_k, s_l, s_{l'}, \mathcal{X}_l, \mathcal{X}_{l'}) \quad (6.1)$$

with the surface region indices  $l = l(k)$  and  $l' = l'(k)$  in the second term depending on the current relation index  $k$ . On the one hand (6.1) is the product of factors for each surface  $s_l$  given the corresponding set of points  $\mathcal{X}_l$ . A set of surface classes  $\mathbf{s}$  that well approximate the underlying data points should



yield a large probability. On the other hand the joint probability depends on the product of factors for each relation  $r_k$  given the surface classes  $s_l$  and  $s_{l'}$  of the two adjacent surfaces and their points  $\mathcal{X}_l$  and  $\mathcal{X}_{l'}$ . We will choose the factors to approximate conditional probabilities with three components for the unary factors  $f_l$  and two components for the ternary factors  $f_k$ . Tab. 6.1 gives an overview of the individual terms.

**Table 6.1:** Components of unary and ternary factors. The factors depend on up to three different terms: the likelihood of the points given the surface parameters, the parameter distribution given the surface and relation class as well as prior probabilities.

	Unary factors $f_l$	Ternary factors $f_k$
Likelihood	F-testing variance factor $\hat{\sigma}_0^2$ of estimated primitive (Section 6.2.1)	—
Distribution	orientation $\theta_n$ , radius $\theta_r$ , opening angle $\theta_\alpha$ (Section 6.2.2)	angle $\delta$ between two related parameter vectors (Section 6.3.1)
Prior	prior probability of surface classes (Section 6.2.3)	prior probability of relations given two adjacent regions (Section 6.3.2)

In the following Sections 6.2 and 6.3 we will derive a formulation for both types of factors. Thereafter, in Section 6.4 we will search for a configuration of surface and relation classes

$$(\hat{\mathbf{s}}, \hat{\mathbf{r}}) = \underset{\mathbf{s}, \mathbf{r}}{\operatorname{argmax}} P(\mathbf{s}, \mathbf{r}) \quad (6.2)$$

that maximizes the joint probability  $P(\mathbf{s}, \mathbf{r})$ .

## 6.2 Unary factors

We propose factors  $f_l(s_l, \mathcal{X}_l)$  approximating the conditional probabilities  $P(s_l | \mathcal{X}_l)$  of a region  $l$  being describable with a primitive of class  $s_l$  given the points  $\mathcal{X}_l$  belonging to that region. So far we already determined the primitive that minimizes the description length for this region in Chapter 5. Now, however, we have to reconsider all other classes, since together with adjacent regions  $l'$  and relations  $k$  in between they can lead to a larger overall probability  $P(\mathbf{s}, \mathbf{r})$ .

In the following we will split the probability  $P(s_l | \mathcal{X}_l)$  into three terms: likelihood, parameter distribution and surface prior. Afterwards we will define factors for each of the three terms.

Bayes' theorem allows us to rewrite the posterior probability  $P(s_l | \mathcal{X}_l)$  with likelihood  $p(\mathcal{X}_l | s_l)$  and prior probability  $P(s_l)$ :

$$P(s_l | \mathcal{X}_l) \propto p(\mathcal{X}_l | s_l)P(s_l). \quad (6.3)$$

The likelihood  $p(\mathcal{X}_l | s_l)$  can be rewritten using the marginal likelihood

$$p(\mathcal{X}_l | s_l) = \int_{\boldsymbol{\theta}_l} p(\mathcal{X}_l | \boldsymbol{\theta}_l)p(\boldsymbol{\theta}_l | s_l) \mathrm{d}\boldsymbol{\theta}_l \quad (6.4)$$

and approximated following the argumentation of Minka (2001, pg. 44): “A practical substitute is to use a single value of [the parameters  $\theta_l$ ] which approximates the predictive distribution as well as possible.” We obtain

$$P(s_l | \mathcal{X}_l) \approx \frac{1}{Z} p(\mathcal{X}_l | \hat{\theta}_l) p(\hat{\theta}_l | s_l) P(s_l) \quad (6.5)$$

with the partition function  $Z$  and the maximum-likelihood estimation of the surface parameters  $\hat{\theta}_l$  given the points  $\mathcal{X}_l$  and the surface class  $s_l$ .

Thus, we define the factor  $f_l(s_l, \mathcal{X}_l)$  as product of three potentials

$$f_l(s_l, \mathcal{X}_l) := \underbrace{\phi_l(\mathcal{X}_l, \hat{\theta}_l)}_{\text{likelihood}} \underbrace{\phi_l(\hat{\theta}_l, s_l)}_{\text{distribution}} \underbrace{\phi_l(s_l)}_{\text{prior}}. \quad (6.6)$$

Using potentials  $\phi$  rather than probabilities  $P$  or probability densities  $p$  allows us to choose the normalization more flexibly, which will be advantageous for the second term in Section 6.2.2. We will formulate the three potentials in the following sections.

### 6.2.1 Likelihood of points given the parametrization

The first potential  $\phi_l(\mathcal{X}_l, \hat{\theta}_l) \propto p(\mathcal{X}_l | \hat{\theta}_l)$  in (6.6) represents the likelihood of the points  $\mathcal{X}_l$  given the estimated parameter vector  $\hat{\theta}_l$ . We will compute its value evaluating the estimated variance coefficient  $\hat{\sigma}_0^2$ .

Like during the MDL-based HFC in Chapter 5 we again obtain approximate solutions using the formulas in Appendix A.1. Now we subsequently perform a least-squares estimation of the surface parameters  $\theta$  in a Gauss-Helmert model with homogeneous parameter vectors (Section 2.3.2) in order to obtain a full covariance matrix  $\Sigma_{\theta\theta}$  and an estimated variance coefficient  $\hat{\sigma}_0^2$ .

The latter represents the inner consistency of the observed points  $\mathcal{X}_l$  with the underlying model  $\theta_l$ , i.e. the fitted primitive of type  $s_l$ . Under the null hypothesis “The empirical variance coefficient is equal to the theoretical variance coefficient”  $H_0 : E(\hat{\sigma}_0^2) = \sigma_0^2$  the quotient  $\hat{\sigma}_0^2/\sigma_0^2$  follows a Fisher distribution:

$$\left. \frac{\hat{\sigma}_0^2}{\sigma_0^2} \right| H_0 \sim F(R, \infty). \quad (6.7)$$

The redundancy  $R = G - U$  of the estimation  $\hat{\sigma}_0$  depends on the number of constraints  $G$  and the number of unknown parameters  $U$ . The apriori variance coefficient is  $\sigma_0^2 = 1$  throughout this thesis. Theoretically its redundancy is  $\infty$ . Practically, however, this is too large to reflect realistic prior information. Instead, the apriori variance  $\sigma_0^2$  is uncertain as well with a prespecified variance  $\sigma_{\sigma_0}^2$  corresponding to the redundancy (Koch, 1997, p. 259)

$$R_0 = \left\lceil \frac{1}{2\sigma_{\sigma_0}^2} \right\rceil. \quad (6.8)$$

Therefore, we choose the test statistic

$$q = \left. \frac{\hat{\sigma}_0^2}{\sigma_0^2} \right| H_0 \sim F(R, R_0). \quad (6.9)$$

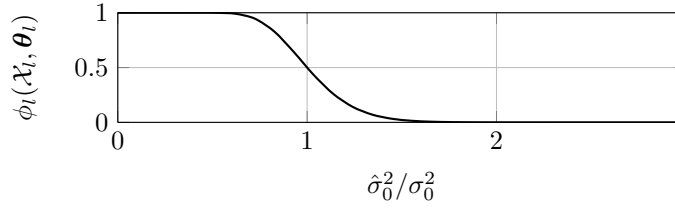
For the potential  $\phi_l(\mathcal{X}_l, \hat{\boldsymbol{\theta}}_l)$  representing the likelihood  $p(\mathcal{X}_l | \hat{\boldsymbol{\theta}}_l)$  we choose

$$\phi_l(\mathcal{X}_l, \hat{\boldsymbol{\theta}}_l) := 1 - F(q, R, R_0) \quad (6.10)$$

using the cumulative Fisher distribution function  $F$ , as by definition

$$P(q | H_0) = F(q, R, R_0). \quad (6.11)$$

Fig. 6.2 shows the potential  $\phi_l(\mathcal{X}_l, \hat{\boldsymbol{\theta}}_l)$  depending on the test statistic  $\hat{\sigma}_0^2/\sigma_0^2$ .



**Figure 6.2:** Potential  $\phi_l(\mathcal{X}_l, \boldsymbol{\theta}_l)$  representing the likelihood of the data points  $\mathcal{X}_l$  of region  $l$  depending on the estimated variance coefficient  $\hat{\sigma}_0^2$ . This plot shows the potential for redundancies  $R = R_0 = 100$ .

Outliers are usually removed during the iterative parameter estimation process. In order to consider the number of detected outliers for computing the likelihood, we count them like inliers but with residuals set to  $3\sigma$ . This way the likelihood slightly decreases with each outlier.

The likelihood for FREEFORM surfaces will always be 1, since they yield no residuals. In order to avoid all surfaces being classified as such FREEFORM surface, we will assign them a low prior probability in Section 6.2.3.

This data-driven likelihood term represents the proximity of the estimated surface primitive w.r.t. the corresponding points. The second term presented in the following section will control the parameter distribution to avoid degenerated primitives and to support less complex surface types.

### 6.2.2 Parameter distribution given a surface class

The second potential  $\phi_l(\hat{\boldsymbol{\theta}}_l, s_l) \propto p(\hat{\boldsymbol{\theta}}_l | s_l)$  in (6.6) represents the parameter distribution conditioned on the surface class  $s_l$ . We define this distribution to suppress degenerated surfaces such as a horizontal or vertical G-PLANE, a flat sphere or cylinder due to huge radii and a cylinder-shaped or flat cone due to an opening angle of  $0^\circ$  or  $180^\circ$ . Therefore, we divide the parameter space into acceptance and rejection intervals smoothed with the estimated variance of the surface parameters. After describing the chosen parameter distribution for each of these three degenerated scenarios we will discuss how to proceed if none or multiple of them apply.

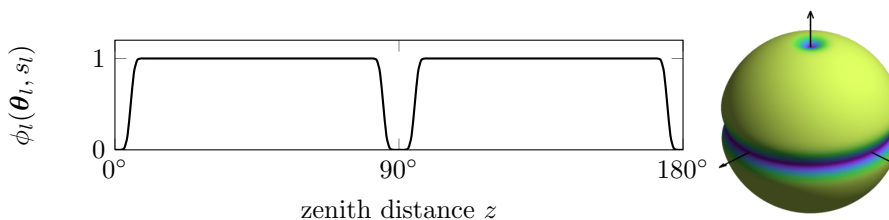
**Orientation of general primitives.** A G-PLANE will always approximate the points better than a H-PLANE or V-PLANE, although its normal vector  $\boldsymbol{\theta}_n$  might be very close to horizontal or vertical. The same argumentation holds for

a G-CYLINDER. In order to penalize normal vectors  $\theta_n$  close to the vertical axis or close to the horizontal plane, we define the potential as

$$\phi_l(\theta_l, s_l) := \Phi\left(\frac{\min(z, |90^\circ - z|, 180^\circ - z) - t_z}{\sigma_z}\right) \quad (6.12)$$

for  $s_l \in \{\text{G-PLANE, G-CYLINDER}\}$

with the zenith distance  $z = \arccos(\theta_{n_z})$  and the cumulative distribution function of the standard normal distribution  $\Phi(x) = \int_x \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$ . Note that this restrictive parameter distribution should only be applied if the more specialized primitives, e.g. H-PLANE and V-PLANE, are within the set of available primitives. Fig. 6.3 visualizes the distribution of the zenith distance  $z$  and the resulting density function of the normal direction exaggerated in height over the unit sphere.



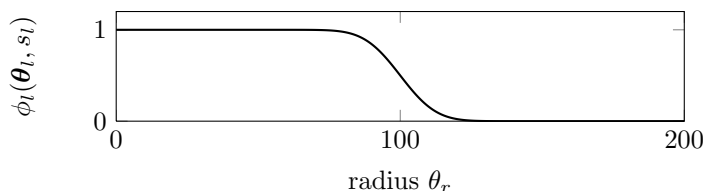
**Figure 6.3:** Parameter distribution  $p(z | s_l) \propto \phi_l(\theta_l, s_l)$  for the zenith distance  $z$  for surfaces  $s_l \in \{\text{G-PLANE, G-CYLINDER}\}$  and resulting distribution of the normal direction  $\theta_n$  with an angular threshold  $t_z = 5^\circ$  and a variance  $\sigma_z = 1^\circ$ .

**Radius of spheres and cylinders.** A SPHERE or G-CYLINDER will always approximate the points better than a G-PLANE, even though its radius  $\theta_r$  might be huge. We support radii smaller than a threshold  $t_r$  and thus define the distribution potential

$$\phi_l(\theta_l, s_l) := 1 - \Phi\left(\frac{\theta_r - t_r}{\sigma_r}\right) \quad (6.13)$$

for  $s_l \in \{\text{V-CYLINDER, G-CYLINDER, SPHERE}\}$ .

Fig. 6.4 visualizes the distribution of the radius  $\theta_r$ .



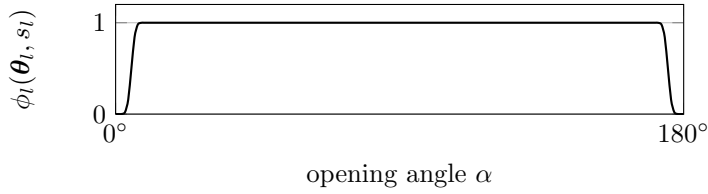
**Figure 6.4:** Parameter distribution  $p(\theta_r | s_l) \propto \phi_l(\theta_l, s_l)$  for the radius  $\theta_r$  of surfaces  $s_l \in \{\text{V-CYLINDER, G-CYLINDER, SPHERE}\}$  with a threshold  $t_r = 100$  and a variance  $\sigma_r = 10$ .

**Opening angle of cones.** A V-CONE with an opening angle  $\alpha$  close to  $180^\circ$  will always approximate planar points slightly better than a H-PLANE. Similarly, cylindrically arranged points will always be better approximated by a V-CONE with an opening angle of  $\alpha \approx 0^\circ$  than by a V-CYLINDER. Thus, we restrict the opening angle  $\alpha = \alpha(\theta_A) = 2 \arctan(\sqrt{\theta_A})$  to be greater than a threshold  $t_\alpha$  and smaller than  $180^\circ - t_\alpha$  by defining the distribution potential for as

$$\phi_l(\boldsymbol{\theta}_l, s_l) := \Phi\left(\frac{\min(\alpha, 180^\circ - \alpha) - t_\alpha}{\sigma_\alpha}\right) \quad (6.14)$$

for  $s_l = \text{V-CONE}$ .

Fig. 6.5 visualizes the distribution of the opening angle  $\alpha$ .



**Figure 6.5:** Parameter distribution  $p(\alpha | s_l) \propto \phi_l(\boldsymbol{\theta}_l, s_l)$  for the opening angle  $\alpha = \alpha(\theta_A)$  of surfaces  $s_l = \text{V-CONE}$  with an angular threshold  $t_\alpha = 5^\circ$  and a variance  $\sigma_\alpha = 1^\circ$ .

**Normalization of mixed and default potentials.** For surface classes affected by more than one distribution, like the G-CYLINDER with a distribution potential for both the normal vector and the radius, we simply multiply both potentials. For surface classes not mentioned above we set the potential to one:

$$\phi_l(\boldsymbol{\theta}_l | s_l) := 1. \quad (6.15)$$

This way all potentials are normalized such that for parameters within the acceptance interval the potential is one, thus does not affect the probability  $P(s_l | \mathcal{X}_l)$ .

This section introduced several control parameters with clear semantics, which are necessary to describe the assumed surface model. The range for, e.g., normal vectors  $t_z$  might be differently chosen for high-precision engine parts than for mediievally building facades.

One last term remains for computing unary factors. In case we know about the frequency of certain surface classes, e.g. learned from training data sets, we can incorporate prior probabilities.

### 6.2.3 Surface priors

The last term  $\phi_l(s_l) \propto P(s_l)$  in (6.6) allows to support or to suppress certain surface classes. The prior could represent the model complexity as in MDL-based approaches or can be learned by analyzing large data sets with a large number of labeled surface regions.

Within our experiments we equally weighted all surface classes but the FREEFORM class by defining the prior potential as

$$\phi_l(s_l) := \begin{cases} 0.01 & \text{if } s_l = \text{FREEFORM} \\ 1 & \text{else} \end{cases}. \quad (6.16)$$

The low prior probability for FREEFORM surfaces will prevent the classification from only yielding FREEFORM surfaces, although they have likelihood and distribution potentials of one due to the lack of residuals and parameters.

With likelihood, parameter distribution and prior probability, we have formulated all three components of the unary surface factors  $f_l$  defined in (6.6). Similarly, the following section will formulate the two components of the ternary relation factors  $f_k$ .

### 6.3 Ternary factors

In analogy to the unary factors for surfaces (6.3) we can formulate the ternary factors  $f_k(r_k, s_l, s_{l'}, \mathcal{X}_l, \mathcal{X}_{l'})$  approximating the conditional probability  $P(r_k | s_l, s_{l'}, \mathcal{X}_l, \mathcal{X}_{l'})$  for a relation  $k$  being of relation class  $r_k$  given the surface classes  $s_l$  and  $s_{l'}$  of the two adjacent regions  $l$  and  $l'$  and the corresponding points  $\mathcal{X}_l$  and  $\mathcal{X}_{l'}$ . Rewriting the conditional probability using Bayes' theorem we obtain

$$P(r_k | s_l, s_{l'}, \mathcal{X}_l, \mathcal{X}_{l'}) = \frac{p(\mathcal{X}_l, \mathcal{X}_{l'} | r_k, s_l, s_{l'})P(r_k | s_l, s_{l'})}{p(\mathcal{X}_l, \mathcal{X}_{l'} | s_l, s_{l'})}. \quad (6.17)$$

We rewrite the likelihoods  $p(\mathcal{X}_l, \mathcal{X}_{l'} | r_k, s_l, s_{l'})$  and  $p(\mathcal{X}_l, \mathcal{X}_{l'} | s_l, s_{l'})$  using the marginal likelihood and obtain

$$P(r_k | s_l, s_{l'}, \mathcal{X}_l, \mathcal{X}_{l'}) = \frac{\int_{\boldsymbol{\theta}_{l'}} \int_{\boldsymbol{\theta}_l} p(\mathcal{X}_l, \mathcal{X}_{l'} | \boldsymbol{\theta}_l, \boldsymbol{\theta}_{l'})p(\boldsymbol{\theta}_l, \boldsymbol{\theta}_{l'} | r_k, s_l, s_{l'}) d\boldsymbol{\theta}_l d\boldsymbol{\theta}_{l'} P(r_k | s_l, s_{l'})}{\int_{\boldsymbol{\theta}_{l'}} \int_{\boldsymbol{\theta}_l} p(\mathcal{X}_l, \mathcal{X}_{l'} | \boldsymbol{\theta}_l, \boldsymbol{\theta}_{l'})p(\boldsymbol{\theta}_l, \boldsymbol{\theta}_{l'} | s_l, s_{l'}) d\boldsymbol{\theta}_l d\boldsymbol{\theta}_{l'}} \quad (6.18)$$

with the maximum-likelihood estimations  $\hat{\boldsymbol{\theta}}_l$  and  $\hat{\boldsymbol{\theta}}_{l'}$  given the points  $\mathcal{X}_l$  and  $\mathcal{X}_{l'}$  and the classes  $s_l$ ,  $s_{l'}$  and  $r_k$ . The integrals are approximated following the very same argumentation of Minka (2001, pg. 44):

$$P(r_k | s_l, s_{l'}, \mathcal{X}_l, \mathcal{X}_{l'}) \approx \frac{p(\mathcal{X}_l, \mathcal{X}_{l'} | \hat{\boldsymbol{\theta}}_l, \hat{\boldsymbol{\theta}}_{l'})p(\hat{\boldsymbol{\theta}}_l, \hat{\boldsymbol{\theta}}_{l'} | r_k, s_l, s_{l'})P(r_k | s_l, s_{l'})}{p(\mathcal{X}_l, \mathcal{X}_{l'} | \hat{\boldsymbol{\theta}}_l, \hat{\boldsymbol{\theta}}_{l'})p(\hat{\boldsymbol{\theta}}_l, \hat{\boldsymbol{\theta}}_{l'} | s_l, s_{l'})}. \quad (6.19)$$

Canceling out equal terms in the nominator and denominator yields

$$P(r_k | s_l, s_{l'}, \mathcal{X}_l, \mathcal{X}_{l'}) \approx \frac{p(\hat{\boldsymbol{\theta}}_l, \hat{\boldsymbol{\theta}}_{l'} | r_k, s_l, s_{l'})}{p(\hat{\boldsymbol{\theta}}_l, \hat{\boldsymbol{\theta}}_{l'} | s_l, s_{l'})} P(r_k | s_l, s_{l'}). \quad (6.20)$$

Although the quotient still depends on the surface classes  $s_l$  and  $s_{l'}$ , we define the factor  $f_k(r_k, s_l, s_{l'}, \mathcal{X}_l, \mathcal{X}_{l'})$  as product of the two following potentials

$$f_k(r_k, s_l, s_{l'}, \mathcal{X}_l, \mathcal{X}_{l'}) := \underbrace{\phi_k(\hat{\boldsymbol{\theta}}_l, \hat{\boldsymbol{\theta}}_{l'}, r_k)}_{\text{distribution}} \underbrace{\phi_k(r_k, s_l, s_{l'})}_{\text{prior}}. \quad (6.21)$$

The first potential only depends on the surface parameters  $\theta_l$  and  $\theta_{l'}$  and their relation  $r_k$ . The second one does not depend on any data or parameters, but solely reflects the prior probability of a relation  $r_k$  given the surface classes  $s_l$  and  $s_{l'}$ . We will formulate the two individual terms in the following sections.

### 6.3.1 Joint parameter distribution given a relation class

Most relations listed in Tab. 3.2 involve a constraint on both adjacent surfaces. The constraint is either an orthogonality  $\perp$ , a parallelism  $\parallel$  or an equality  $=$ . Similarly to the distributions in Section 6.3.1 we would like to specify acceptance and rejection intervals for a contradiction measure  $\delta$  smoothed with its estimated variance  $\sigma_\delta$ , yielding the first potential  $\phi_k(\hat{\theta}_l, \hat{\theta}_{l'}, r_k)$  of the ternary factor (6.21). Some relations, e.g. the identity of two G-PLANE surfaces, involve both angular and translatory differences. After discussing a strict combination of both of them into one contradiction measure  $\delta$ , we will propose an approximation and demonstrate its validity under certain conditions.

Angular and translatory differences  $\angle(\theta_n, \theta'_n)$  and  $|\theta_d - \theta'_d|$  can not be added or multiplied directly. A correct combination would be to, e.g., normalize w.r.t. the uncertainties  $\sigma_\angle$  and  $\sigma_d$ , yielding the Mahalanobis distance (Mahalanobis, 1936)

$$\delta = \sqrt{\left(\frac{\angle(\theta_n, \theta'_n)}{\sigma_\angle}\right)^2 + \left(\frac{\theta_d - \theta'_d}{\sigma_d}\right)^2}. \quad (6.22)$$

Alternatively, we can take the Mahalanobis distance of the reduced parameter vectors  $\theta_r$  and  $\theta'_r$  (Förstner, 2012)

$$\delta = \sqrt{(\theta_r - \theta'_r)^\top (\Sigma_{\theta_r, \theta_r} + \Sigma_{\theta'_r, \theta'_r})^{-1} (\theta_r - \theta'_r)}. \quad (6.23)$$

Since both alternatives yield a dimensionless contradiction  $\delta$ , the choice of a suitable threshold  $t_\delta$  is non-intuitive. Furthermore, this approach would require different contradiction measures for equalities, parallelisms and orthogonalities as well as for the combination with different primitive types.

Therefore, we propose a uniform approach for all three types of constraints based on the angle between homogeneous parameter vectors. As we will demonstrate shortly, this angle is a reasonable approximation for the case of conditioned data, i.e. primitives in a maximum coordinate range of  $[-1, 1]$ . The proposed approach simplifies the implementation of additional relation and surface classes and yields a single intuitive sensitivity threshold  $t_\delta$  for inter-surface relations.

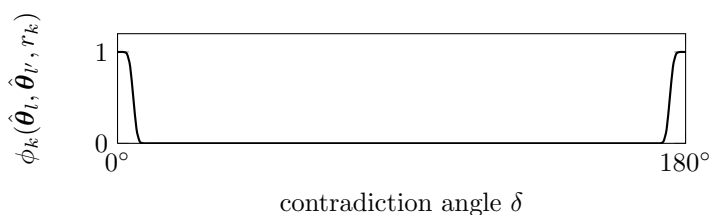
Augmenting the two vectors of an equality with a homogeneous part  $\theta := \begin{bmatrix} \theta \\ 1 \end{bmatrix}$  the equality relation can be expressed as parallelism, which relates to an angle  $\angle(\theta, \theta') \in \{0^\circ, 180^\circ\}$ . Since orthogonality yields a very similar equation  $\angle(\theta, \theta') = 90^\circ$ , all constraints mentioned in Tab. 3.2 lead to a contradiction angle  $\delta = \angle(\theta, \theta')$  or  $\delta = 90^\circ - \angle(\theta, \theta')$  with its corresponding variance  $\sigma_\delta$ , obtained via error propagation of the covariance matrices  $\Sigma_{\theta_l, \theta_l}$  and  $\Sigma_{\theta_{l'}, \theta_{l'}}$  obtained from the least-squares estimation in Section 6.2.

Similarly to angular thresholds in Sections 6.2.2 we now introduce a threshold  $t_\delta$  for relations between adjacent surfaces. We define the joint distribution potential as

$$\phi_k(\hat{\boldsymbol{\theta}}_l, \hat{\boldsymbol{\theta}}_{l'}, r_k) := 1 - \Phi\left(\frac{\min(\delta, 180^\circ - \delta) - t_\delta}{\sigma_\delta}\right) \quad (6.24)$$

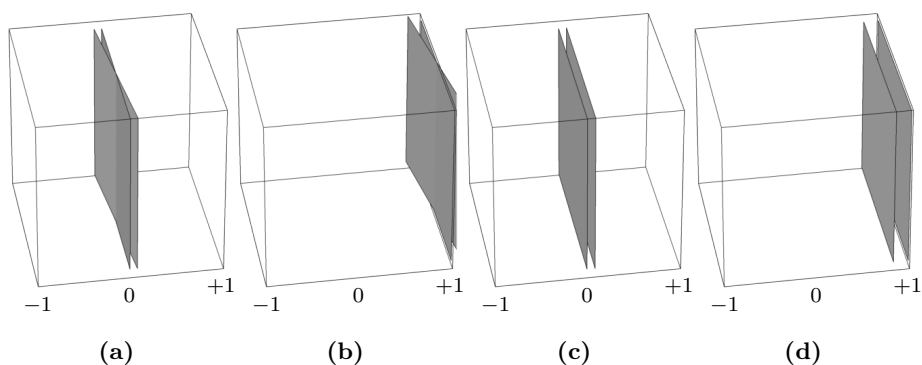
for  $r_k \neq \text{ARBIT}$ .

For relations that do not induce a constraint we define the potential to be  $\phi_k(\hat{\boldsymbol{\theta}}_l, \hat{\boldsymbol{\theta}}_{l'}, r_k) := 1$ . Fig. 6.6 visualizes the distribution of the contradiction angle  $\delta$ .



**Figure 6.6:** Parameter distribution  $p(\delta|r_k) \propto \phi_k(\boldsymbol{\theta}_l, \boldsymbol{\theta}_{l'}, r_k)$  for relations  $r_k \neq \text{ARBIT}$  inducing a contradiction angle  $\delta = \delta(\boldsymbol{\theta}, \boldsymbol{\theta}')$  between two adjacent surface regions with a threshold  $t_\delta = 5^\circ$  and a variance  $\sigma_\delta = 1^\circ$ .

Fig. 6.7 shows different pairs of primitives all with a contradiction angle  $\delta = 5^\circ$  for the IDENT relation. Under the assumption of a conditioned surface mesh, i.e. being translated and scaled to the unit bounding box in a preprocessing step, the single threshold  $t_\delta$  is a reasonable choice to keep the number of control parameters small. Either way, the above-mentioned alternative formulations are possible as well. The specific choice of the contradiction  $\delta$  does not affect the overall reconstruction framework.



**Figure 6.7:** Impact of rotation and translation on the contradiction angle  $\delta$ . In each of the four examples we synthetically generate a pair of v-PLANE surfaces. While they are placed close to the origin, i.e.  $\boldsymbol{\theta} = [1, 0, 0]^T$ , in (a) and (c), they are in the margins of the unit bounding box, i.e.  $\boldsymbol{\theta} = [1, 0, 1]^T$ , in (b) and (d). One plane of each pair is either rotated by  $5.0^\circ$  (a) or  $7.1^\circ$  (b) or translated by 0.088 (c) or 0.161 (d). All four transformations yield a contradiction angle of  $\delta = 5^\circ$ .



After defining this term based on the parameter vectors of two adjacent surface regions, one last term remains. It will allow us to incorporate prior probabilities for relation classes, possibly learned from training data sets. More importantly we will use it to suppress topologically impossible or unlikely combinations of adjacent surface types and their corresponding relation.

### 6.3.2 Relation priors

The second term  $\phi_k(r_k, s_l, s_{l'}) \propto P(r_k \mid s_l, s_{l'})$  of the ternary factor (6.21) represents the probability of a relation  $r_k$  given the two adjacent surface classes  $s_l$  and  $s_{l'}$ . Therefore, it holds both the prior probability of the relation itself as well as the probability of the whole combination involving all three variables. We will make use of both aspects.

All allowed combinations of adjacent surface classes  $s_l$  and  $s_{l'}$  as well as their relation  $r_k$  are listed in Tab. 3.2. We assign equal prior probability to all these combinations. All other combinations have zero prior probability.

Furthermore, we use the prior term to suppress the non-restrictive relation ARBIT. Otherwise its probability will always be largest, since there is no restriction for the parameters  $\theta_l$  and  $\theta_{l'}$  induced by the ARBIT relation.

This leads to the prior term

$$\phi_k(r_k, s_l, s_{l'}) := \begin{cases} \begin{cases} 0.01 & \text{if } r_k = \text{ARBIT} \\ 1 & \text{else} \end{cases} & \text{if } (r_k, s_l, s_{l'}) \text{ in Tab. 3.2} \\ 0 & \text{else} \end{cases} . \quad (6.25)$$

As indicated above, this term can be learned from labeled training data sets.

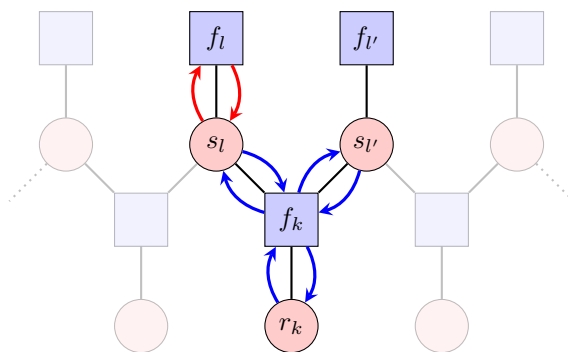
Together with the joint parameter distribution described in the previous section, this prior term concludes the ternary factor  $f_k$ . It will make the crucial difference between a local and a global classification, since it links the classification of adjacent surfaces.

Now we have collected all necessary terms to compute the unary and ternary factors in the factor graph (Fig. 6.1). We will proceed with solving for an optimal configuration  $(\hat{\mathbf{s}}, \hat{\mathbf{r}})$  of surface and relation classes, before determining surface parameters in a global, constraint parameter estimation.

## 6.4 Classification using the max-sum algorithm

We are interested in the most likely configuration of surfaces and relations, maximizing the joint probability  $P(\mathbf{s}, \mathbf{r})$ . As described in Section 2.5 we infer an approximation using the max-sum algorithm with loopy belief propagation (Frey and MacKay, 1998), which is based on exchanging messages between nodes of the factor graph.

Our graphical model with all messages involved in the belief propagation is shown in Fig. 6.8. Note that each factor  $f'_k$  has exactly three neighboring variable nodes,  $s_l$ ,  $s_{l'}$  and  $r_k$ , but a surface variable  $s_l$  can have multiple neighboring factors  $f_k$ , one for each adjacent surface region.



**Figure 6.8:** Factor graph with all messages involved. The factor graph from Fig. 6.1 is split into two groups of edges: The two red arrows between factor  $f_l$  and surface  $s_l$  indicate messages that need to be updated for each surface  $l$ . The six blue arrows from and to factor  $f_k$  represent messages for each relation  $k$ . This way each message is processed exactly once with a loop over  $l$  and a loop over  $k$ .

Although the messages between  $s_l$  and  $f_k$  as well as between  $s_{l'}$  and  $f_k$  are symmetrical, it is convenient to derive and to implement update formulas for both of them. This way we have exactly two messages for each region  $l$  (red arrows in Fig. 6.8) and six messages for each relation  $k$  (blue arrows in Fig. 6.8).

Since our graph has no chain or tree structure, the max-sum algorithm, as described in Section 2.5, needs to be iteratively called until it converges to the desired solution. The messages are initialized according to (2.82) and (2.83) and updated after a predefined message passing schedule with formulas (2.80) and (2.81). For the bipartite factor graph an intuitive schedule is to update all messages from factor to variable nodes first, followed by updates for messages from variable to factor nodes. We further group these messages according to the type of factor involved, either a surface factor  $f_l$  or a relation factor  $f_k$ . For the sake of readability, we will omit the arguments  $\mathcal{X}_l$  and  $\mathcal{X}_{l'}$  for the factors  $f_l$  and  $f_k$ .

**Message updates for each surface region  $l$ .** First we define the two messages involved with each surface region  $l$  represented with red arrows in Fig. 6.8. The message sent from a surface factor  $f_l$  is

$$\mu_{f_l \rightarrow s_l}(s_l) = \log f_l(s_l) \quad (6.26)$$

with the unary factors  $f_l$  from (6.6). The message back is

$$\mu_{s_l \rightarrow f_l}(s_l) = \sum_{f_k \in \text{ne}(s_l)} \mu_{f_k \rightarrow s_l}(s_l) \quad (6.27)$$

with the sum over all ternary nodes  $f_k \in \text{ne}(s_l)$  depending on the surface variable  $s_l$ .

**Message updates for each relation  $k$ .** Now we define the six messages involved with each relation  $k$  between adjacent surfaces  $l$  and  $l'$  represented with blue arrows in Fig. 6.8. The three messages sent from  $f_k$  are

$$\mu_{f_k \rightarrow s_l}(s_l) = \max_{s_{l'}, r_k} (\log f_k(r_k, s_l, s_{l'}) + \mu_{s_{l'} \rightarrow f_k}(s_{l'}) + \mu_{r_k \rightarrow f_k}(r_k)) , \quad (6.28)$$

$$\mu_{f_k \rightarrow s_{l'}}(s_{l'}) = \max_{s_l, r_k} (\log f_k(r_k, s_l, s_{l'}) + \mu_{s_l \rightarrow f_k}(s_l) + \mu_{r_k \rightarrow f_k}(r_k)) , \quad (6.29)$$

$$\mu_{f_k \rightarrow r_k}(r_k) = \max_{s_l, s_{l'}} (\log f_k(r_k, s_l, s_{l'}) + \mu_{s_l \rightarrow f_k}(s_l) + \mu_{s_{l'} \rightarrow f_k}(s_{l'})) \quad (6.30)$$

with the ternary factors  $f_k$  from (6.21). The messages back to  $f_k$  are

$$\mu_{s_l \rightarrow f_k}(s_l) = \mu_{f_{s_l} \rightarrow s_l}(s_l) + \sum_{f_{k'} \in \text{ne}(s_l) \setminus f_k} \mu_{f_{k'} \rightarrow s_l}(s_l) , \quad (6.31)$$

$$\mu_{s_{l'} \rightarrow f_k}(s_{l'}) = \mu_{f_{s_{l'}} \rightarrow s_{l'}}(s_{l'}) + \sum_{f_{k'} \in \text{ne}(s_{l'}) \setminus f_k} \mu_{f_{k'} \rightarrow s_{l'}}(s_{l'}) , \quad (6.32)$$

$$\mu_{r_k \rightarrow f_k}(r_k) = 0 \quad (6.33)$$

with sums over all other ternary factor nodes  $f_{k'} \in \text{ne}(s_l) \setminus f_k$  depending on the surface variable  $s_l$ . The message from  $r_k$  is an empty sum, thus this message is always zero.

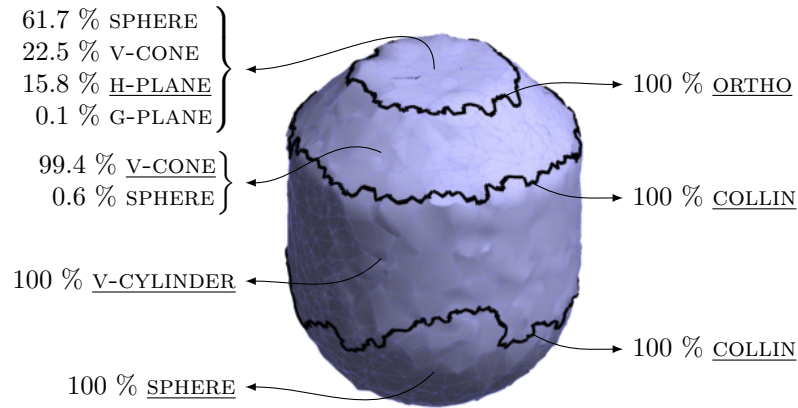
For all experiments within this thesis we use the Probabilistic Graphical Model Library from Andres et al. (2008). Results of the max-sum inference on a small test object are given in Fig. 6.9. Although many unary surface factors already point to the correct surface class before performing belief propagation, the classification for the two surfaces on top is significantly improved. While the planar surface is most likely a SPHERE according to its unary factor, it is correctly classified as H-PLANE after inference.

This classification represents the most probable configuration of the observed surface structure given the original surface mesh as well as model knowledge about man-made surfaces. Using the inferred surface classes and inter-surface constraints induced by the inferred relation classes we will estimate an idealized surface model in the following section.

## 6.5 Surface parameter estimation

In Section 6.2.1 we individually estimated surface parameters  $\theta_l$  for all surface regions  $l$  and for all possible surface types  $s_l$ . The classification in Section 6.4 yields the most probable configuration of surface types  $\hat{\mathbf{s}}$ , which allows us to select an optimal parametrization  $\theta_l$  for each surface  $l$  among all possible types  $s_l$ . Additionally the classification yields corresponding relation types  $r_k$  for all pairs of adjacent surface regions  $(l, l')$ , which possibly induce constraints (Tab. 3.2). Starting from the locally estimated parameters  $\theta_l$  we now need to find a global parametrization that fulfills those constraints.

We formulate this estimation problem in terms of the model with constraints between observations only (Section 2.3.4). The previously estimated surface parameters  $\theta_l$  and their covariance matrices  $\Sigma_{\theta\theta}$  are treated as observations  $\mathbf{y}$  with constraints  $\mathbf{g}(\mathbf{y})$ . As described in Section 2.3.3 we need to reduce

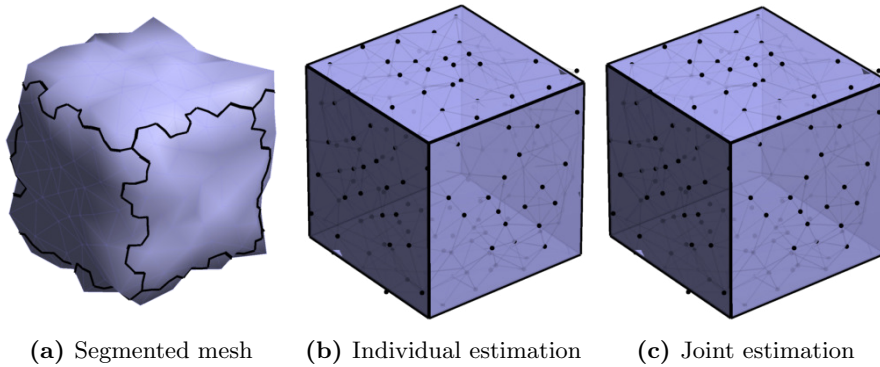


**Figure 6.9:** Results of the max-sum inference on a synthetically generated mesh with  $V = 2000$  vertices and  $\sigma = 0.5\%$  Gaussian noise w.r.t. the object diameter. Each surface  $l$  and relation  $r$  is annotated with possible classes, i.e. classes with non-zero probability. The numbers for surface classes  $s_l$  on the left-hand side represent unary factors  $f_l$  normalized to  $\sum_{s_l} f_l(s_l) = 100\%$ . The numbers on the right-hand side represent marginal probabilities  $P(r_k)$  for relation classes  $r_k$ . Especially the small region on top is not yet perfectly recognized as H-PLANE. Also the region below has a small chance for being recognized as a SPHERE instead of V-CONE. The underlined class names are the classification results after inference with the max-sum algorithm. Thus, all surfaces and relations are correctly classified.

homogeneous parameter vectors to their null space with the transformation matrix  $J$  given in Tab. 3.1, in order to avoid singularities when inverting their covariance matrices.

Fig. 6.10 shows the example of a small synthetic cube being segmented, classified and reconstructed using the described parameter estimation with constrained observations. Although visually poorly recognizable, the surfaces' locations improve significantly when introducing pair-wise constraints and performing a joint estimation. The intersection angles between the four vertical planes are off by up to  $4.7^\circ$  w.r.t. the right angle. The joint estimation yields perfect right angles. The other eight intersection angles are already  $90^\circ$  due to the involved surface types H-PLANE and V-PLANE, which only allow the relation ORTHO, but do not require an additional constraint.

Note that the constraints inferred in the previous section could contain redundancies or logical contradictions. E.g., the four vertical planes of a cube lead to four orthogonalities, while one of them is redundant due to the fixed sum of interior angles. On the other hand, a large number of cylindrically aligned planes might yield IDENT relations only, which is topologically impossible without collapsing the cylindrical shape to a single plane. The consistency and redundancy of inter-regional constraints is, however, out of scope of this thesis. The interested reader is kindly referred to the literature on topological constraints in polyhedral object models, like e.g. the approach from Loch-Dehbi and Plümer (2009, 2011) based on symbolic geometric reasoning with so-called *Wu's method*. An alternative approach for detecting contradictions would be to introduce soft



**Figure 6.10:** Joint parameter estimation with constraints on a synthetically generated cube with  $V = 200$  vertices and  $\sigma = 5\%$  Gaussian noise w.r.t. the coordinate range. Although the original synthetic mesh is of very poor accuracy and low density, the surfaces and relations are correctly classified as H-PLANE and V-PLANE being ORTHO. Visually the joint estimation with constraints has no big impact. But the pair-wise intersection angles are significantly improved from  $86.3^\circ$  to  $94.7^\circ$  before and exactly  $90.0^\circ$  after the joint estimation.

constraints and to downweight constraints with large residuals, similarly to the outlier re-weighting scheme described in Section 2.3.5.

The classification and constraint estimation of an idealized surface model concludes the third and final step of the proposed reconstruction framework for man-made surfaces. The following chapter will analyze its performance on various synthetic and real-world data sets.



# CHAPTER 7

---

## Empirical evaluation

---

The previous chapters presented the individual processing steps of the proposed reconstruction framework. We will now demonstrate its application to various types of triangulated point clouds. After a general demonstration on a small example mesh we analyze the performance w.r.t. different criteria specified in Section 3.1.4 on synthetically generated meshes. Later we investigate how well these findings transfer to real data sets.

---

<b>7.1</b>	<b>Demonstration of the complete algorithm . . . . .</b>	<b>108</b>
<b>7.2</b>	<b>Experiments with synthetic data. . . . .</b>	<b>110</b>
7.2.1	Accuracy of surface parameters . . . . .	111
7.2.2	Sensitivity w.r.t. various data properties . . . . .	112
7.2.2.1	Noise. . . . .	112
7.2.2.2	Outliers. . . . .	113
7.2.2.3	Model violation . . . . .	113
7.2.2.4	Sampling density . . . . .	114
7.2.3	Sensitivity w.r.t. model assumptions. . . . .	115
7.2.3.1	Control parameters. . . . .	116
7.2.3.2	Set of surface classes . . . . .	116
7.2.4	Computational complexity . . . . .	118
<b>7.3</b>	<b>Experiments with real data . . . . .</b>	<b>120</b>
7.3.1	Accuracy analysis on calibration objects . . . . .	120
7.3.2	Flexibility w.r.t. different acquisition techniques . . . . .	121
7.3.2.1	CAD meshes . . . . .	121
7.3.2.2	Multi-view stereo point clouds. . . . .	124
7.3.2.3	Structured-light reconstruction . . . . .	127
7.3.2.4	Terrestrial laser scan . . . . .	128
7.3.2.5	Laser scanning arm . . . . .	131
7.3.3	Sensitivity w.r.t. model assumptions. . . . .	131
7.3.4	Computational complexity . . . . .	134

---

## 7.1 Demonstration of the complete algorithm

We start with a demonstration of all three processing steps on a visually reconstructed desktop scene, before analyzing different aspects of the proposed reconstruction framework. This involves the point cloud reconstruction and triangulation, DijkstraFPS pre-segmentation and MDL-based HFC refinement as well as surface classification and constraint parameter estimation.

For this demonstration as well as throughout the remainder of this chapter – when not otherwise indicated – we use the following setup of control parameters (Tab. 7.1): The primitive normals with zenith angle of less than  $t_z = 1^\circ$  are considered being vertical, cones need to have an opening angle of at least  $t_\alpha = 5^\circ$  and radii need to be smaller than  $t_r = 100$  times the maximum coordinate range. The angular tolerance for adjacent parameters is  $t_\delta = 1^\circ$ . All surfaces types have a uniform prior, except for FREEFORM surfaces and ARBIT relations with a prior which is 100 times smaller:  $P(\text{FREEFORM}) = P(\text{ARBIT}) = 0.01 \cdot P(\text{others})$ . Furthermore, to obtain more robust results, we usually restrict the set of surface primitives to the ones actually being present in the given surface structure. In the following example, which obviously consists of piece-wise planar surfaces, we only allow for different types of planes.

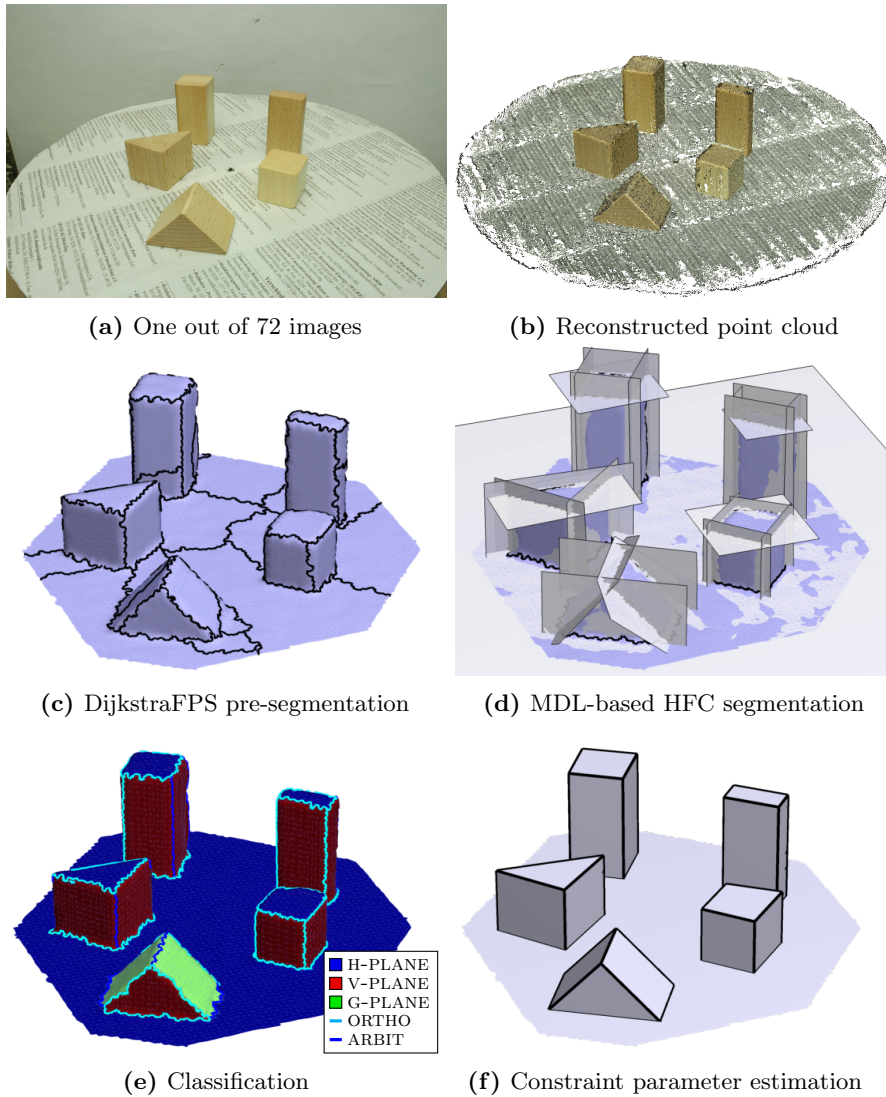
**Table 7.1:** Default values for model parameters. When not otherwise indicated, the control and model parameters of the proposed reconstruction framework are set to the default values given in this table. The set of surface types is usually specified depending on the expected surface structure, i.e. planar regions only or with quadrics or FREEFORM surfaces as well.

Parameter	Symbol	Default value
Tolerance for zenith distances	$t_z$	$1^\circ$
Minimum opening angle	$t_\alpha$	$5^\circ$
Maximum radius	$t_r$	$100 \times \text{max. coordinate range}$
Tolerance for angular deviation	$t_\delta$	$1^\circ$
Surface priors	$P(s_l)$	$\propto \begin{cases} 0.01 & \text{for } s_l = \text{FREEFORM} \\ 1 & \text{else} \end{cases}$
Relation priors	$P(r_k)$	$\propto \begin{cases} 0.01 & \text{for } r_k = \text{ARBIT} \\ 1 & \text{else} \end{cases}$
Set of surfaces types		depending on data set

Fig. 7.1 shows intermediate steps and the final result of the proposed reconstruction framework for a small desktop scene with five BRICKS arranged on a horizontal turntable. The 3D point cloud with  $V = 6713$  vertices (Fig. 7.1b) was computed from 72 images using the structure-from-motion system Bundler (Snavely et al., 2006), the dense patch-based multi-view stereo reconstruction PMVS2 (Furukawa and Ponce, 2010) and the Poisson surface reconstruction (Kazhdan et al., 2006) to obtain a triangular mesh. Note that sharp edges tend to get smoothed due to this specific surface reconstruction.

We assume a coordinate uncertainty of  $\sigma = 0.15\%$  w.r.t. the maximum coordinate range. As verified at the end of the reconstruction procedure, this value yields estimated variance coefficients close to one  $\hat{\sigma}_0^2 \approx 1$  for most surface





**Figure 7.1:** Reconstruction of the multi-view stereo mesh BRICKS with  $V = 6713$  vertices. From 72 images (a) we reconstruct a dense 3D point cloud (b) with a structure-from-motion approach and a patch-based multi-view stereo reconstruction system. After deriving a triangulated mesh, we apply the DijkstraFPS pre-segmentation (c) yielding quite some oversegmentation. The MDL-based HFC refinement merges several pairs of adjacent regions and slightly moves the segmentation boundaries to better approximate the object edges (d). The HFC is based on fitting primitives, which are shown as gray plane segments. A classification (e) and constraint parameter estimation (f) yield the final surface structure augmented with surface and relation classes as well as accurate surface parameters.

regions, thus well represents the noise contained in the data. We apply our DijkstraFPS surface segmentation with its automatic stopping criterion as described in Chapter 4. After obtaining a minimum description length at 74 regions during the incremental segmentation step, the number of regions is reduced to 43 during the decremental segmentation step (Fig. 7.1c). The whole process takes about 6.9 seconds. Although all surface parts might be describable with planes, the surface remains oversegmented and cannot be further simplified within the DijkstraFPS scheme.

A subsequent segmentation using the MDL-based HFC (Chapter 5) is better suited at this mid-level reconstruction stage. Its optimization involves coordinate residuals as well as model complexity and allows to merge adjacent regions and to diffuse boundary vertices. It yields the intuitive segmentation with 24 regions and its corresponding surface primitives:  $5 \times 3 = 15$  for the cuboids,  $4 \times 2 = 8$  for the triangular prisms and 1 for the ground plane (Fig. 7.1d). Due to the initialization with the DijkstraFPS result, the HFC is performed in 1.1 seconds only.

The global classification (Section 6.4) yields surfaces and relations depicted in Fig. 7.1e. While the used max-sum algorithm itself converges after 14 iterations, the whole classification takes about 2.2 seconds and spends most computing time for robustly estimating parameters for each region and each possible surface class. All surface classes are correctly determined, i.e. all regions are classified as H-PLANE (5 regions), V-PLANE (17 regions) or G-PLANE (2 regions), respectively. The relations, however, only partly correspond to the intuitive expectation. While relations between H-PLANE and V-PLANE are only allowed to be ORTHO (Tab. 3.2), some relations between two V-PLANE regions – also practically orthogonal – are classified as ARBIT, e.g. the blue vertical edge of the left triangular prism or the ridge edge of the other one. A narrow inspection unveils: Due to smoothed edges and bumpy faces caused by the initial surface triangulation the segmentation is not always accurate and the reconstructed cutting angles overly deviate from  $90^\circ$ .

Many detected relations – up to, e.g., ARBIT – act as constraints in a final parameter estimation (Section 6.5). The resulting surface primitives, obtained after 0.6 seconds and clipped at adjacent primitives, are shown in Fig. 7.1f. Sometimes erroneous classification results are compensated by the object’s redundancy as for the long cuboid on the right: While one edge is supposed to be ARBIT, the other three right angles enforce the forth one to be  $90^\circ$  as well. In other cases, like for the small cube, only two edges are recognized as ORTHO, thus one V-PLANE is slightly rotated.

After demonstrating the overall procedure on one example data set, we will focus on more detailed aspects in the following sections. First we will restrict ourselves to synthetic surface meshes, before applying the proposed framework to other real-world data sets.

## 7.2 Experiments with synthetic data

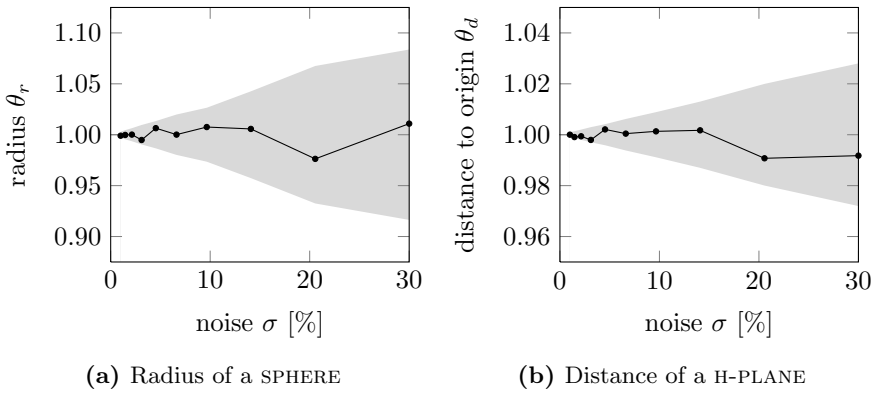
In this section we will investigate the performance of the proposed reconstruction framework on synthetically generated surface meshes. We take up again the criteria stated in Section 3.1.4 and analyze them individually in terms of correctness of the resulting surface structure.

### 7.2.1 Accuracy of surface parameters

In this experiment we analyze the accuracy of reconstructed surface parameters. We use synthetic data in order to be able to compare the estimation with ground truth information.

We generate two meshes, a SPHERE and a H-PLANE, with  $V = 1000$  vertices each and add synthetic Gaussian noise with increasing variance. After applying the proposed reconstruction framework we obtain a segmentation, which is trivial for these examples, and estimated surface parameters. Fig. 7.2 shows the estimated radius  $\theta_r$  and distance to the origin  $\theta_d$ , respectively. As indicated by the shaded  $3\hat{\sigma}$  confidence interval, the estimation does not significantly deviate from the ground truth values at  $\tilde{\theta}_r = 1$  and  $\tilde{\theta}_d = 1$ . Furthermore, the empirical accuracy of, e.g.,  $\hat{\sigma}_{\theta_d} = 0.01 = 1\%$  of the radius at a noise level of  $\sigma = 30\%$  w.r.t. the radius confirms the theoretical expectation, since error propagation for the distance to the origin  $\theta_d = \frac{1}{V} \sum_v X_{v,z}$  (A.5) yields

$$\sigma_{\theta_d} = \sqrt{\frac{1}{V} \sum_v \left( \underbrace{\frac{\partial \theta_d}{\partial X_{v,z}}}_{=1} \right)^2} \sigma^2 = \frac{\sigma}{\sqrt{V}} = \frac{30\%}{\sqrt{1000}} \approx 1\%. \quad (7.1)$$



**Figure 7.2:** Reconstruction accuracy of synthetically generated meshes with  $V = 1000$  vertices each and increasing Gaussian noise given relatively w.r.t. to the object radius. The black curve represents the estimated radius  $\theta_r$  (a) and the estimated distance to the origin  $\theta_d$  (b). The shaded area is the  $3\hat{\sigma}$  confidence interval using the empirical variances  $\hat{\sigma}_{\theta_r}^2$  and  $\hat{\sigma}_{\theta_d}^2$ , respectively. In both cases the estimation does not significantly deviate from the true values indicated by the thin horizontal line and confirms the expected accuracy obtained via error propagation.

We could, of course, analyze the reconstruction for many more types of surface primitives. We will, however, restrict ourselves to these two examples, since they confirm our expectations and the results are easily transferable to other scenarios. In Section 7.3.1 we will perform another accuracy analysis to verify the results on real data sets.

## 7.2.2 Sensitivity w.r.t. various data properties

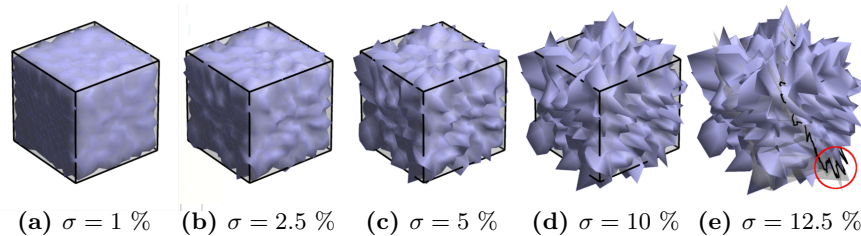
The following section investigates the sensitivity of the proposed reconstruction framework w.r.t. properties of the input data. This involves the presence of noise and outliers, systematic model violations as well as poor sampling density.

For simplicity we restrict the reconstruction to planar primitives throughout this section. Furthermore, in order to focus on the segmentation, classification and parameter estimation, we assume the correct triangulation to be given. This is usually not the case for real applications, especially in the presence of large amounts of noise and outliers, as in the following experiments.

### 7.2.2.1 Noise

In this experiment we want to determine the sensitivity to noise on a small example data set. All three steps of the reconstruction framework are designed for noisy surface meshes. At a certain signal-to-noise ratio, however, the reconstruction will fail.

Fig. 7.3 shows the example mesh used in this experiment. The points are disturbed with Gaussian noise of increasing variance. The reconstructed surface structure is indicated by a black wireframe. With 10 % noise w.r.t. the coordinate range the cube is still correctly reconstructed. Only at a noise level of 12.5 % the surface model collapses to the topologically impossible configuration of two arbitrarily related planes, which causes an invalid wireframe rendering in Fig. 7.3e.



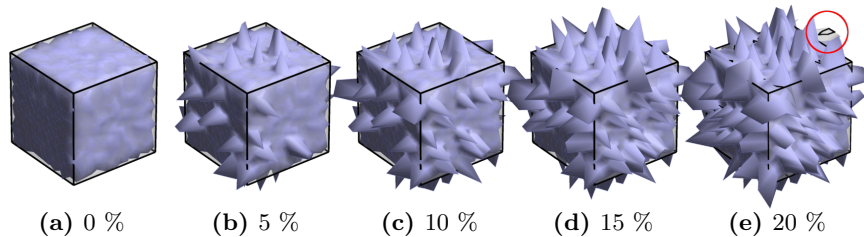
**Figure 7.3:** Sensitivity analysis w.r.t. noise on a synthetically generated cube with  $V = 1000$  vertices and increasing Gaussian noise. The surface structure, shown as black wireframe, is correctly reconstructed for noise levels up to  $\sigma = 10\%$  w.r.t. the coordinate range (d), until the reconstruction fails at  $\sigma \geq 12.5\%$  (e). Despite the simplicity of this example, it proves robustness of the proposed reconstruction framework to quite some random noise.

The maximum amount of noise, here 10 %, depends on many factors like the complexity of the surface structure or the number of vertices  $V$ . For, e.g.,  $V = 100$ , the reconstruction fails for  $\sigma \geq 10\%$ . With  $V = 10000$  vertices it only fails at  $\sigma \geq 12.5\%$ .

Note that the mesh is generated on the original noise-free point cloud. With 10 % or 12.5 % Gaussian noise the surface triangulation would certainly fail. Although this example is a highly simplified scenario, it clearly demonstrates robustness to noisy point coordinates.

### 7.2.2.2 Outliers

Besides small random noise we need to analyze random outliers and their impact on the reconstruction process. Therefore, we use the very same synthetic surface mesh from Fig. 7.3 and perturb it with an increasing amount of outliers, i.e. points with large deviations, 25 % of the coordinate range in our case (Fig. 7.4). Only with 20 % outliers the reconstruction yields an oversegmentation of  $L = 12$  regions with a topologically impossible classification and parameter estimation, best visible as an artefact on the top right edge.



**Figure 7.4:** Sensitivity analysis w.r.t. outliers on a synthetically generated cube with  $V = 1000$  vertices,  $\sigma = 1$  % Gaussian noise w.r.t. the coordinate range and an increasing percentage of outliers. The outliers are randomly sampled and perturbed by a shift of 25 % into normal direction. The reconstructed surface model is shown as black wireframe. With an amount of 15 % outliers the surface structure is still correctly recognized.

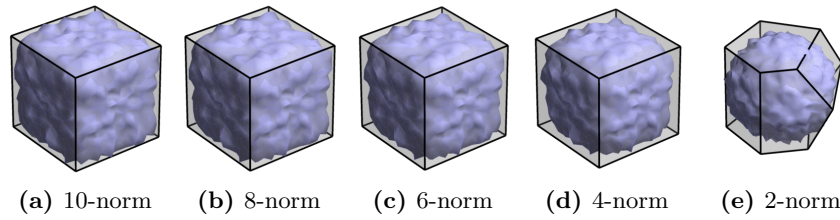
We conclude that the proposed surface reconstruction framework is robust to small random noise as well as a considerable proportion of outliers. The next experiment will cover another practically relevant issue, namely surface meshes that are not exactly describable by the surface model of this thesis.

### 7.2.2.3 Model violation

In practical applications we often need to deal with meshes that systematically deviate from the assumption of piece-wise planar or quadratic surfaces. In this experiment we simulate such a model violation by reconstructing a cube with round edges. Although the roundness of object edges cannot be efficiently represented with the proposed model for man-made surface structures, we would expect the reconstruction to yield a simplified, idealized cubical surface model.

Fig. 7.5 shows the synthetic surface meshes used in this experiment. The cube with smoothed edges is generated as a generalized sphere with non-Euclidean norm. Despite the systematical deviation from the model assumption, i.e. the surface being a compound of piece-wise planar regions, the cube is correctly reconstructed even when given the mesh of a 4-norm sphere. Only the standard 2-norm sphere yields a different result, which is little surprising considering the rotationally symmetric shape without any visible edges.

As shown by this experiment, the reconstruction framework is able to overcome small model violations in favor of a clean reconstruction with a reasonable trade-off between approximation accuracy and model complexity. Restricting the reconstruction to planar surface primitives only is, of course, an important hint for the MDL-based HFC and the subsequent classification. Including a larger number of primitive types would certainly result in larger reconstruction errors.



**Figure 7.5:** Sensitivity analysis w.r.t. model violation on a synthetically generated cube with  $V = 1000$  vertices,  $\sigma = 2.5\%$  Gaussian noise w.r.t. the coordinate range and increasingly smoothed edges. To achieve different amounts of smoothing, the mesh is generated as a generalized sphere with non-Euclidean norm. Note that we only allow planar primitive types in this section. Despite the systematic model violation close to the region boundaries, the cube is correctly reconstructed. Only when applying the procedure to a Euclidean sphere (e) the reconstruction with planar primitive types does not recognize a cube.

We will investigate this effect in Section 7.2.3 and continue with a fourth data property: the sampling density.

#### 7.2.2.4 Sampling density

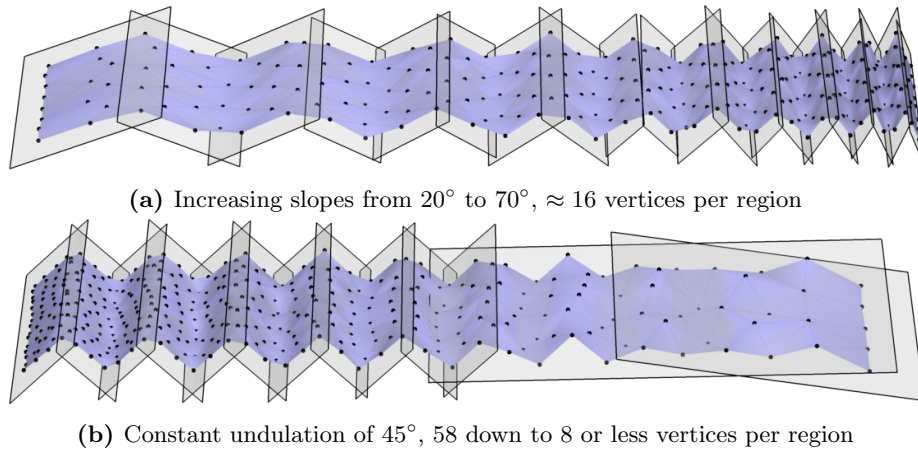
An important data property is the density of the given surface sampling. We want to investigate this aspect empirically, since it can be expected to significantly effect the reconstruction quality.

We analyze two scenarios: First, we generate and reconstruct a piece-wise planar surface with increasing undulation frequency but a constant number of vertices per region (Fig. 7.6a). Second, the undulation is constant and the number of vertices per region decreases (Fig. 7.6b). The points are sampled using a mesh generator from Persson and Strang (2004) such that the rectangular region is covered with exponentially increasing or decreasing point density.

One might expect problems at a high undulation frequency as well as at low vertex density. The planes covered by very few vertices are indeed not correctly segmented. Several regions are merged to one large plane, which is not the expected surface structure. Strong undulations, however, do not cause any reconstruction problems. Since the triangular structure is given, the large slope angles do not hinder the segmentation and thus can be reconstructed correctly. In fact, acute angles even support the segmentation via curvature-adaptive distance transforms, because region boundaries involve large angles between vertex normals. In case the triangular mesh is not given, however, the strong undulation together with limited sampling density would impede the point cloud triangulation, since most methods require a certain sampling density. Fig. 7.7 shows the point cloud from Fig. 7.6a triangulated with the algorithm from Amenta et al. (2000). Especially in areas with strong undulation the method fails due to the limited point density.

In Section 3.1.2 we introduced the concept of local feature size, the distance of surface points to the medial axis. The increasing undulation in Fig. 7.6a leads to a decreasing local feature size  $\text{lfs} \rightarrow 0$ , since the medial axis approaches or even touches the surface. Consequently, the point cloud is an  $\epsilon$ -sample with a very large value  $\epsilon \rightarrow \infty$ . Nevertheless, the reconstruction is correct, suggesting an





**Figure 7.6:** Sensitivity analysis w.r.t. sampling density on synthetically generated meshes with  $\sigma = 0.1\%$  Gaussian noise w.r.t. the maximum coordinate range. In the first experiment (a) the slope of the planes increases from  $20^\circ$  to  $70^\circ$ , while the number of vertices per plane remains constant. Even with very acute intersection angles the surface structure is correctly reconstructed. The second experiment (b) preserves a constant slope, but gradually decreases the sampling density from 58 down to 8 or less vertices per region. Somewhere in the middle the sampling density gets to low for the segmentation to succeed, causing the remaining planes to be partly merged.



**Figure 7.7:** Triangulation of the sparsely sampled surface from Fig. 7.6a with the algorithm from Amenta et al. (2000). Especially on the right-hand side the sampling density is too low for a correct triangulation.

independence of the reconstruction procedure w.r.t. the local feature size as well as the sampling density – given a correct surface triangulation. Triangulation algorithms, however, depend on densely sampled surfaces; therefore it remains an important criterion for reconstructing man-made surface structures.

As shown by the experiments in this section, the proposed reconstruction framework is robust to many defects of the given data. It is especially designed for noisy point clouds and can handle a considerable proportion of outliers. Small model violations can be overcome and a reconstruction is possible even with rather low sampling density, as long as the given triangulation is correct.

### 7.2.3 Sensitivity w.r.t. model assumptions

The proposed model for man-made surface structures was previously defined in Section 3.2.1. It contains multiple control parameters to guide the surface classification. Furthermore, there is the possibility to restrict or to extend the set of surfaces and relations. In this section we analyze both the sensitivity of

the reconstruction w.r.t. the setting of the control parameters as well as the impact of a smaller or larger set of surface classes.

### 7.2.3.1 Control parameters

The classification in Chapter 6 based on unary and ternary factors depends on prespecified control parameters to restrict the distribution of certain surface parameters:  $t_z$  for the zenith angle of a G-PLANE or a G-CYLINDER,  $t_\alpha$  for the opening angle of a V-CONE,  $t_r$  for the radius of a SPHERE, G-CYLINDER or V-CYLINDER and  $t_\delta$  for the contradiction angle between two adjacent surface parameters. Furthermore, we can specify prior probabilities, e.g. for the FREEFORM surface class.

Fig. 7.8 demonstrates the sensitivity of the reconstruction process w.r.t. four of these control parameters. In each experiment we increase the noise  $\sigma$  of a synthetically generated horizontal disk and observe the possible range of the respective control parameter, whose minimum or maximum value is found with a branch-and-bound optimization scheme.

The resulting plots can be interpreted as follows: A horizontal, circular disk with  $\sigma = 1$  % Gaussian noise w.r.t. its diameter is recognized as H-PLANE if the threshold is  $t_z \geq 0.12^\circ$ . Otherwise the estimated normal vector  $\theta_n$  of a G-PLANE is too far from the vertical direction w.r.t. the threshold  $t_z$ , thus yielding a classification of the horizontal disk as G-PLANE. Similarly, in order not to be classified as V-CONE or SPHERE, the opening angle for cones  $\alpha$  needs to be restricted to  $t_\alpha \geq 2.6^\circ$  and the radii need to be restricted to  $t_r \leq 325$  times the disk radius. Last but not least we observe the possible range for the prior of FREEFORM surfaces to be independent of the underlying noise level: A value of  $P(\text{FREEFORM}) \leq 48.8$  % w.r.t. the priors of all other surface classes yields the correct reconstruction as H-PLANE in this example.

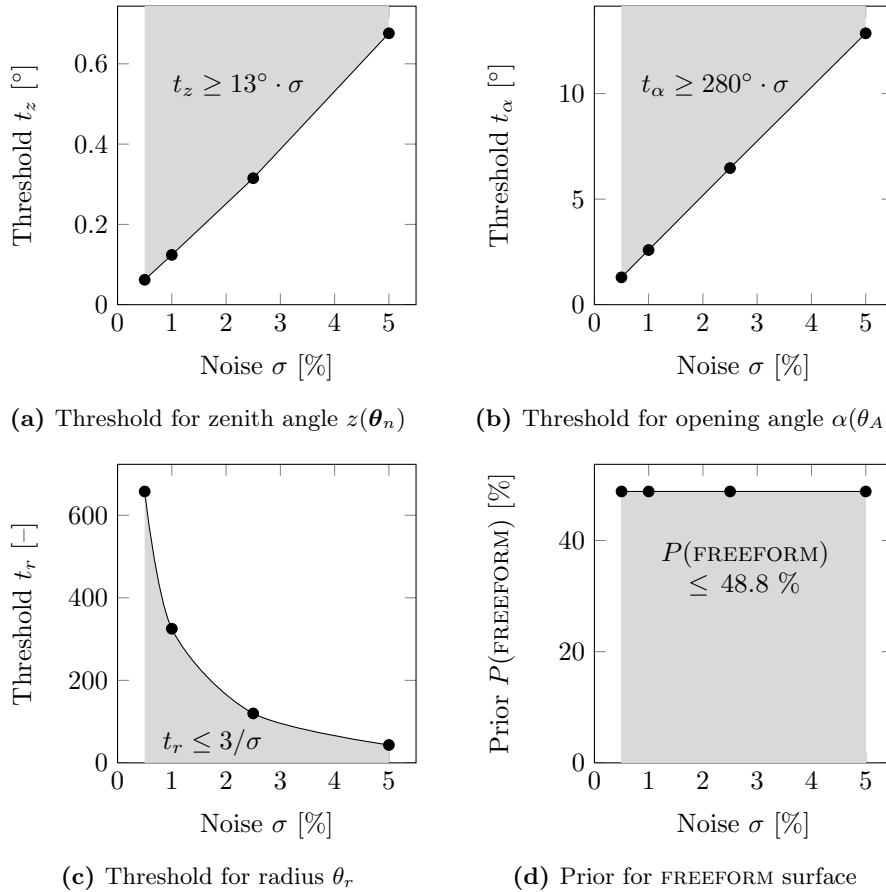
Consequently, the default parameters  $t_z = 1^\circ$ ,  $t_\alpha = 5^\circ$ ,  $t_r = 100$  % and  $P(\text{FREEFORM}) = 1$  % used throughout this chapter (Tab. 7.1), would yield correct results for  $\sigma \leq 2$  % noise w.r.t. the size of a surface primitive, which is a reasonable assumption for most acquisition methods. Note that the linear or hyperbolic dependencies between noise  $\sigma$  and minimum or maximum threshold also depend on surface properties like size and sampling density. They could also be derived algebraically via statistical error propagation.

### 7.2.3.2 Set of surface classes

In Tab. 3.1 we defined eight surface classes: three types of planes, four quadrics and a parameterless FREEFORM surface. As mentioned earlier, this list can be easily extended with other, possibly application-specific primitives like general cones, other quadratic surfaces or tori. The list can be restricted to a subset of primitives as well, e.g. when the surface is known to be piece-wise planar. In this experiment we investigate, whether additional, unnecessary surface classes negatively affect the reconstruction.

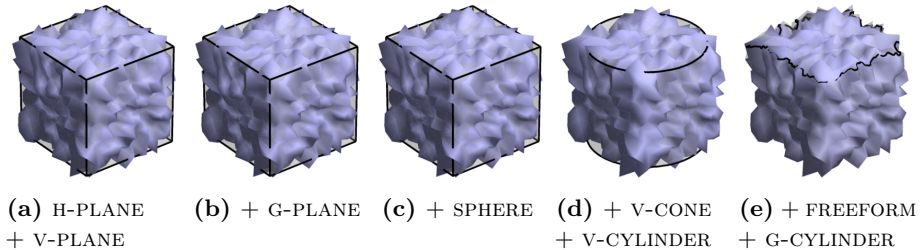
Fig. 7.9 shows a synthetically generated cube with  $\sigma = 5$  % Gaussian noise w.r.t. its edge length. While we only allow the surface classes H-PLANE and V-PLANE in the first experiment, we progressively add more classes in each of the following experiments. Due to the particularly large noise level, the cubical surface structure is not correctly recognized if cylinders are among the set of





**Figure 7.8:** Sensitivity analysis w.r.t. control parameters on a synthetically generated disk of circular shape with  $V = 1000$  vertices, radius 1 and increasing Gaussian noise  $\sigma$ . The shaded areas represent possible values of the respective control parameters for the disk to be reconstructed correctly as H-PLANE and not as G-PLANE (a), V-CONE (b), SPHERE (c) or FREEFORM surface (d). For a realistic noise assumption of  $\sigma \leq 2\%$  w.r.t. the coordinate range, the default values of the control parameters used in this chapter yield correct results. Small deviations do not affect the reconstruction negatively.

surface classes. For smaller noise variances  $\sigma \leq 4\%$ , however, the reconstruction succeeds throughout all five experiments.



**Figure 7.9:** Sensitivity analysis w.r.t. the set of surface classes on a synthetically generated cube with  $V = 1000$  vertices and  $\sigma = 5\%$  Gaussian noise w.r.t. to the coordinate range. In (a) we allow H-PLANE and V-PLANE only and add G-PLANE in (b), SPHERE in (c), V-CYLINDER as well as V-CONE in (d) and G-CYLINDER as well as FREEFORM in (e). With altogether four classes in (c) the reconstruction is still correct. Adding the V-CYLINDER class in (d) causes the four vertical faces to be merged, which does not comply with the ground truth surface structure. Note that we chose a particularly large noise level  $\sigma$ ; a more accurate mesh with  $\sigma \leq 4\%$  leads to correct results.

The obtained results suggest that the set of possible surface classes does not have to be specifically tailored to the given data set. With reasonable small noise and the other data requirements met, the reconstruction succeeds. Of course, here the large angles between the faces support the distinction between planes and cylinders. In case the regions and their mutual angles would be smaller, misclassifications would increase. Under difficult conditions it thus can be advantageous to restrict the model to those surface classes that are actually present in the data, e.g. if one knows the scene is a suburban area where the type of buildings does not vary too much.

As shown in this section, the proposed reconstruction is not very sensitive to model assumptions like the predefined value of control parameters or the given set of surface classes. Therefore, the default values can be retained unchanged in most cases. The clear semantics of these model parameters, however, allow their application-specific tuning.

## 7.2.4 Computational complexity

After evaluating correctness, accuracy and sensitivity of the reconstruction framework, we are interested in its computational complexity. Therefore, we generate synthetic cubes with increasing number of vertices  $V$  and measure the computing times for each major processing step.

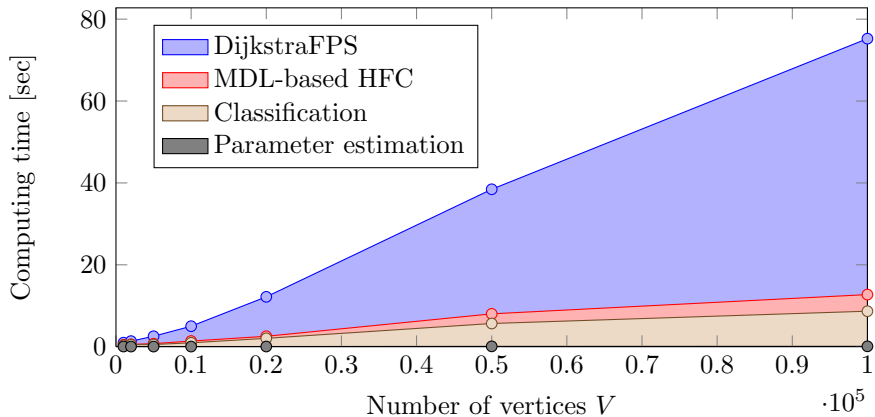
The results are presented in Fig. 7.10. In this experiment most time is spent during DijkstraFPS. For a constant number of regions  $L$ , the theoretical complexity of this step is linearithmic  $O(V \log V)$  with the number of vertices  $V$ , which approximately corresponds to the empirical result.

The classification is the second most expensive processing step, while most of its computing time is spent for estimating primitives for all  $L$  regions and all  $S$  possible surface classes. Building the normal equation matrices depends on

the number of vertices  $V$ , while the inference with the max-sum algorithm only depends on the surface structure.

The MDL-based HFC segmentation and the final constraint parameter estimation use almost constant computing times, since they mainly depend on the number of regions  $L$ . For large meshes, however, the number of possible diffusion operations of boundary vertices increases, which need to be evaluated in each HFC iteration. Thus, its computing time grows as well.

For the largest mesh in this experiment with  $V = 100\,000$  vertices the reconstruction took 62.5 seconds for the DijkstraFPS pre-segmentation, 4.1 seconds for the MDL-based HFC refinement, 8.6 seconds for the classification and 0.1 seconds for the final parameter estimation. Thus, the surface model is reconstructed in about 75 seconds.



**Figure 7.10:** Computing times for synthetically generated cubes with  $\sigma = 0.5\%$  Gaussian noise w.r.t. the object size and an increasing number of vertices  $V$ . The experiment is carried out with a Matlab implementation on a 64-bit Linux machine with a  $2 \times 3.0$  GHz CPU and 8 GB RAM. Times for the four major processing steps DijkstraFPS pre-segmentation, MDL-based HFC refinement, classification and constraint parameter estimation are stacked from top to bottom. Most computing time is spent for the pre-segmentation and the classification. Overall we observe a linearithmic complexity  $O(V \log V)$ . The largest mesh with  $V = 100\,000$  vertices is reconstructed in 75 seconds.

Since every reconstruction approach involving all  $V$  3D points has at least linear complexity  $O(V)$ , the observed linearithmic complexity  $O(V \log V)$  is close to optimal. A more efficient implementation, especially of Dijkstra's algorithm and the local parameter estimation within the surface classification would further improve the timing results.

The reconstruction of carefully designed synthetic meshes helped us to analyze specific properties of the proposed reconstruction framework. In real-world scenarios, however, we often have to deal with the combination of different data defects. Therefore, the following section will address the reconstruction of real data sets and investigate how well the previous findings apply.

### 7.3 Experiments with real data

Real data sets usually contain a variety of defects that hamper a correct reconstruction. In this section we focus on the reconstruction of real meshed point clouds captured with different acquisition techniques. We evaluate the accuracy of the resulting surface model, demonstrate the effect of different model parameters and present computing times for all experiments on real data sets.

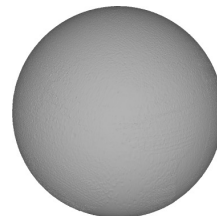
#### 7.3.1 Accuracy analysis on calibration objects

In contrast to the accuracy evaluation in Section 7.2.1 we will analyze the reconstruction of two real test objects in this section. Both point clouds, a BALL and a CONE, are captured using a high-precision laser scanning arm. Since we are interested in the accuracy of estimated surface parameters and not in the topology of the surface structure, we skip the triangulation as well as the segmentation and simply label all points as being member of one single region  $L = 1$ . After reconstructing the original point clouds with up to 1 million points, we repeat the experiment on downsampled point clouds with 10 000 points.

As shown in Tab. 7.2, the radius of the BALL is estimated with an accuracy of  $0.14 \mu\text{m}$ . After downsampling the point cloud by a factor  $\approx 100$ , the accuracy is  $1.4 \mu\text{m}$ , which perfectly conforms to the theoretical variance propagation. Apart from that, the inlier ratio remains almost constant.

**Table 7.2:** Reconstruction accuracy of a laser scanned BALL. With the original point cloud of about 1 million points the radius  $\theta_r$  is estimated with an accuracy of  $\sigma_{\theta_r} = 0.14 \mu\text{m}$ . In conformance with theoretical considerations the uncertainty increases by a factor  $\sqrt{\frac{1\,057\,824}{10\,000}} \approx 10$  when reconstructing a point cloud with only 10 000 randomly selected points. The percentage of inliers remains constant.

Number of points $V$	1 057 824	10 000
Point uncertainty $\sigma$	0.1 mm	0.1 mm
Radius $\hat{\theta}_r$	80.104 78 mm	80.106 72 mm
Uncertainty $\sigma_{\theta_r}$	0.000 14 mm	0.001 38 mm
Variance coefficient $\hat{\sigma}_0$	1.19	0.93
Inlier ratio	98.26 %	98.28 %



The result for the CONE shown in Tab. 7.3 is similar. The opening angle is estimated with an accuracy of  $0.000\,21^\circ$ . After downsampling the point cloud by a factor  $\approx 46$ , the accuracy is about  $\sqrt{46} = 6.7$  times larger, namely  $0.001\,40^\circ$ . Again, the inlier ratio remains almost constant.

As already discussed in Section 7.2.1, individual surface regions can be accurately reconstructed. The empirically obtained accuracies conform to the theoretical derivations. Of course the parameter estimation depends on a correct segmentation, which we assumed to be given in these two experiments.

Unfortunately, there is no sufficiently accurate ground truth parametrization available for these calibration objects. Nevertheless, this experiment demonstrates that we can use the proposed framework to analyze properties of geometric objects like point residuals w.r.t. to a best fitting geometric primitive and the

**Table 7.3:** Reconstruction accuracy of a laser scanned CONE. With the original point cloud of about 0.46 million points the opening angle  $\alpha = \alpha(\theta_A)$  is estimated with an accuracy of  $\sigma_\alpha = 0.00021^\circ$ . In conformance with theoretical considerations the uncertainty increases by a factor  $\sqrt{\frac{456\,064}{10\,000}} \approx 6.7$  when reconstructing a point cloud with only 10 000 randomly selected points. The percentage of inliers remains almost constant.

Number of points $V$	456 064	10 000
Point uncertainty $\sigma$	0.1 mm	0.1 mm
Opening angle $\hat{\alpha} = \alpha(\hat{\theta}_A)$	39.983 25°	39.986 26°
Uncertainty $\sigma_\alpha$	0.000 21°	0.001 40°
Variance coefficient $\hat{\sigma}_0$	0.47	0.47
Inlier ratio	99.90 %	99.85 %



corresponding inlier ratio: While the BALL with full resolution yields point uncertainties of  $\hat{\sigma} = \hat{\sigma}_0 \cdot \sigma = 0.12$  mm with 1.74 % outliers, the CONE seems to be more accurately manufactured with  $\hat{\sigma} = \hat{\sigma}_0 \sigma = 0.05$  mm and 0.10 % outliers.

### 7.3.2 Flexibility w.r.t. different acquisition techniques

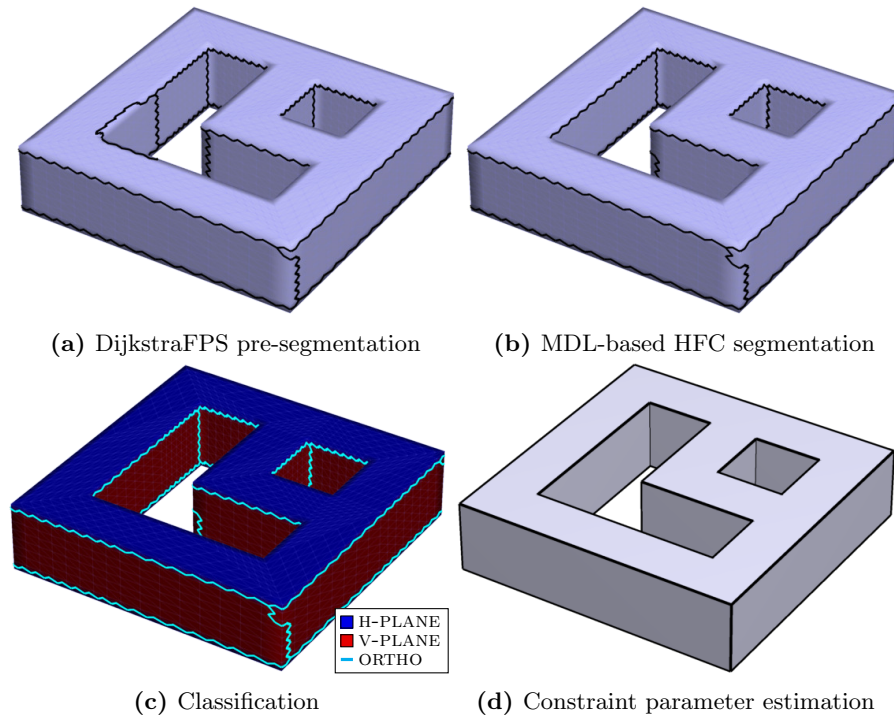
The goal of this thesis is to provide a reconstruction procedure for man-made surfaces independent of the acquisition technique used for capturing the input data. Therefore, this section demonstrates the application to different types of data sets. This involves two examples of noise-free CAD meshes, two multi-view stereo reconstructions, a 2.5D surface mesh captured via structured light projection and two terrestrial laser scans.

#### 7.3.2.1 CAD meshes

Meshes created via computer aided design (CAD) are usually a compound of noise-free, parametrized surface parts. Strictly speaking they are synthetically generated, thus should have been addressed earlier in Section 7.2. Since we, however, did not design them ourselves to fulfill certain requirement but obtained them from publicly available sources, we classify them as *real* data sets. We will discuss two data sets obtained from the AIM@SHAPE Shape Repository ([shapes.aimatshape.net](http://shapes.aimatshape.net)): one rather trivial example called DOUBLE TORUS and the popular but more complex surface structure called FAN DISK.

The reconstruction of the simple DOUBLE TORUS is shown in Fig. 7.11. The points are completely free of random noise and perfectly sample the planar surface primitives. Nevertheless, we assume a noise level of  $\sigma = 0.1$  % w.r.t. to the maximum coordinate range to avoid divisions by zero when evaluating description lengths and singular covariance matrices when estimating surface parameters. The simple surface structure of horizontal and vertical planes is almost perfectly segmented during the first processing step, the DijkstraFPS pre-segmentation. Only one oversegmented region on the left-hand side needs to be merged by the MDL-based HFC refinement. Due to the lack of data noise, the classification and final parameter estimation is trivial.

Note that the curvy boundaries are caused by the different point semantics in this data set: As discussed in the context of Fig. 3.2, we assume a point

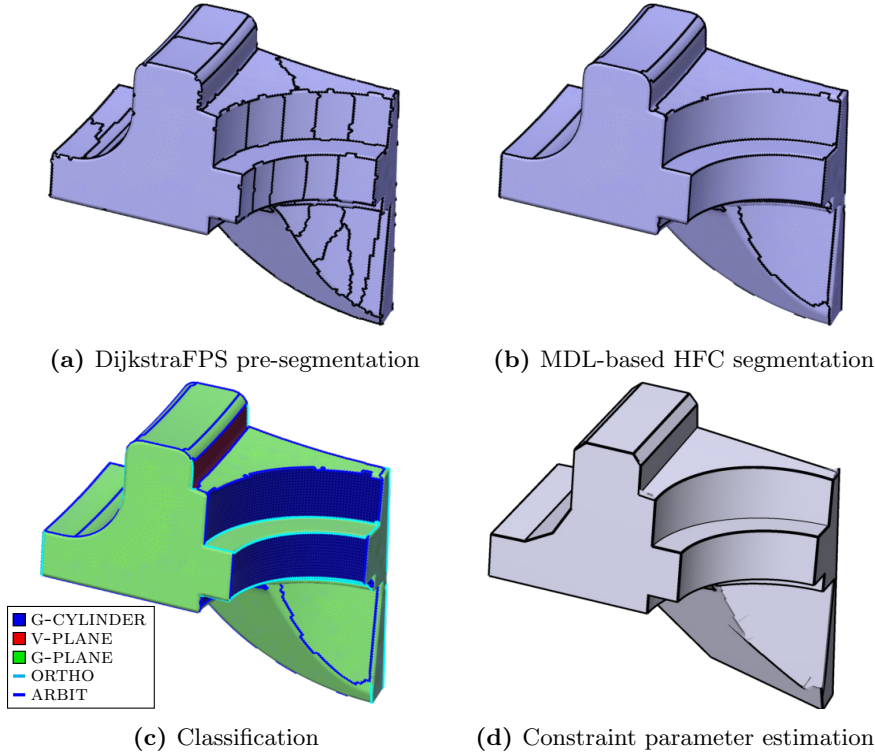


**Figure 7.11:** Reconstruction of the DOUBLE TORUS CAD data set with  $V = 2686$  vertices. Although the points are free of noise, we assume a noise level of  $\sigma = 0.1\%$  w.r.t. to the maximum coordinate range to avoid numerical problems. The surface structure is almost perfectly segmented during the DijkstraFPS pre-segmentation step (a). The small oversegmentation is removed by the MDL-based HFC refinement (b). Given the perfect segmentation and the surface structure with planar primitives only, the final classification and reconstruction is trivial (c, d). Data set from the AIM@SHAPE Shape Repository ([shapes.aimatshape.net](http://shapes.aimatshape.net)).

to correspond to exactly one surface region. In this example, however, some points lie on the boundary between two adjacent regions. Since the segmentation boundary passes through triangles and not through vertices, it is curved but correct.

A more complex example is given in Fig. 7.12. Again, we assume a non-zero noise level of  $\sigma = 0.2\%$  w.r.t. to the maximum coordinate range. Besides the suppression of numerical problems, we need to account for small deviations of the given mesh w.r.t. our model of man-made surface structures: The FAN DISK is a small section of a larger, rotationally symmetric object. Therefore, the vertically appearing planes are actually slanted and rounded object edges on top are no cylinders but tori. Considering these deviations from our model for man-made surface structures, the reconstruction is entirely satisfactory.

In this example we observe pairs of adjacent surface classes that might be added to the list of possible combinations in Tab. 3.2: Since the G-PLANE between the V-PLANE and the upper G-CYLINDER is actually curved due to the rotational symmetry, it should be reconstructed as SPHERE with a very large radius  $\theta_r$ . Besides the corresponding threshold  $t_r$  to be adjusted, we would need to add



**Figure 7.12:** Reconstruction of the FAN DISK CAD data set with  $V = 25\,894$  vertices. To avoid numerical problems and to account for small model deviations we assume a noise level of  $\sigma = 0.2\%$  w.r.t. the maximum coordinate range. While the DijkstraFPS pre-segmentation oversegments the curved surface regions due to the curvature-adaptive distance metric (a), the primitive fitting approach of the MDL-based HFC refinement yields a reasonable segmentation (b). The final classification and parameter estimation is adequate given the limited set of surface primitives (c, d). Data set from the AIM@SHAPE Shape Repository ([shapes.aimatshape.net](http://shapes.aimatshape.net)).

ARBIT relations for cylinders and planes adjacent to a SPHERE. In this thesis we will stick to the current set of combinations and leave the extension of Tab. 3.2 up to future work.

Another problem we observe with this data set arises from the direct solutions for fitting cylinders and cones defined in Appendix A.1. Given a small section of a large cylinder or cone, the initial solution for a subsequent parameter estimation is often far off the correct solution, possibly leading to a diverging estimation process. An alternative solution based on fitting graph surfaces like in Section 2.1.2.2 might be better suited for such cases. Making use of vertex normals might as well be a promising approach for initializing surface parameters more robustly. This is, however, out of scope of this thesis.

Although some weakly curved regions of the FAN DISK example are approximated by planes, the reconstruction of both CAD models was successful. Furthermore, the proposed model for man-made surface structures is flexible

enough to be adjusted to more complex data sets. We will continue with visually reconstructed meshes of real objects.

### 7.3.2.2 Multi-view stereo point clouds

An emerging acquisition technique for reconstructing terrestrial 3D point clouds is the multi-view stereo approach. In addition to the example in Section 7.1 we will reconstruct two such data sets in this section: While the first one is a CONCRETE BLOCK intuitively describable as a piece-wise planar surface structure, the second one, a decorative FACADE, can only be approximated as such. Both point clouds were derived using the structure-from-motion system Bundler (Snavely et al., 2006) and the dense patch-based multi-view stereo reconstruction PMVS2 (Furukawa and Ponce, 2010). The triangular mesh is generated using the Poisson surface reconstruction from Kazhdan and Hoppe (2013).

Fig. 7.13 shows the reconstruction of the CONCRETE BLOCK. We assume the vertices to be disturbed by random Gaussian noise with  $\sigma = 0.3\%$  standard deviation w.r.t. the length of the block, which is about 4 mm in reality and slightly larger than the possible accuracy of a multi-view stereo reconstruction. After obtaining a clear oversegmentation by the DijkstraFPS pre-segmentation, the MDL-based HFC correctly refines the segmentation and yields four vertical and one horizontal regions and eight little hats consisting of four regions each.

The final reconstruction is satisfactory, although the flattened tip of each hat is missing. This is because the structure-from-motion system Bundler in combination with the dense point cloud reconstruction PMVS2 creates a number of holes on top of the point cloud – visible as dark spots in Fig. 7.13b –, which probably was observed by too few camera views or with too shallow viewing angles. These holes result in a quite bumpy Poisson surface, misleading the reconstruction of a more accurate surface structure.

One might argue that the noise  $\sigma$  is overestimated and therefore many details get lost. When decreasing the noise level, some more detailed object edges are correctly covered with additional segmentation boundaries. Others are missing, especially where the sampling density is too small to yield sufficiently sharp edges. This imbalance results in a topologically unreasonable configuration and a worthless reconstruction.

The second example for multi-view stereo reconstructions is presented in Fig. 7.14. Since windows are partly transparent partly reflective, they cause many artifacts in form of outliers and tails towards the inside of the building. As argued in Section 3.1.3, methods for interpreting facades based on images and/or point clouds exist, allowing to detect doors, windows, balconies and similar structures. Therefore, we preprocess the point cloud by manually removing all windows and the door before reconstructing the Poisson surface.

For reconstructing the surface structure we use the less restrictive threshold  $t_z = 5^\circ$  for the zenith angle of planes, since we expect the amount of stucco and decorative elements to perturb the parameters of vertical planes. Furthermore, we restrict the reconstruction to planar primitives only, since apparently there are no relevant quadrics within the given point cloud. Allowing the use of all eight surface classes would introduce unnecessary ambiguities and topological inconsistencies.

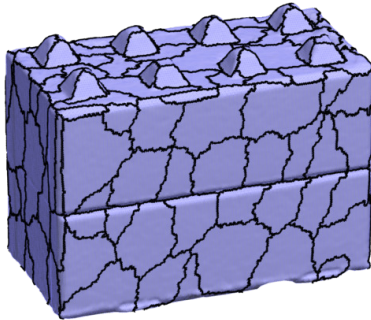




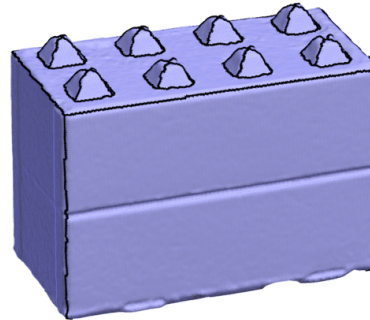
(a) One out of 62 images



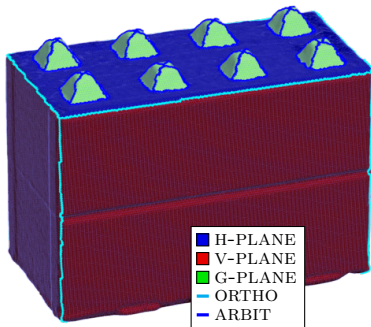
(b) Reconstructed point cloud



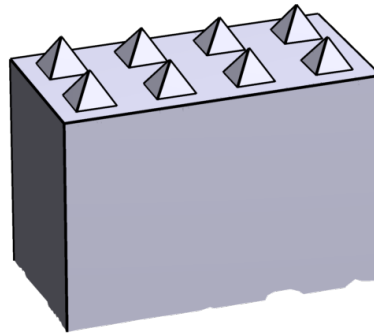
(c) DijkstraFPS pre-segmentation



(d) MDL-based HFC segmentation

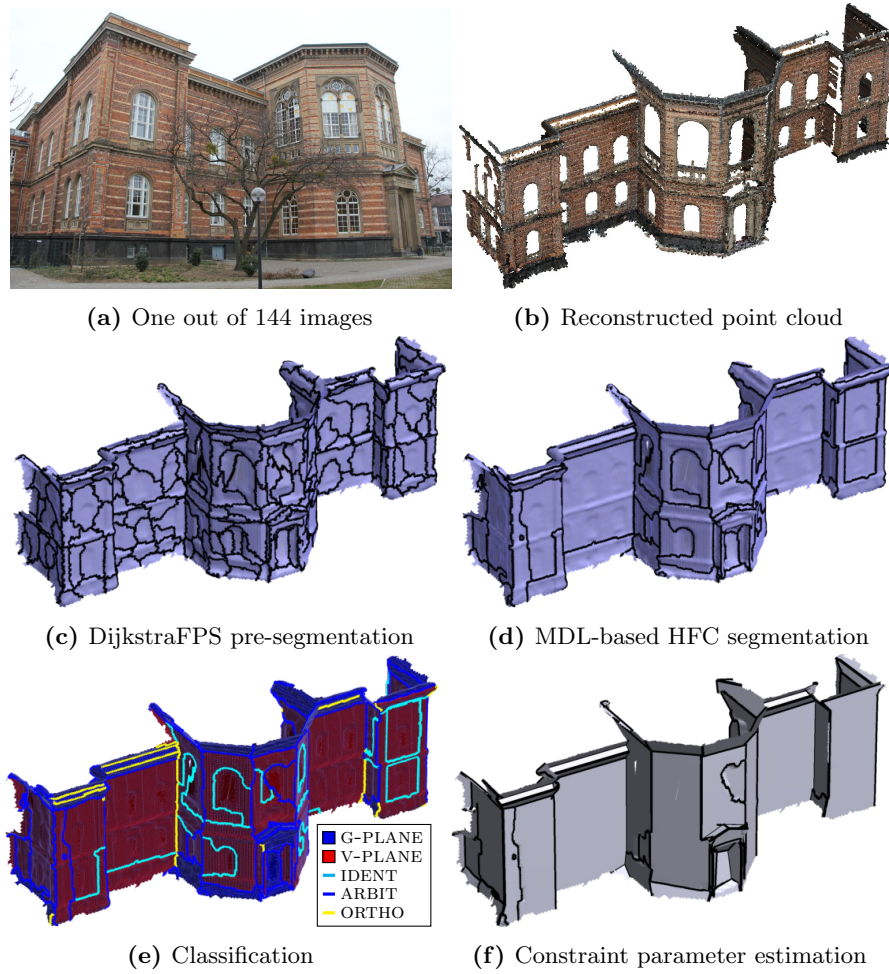


(e) Classification



(f) Constraint parameter estimation

**Figure 7.13:** Reconstruction of the multi-view stereo mesh CONCRETE BLOCK with  $V = 108\,650$  vertices. A noise assumption of  $\sigma = 0.3\%$  w.r.t. to the length of the block yields the depicted result. Unfortunately, the structure-from-motion reconstruction yields several holes on top of the point cloud, which causes bumps and dents on and around the eight little hats (b). Therefore, the segmentation fails to preserve the horizontal planes on top of each hat (c, d), leading to the reconstruction of eight pyramids instead of eight frustums (f).



**Figure 7.14:** Reconstruction of the multi-view stereo mesh FACADE with  $V = 44\,184$  vertices. We use a more relaxed threshold  $t_z = 5^\circ$  for the zenith angle of planes as well as a noise level of  $\sigma = 5$  cm, which is about 0.1 % of the maximum coordinate range. Windows were cut out before starting the reconstruction procedure. Some areas are successfully reconstructed with planes, others do not well correspond to our model of man-made surface structures. Especially the entrance and the roof edge are to complex considering the available point density.

In most areas the resulting surface structure is a reasonable approximation. Some regions, however, are oversegmented, while others, especially around the entrance and the roof edge, do not correctly represent the true surface. The point density of the latter object parts is too small for an accurate reconstruction within the proposed framework. It is even doubtful that humans are able to provide a reasonable piece-wise planar reconstruction of the given point cloud, considering the many holes and ambiguous details.

The data sets presented in this section were in principle successfully reconstructed. Locally, however, both suffer from too small sampling density w.r.t. the size of small object parts. While this defect could be easily resolved for the CONCRETE BLOCK by capturing additional images, we will not be able to avoid the problem for the FACADE data set: Although additional images will increase the point density and unveil new object details, these details are usually not describable with our simple surface model and its limited set of primitive types.

Thus, the proposed reconstruction framework is applicable for levels of detail, where an efficient description with piece-wise planar or quadric surface regions is possible. It is not suited to approximate meshes with low point density, i.e. few points per surface region. In such cases the probability of topological inconsistencies increases as shown in Fig. 7.6b, which is currently not taken care of.

### 7.3.2.3 Structured-light reconstruction

After reconstructing two surface meshes generated with the passive multi-view stereo approach, this section gives an example for an active depth sensor based on projecting structured light. We use a recently popularized low-cost sensor, namely the Microsoft Kinect depth camera. It is based on projecting a known pattern of infrared light onto the target object and observing the object with an infrared camera. Camera and projector are embedded in a small camera rig with a basis of 7.5 cm. For each pixel the disparity and the intersected 3D location is computed in real-time. After connecting these 3D points according to their horizontal, vertical and diagonal adjacencies within the image grid, we obtain a 2.5D triangular mesh.

Fig. 7.15 shows the KINECT data set. It represents a small desktop scene with a ground plane and multiple objects conformable to our assumptions about man-made objects.

Before starting to reconstruct the surface structure, we preprocess the mesh: First we remove long triangular edges that occur when the foreground object partly occludes the background. We cut out the region of interest, select the largest connected component and restore the vertical direction, in our case using the horizontal ground surface. In order to suppress discretization effects, caused by propagating the integer disparities to discrete depth values, we apply Laplacian smoothing and subsequently a feature preserving two-step smoothing, both operations being part of the 3D mesh processing software system *MeshLab*.

As expected for partly curved surfaces, the DijkstraFPS pre-segmentation oversegments the surface. In the middle, however, the bucket and the ground surface share a common region. Both deficiencies are corrected by the MDL-based HCF refinement. Especially for the undersegmented region in the middle, the diffusion operations pay off, since a correct segmentation cannot be reached

with merge operations alone. For the classification step we use more relaxed thresholds for both the zenith angle of planes  $t_z = 5^\circ$  as well as the contradiction angle  $t_\delta = 5^\circ$  between adjacent surface parameters. This causes vertical regions more likely to be classified as V-PLANE and adjacent regions more likely to be IDENT, like the individual regions on the bucket. Note that the bucket is actually cone-shaped with a too small opening angle  $\alpha$  and therefore classified as V-CYLINDER. Furthermore, the rim of the bucket has indeed a larger radius than the remaining surface; the adjacency of two different V-CYLINDER surfaces, however, is not supported by our current model for man-made surfaces.

In this single-view example the captured objects are only partly reconstructed: Two vertical regions of the large box are not observed and consequently not contained in the final surface structure. The cylindrical bucket, however, is sufficiently defined by the region facing the camera. Combining multiple views via so-called *mesh zipping* would further stabilize the reconstruction.

This section demonstrated the applicability of the proposed reconstruction framework to structured-light reconstructions. For the experimental evaluation we made use of the low-cost Kinect depth camera, which suffers from discretization effects and is limited to small indoor scenarios. In the following section we will use a more elaborate sensor in terms of accuracy, density and applicability.

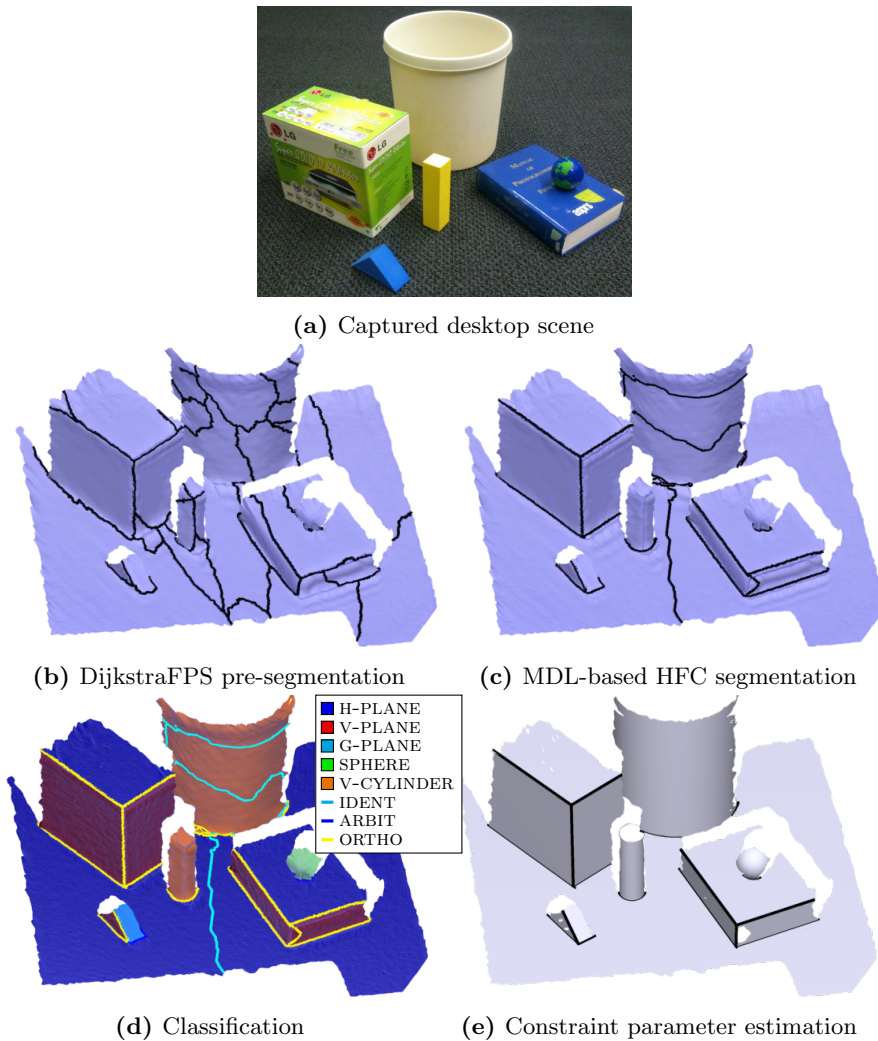
#### 7.3.2.4 Terrestrial laser scan

The data set used in this section is captured with a terrestrial laser scanner from multiple view points around a small building. The original point cloud with 4.4 million points is the basis for the reconstructed Poisson surface shown in Fig. 7.16. Note that a larger reconstruction depth in form of a finer voxel grid for the Poisson surface reconstruction does not further improve the surface mesh, since many small details are smoothed away due to unstable vertex normals. Therefore, we use the mesh with  $V = 195\,154$  vertices as an approximation of the original data points.

To take the rough roofing tiles into account, we use a noise level of  $\sigma = 5$  cm, which is significantly larger than the actual precision of the laser scanner. Due to the large noise level many windows and doors are merged with the surrounding wall. Three of them are correctly segmented; but their indentation is too small for a 3D reconstruction with frames being represented by individual regions. Instead, they are classified as IDENT to the wall. Introducing a *parallel* relation could account for such indented regions. The resulting surface structure would, however, no longer be a watertight compound of planes.

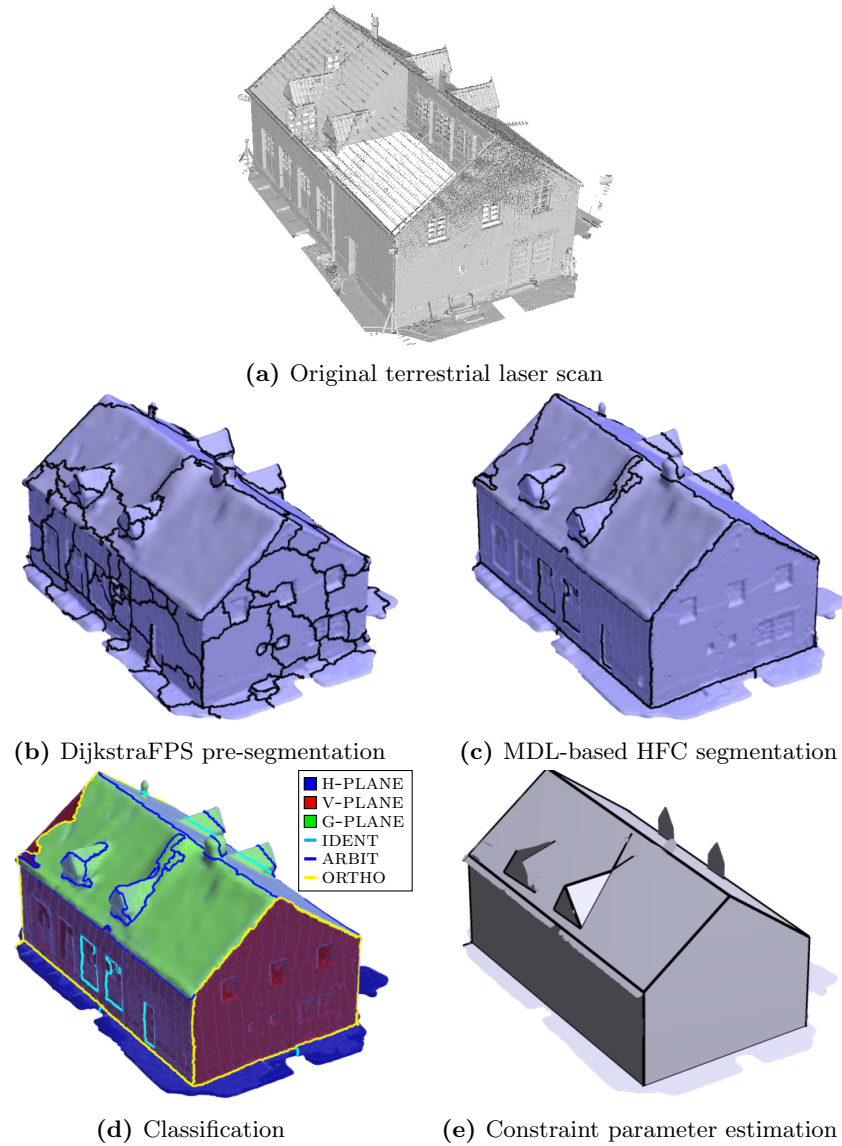
While a terrestrial laser scanner yields the most accurate points with largest density, the inconvenient registration of additional view points makes it difficult to fully capture a surface without occlusions. This causes the largest defect of the resulting reconstruction: The low viewing positions on street level lead to large occlusions on top of the roof and behind the dormers. Therefore, we obtain a bumpy Poisson surface which misleads the segmentation and finally the determination of the surface structure.

The basic shape of the building and the vertical dormer elements, however, are correctly reconstructed. With a better sampling on top we would certainly obtain a topologically consistent, watertight surface model.



**Figure 7.15:** Reconstruction of the KINECT data set with  $V = 137\,487$  vertices. We assume a Gaussian noise of around  $\sigma = 2$  mm, which is about 0.2 % of the maximum coordinate range. The oversegmentation obtained with DijkstraFPS (b) is mostly removed with the MDL-based HFC refinement (c). To suppress unimportant details like the small inclination of vertical planes and the slightly wider rim of the bucket, we loosen the thresholds for the zenith angle  $t_z = 5^\circ$  and the contradiction angle  $t_\delta = 5^\circ$ , leading to an almost perfect classification (d). One tiny mistake, however, remains in the final estimation (e): The small rectangular prism is undersegmented, causing it to be reconstructed as a V-CYLINDER and not as a compound of three planes.





**Figure 7.16:** Reconstruction of the BUILDING laser scan with  $V = 195\,154$  vertices. Although a terrestrial laser scanner is able to capture points with higher precision, we choose a noise level of  $\sigma = 5$  cm, about 0.25 % of the maximum coordinate range, to account for the roughness of the roofing tiles and other minor elements. Like with the FACADE data set in Fig. 7.14, the doors and windows cannot be completely reconstructed and might be removed within a preprocessing step. Some of them are even reconstructed as separate regions based on their indentation. A *parallel* relation would be required to avoid them to be merged with the surrounding wall, since the orthogonal planes of the frames are missing. The poor sampling on top of the roof and behind the dormers causes a very bumpy Poisson surface and thus prevents the roof to be reconstructed correctly.

### 7.3.2.5 Laser scanning arm

As a final acquisition technique we evaluate a surface mesh captured using a high-precision Romer Infinite 2.0 measuring arm with a Perceptron ScanWorks V5 triangulation laser scanner attached. With this setup it is possible to reach a resolution of about 0.015 mm and 0.045 mm accuracy within a range of 1.4 m radius.

The object, a small paper house with a footprint of about  $12\text{ cm} \times 9\text{ cm}$  as previously shown in Fig. 1.1, originally consists of about 600 000 triangulated 3D points. In order to speed up the reconstruction and sensitivity analysis in the following section, the mesh is downsampled using quadric edge collapse decimation. This operation, a variant of the approach from Garland and Heckbert (1997), is contained in the mesh processing software system *MeshLab* and creates a simplified mesh with  $V = 50\,854$  vertices shown in Fig. 7.17.

To account for small deviations from perfect planes, we choose a point uncertainty of  $\sigma = 0.4\text{ mm}$ . The combined segmentation with DijkstraFPS and MDL-based HFC yields perfectly segmented surface regions. Subsequently, the global surface classification yields visually reasonable results. Only two of the slanted roof edges might be expected being ORTHO, but are classified as ARBIT. The redundancy contained in the surface structure, however, enforces the roof surfaces to be orthogonal to the front and back wall.

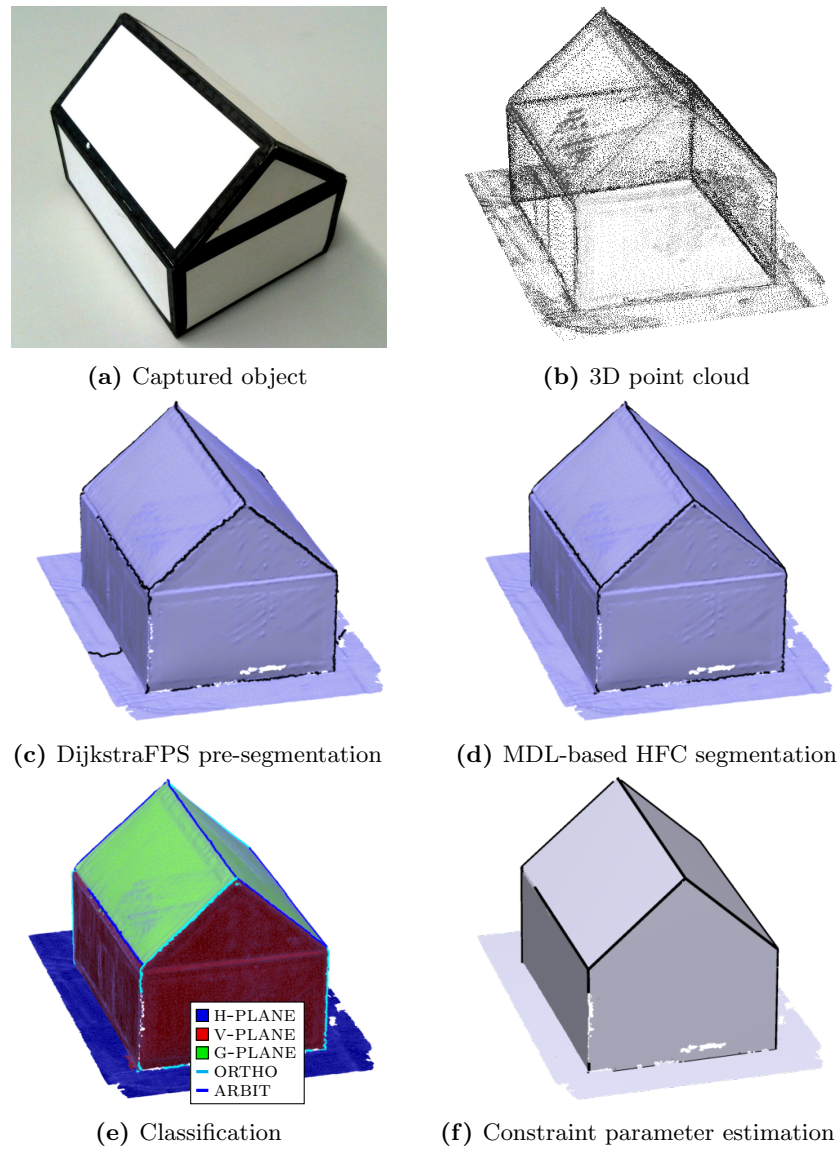
This example illustrates the deficiency of the reconstruction method mentioned in Section 6.5: The classification yields a topologically inconsistent configuration. Since front and back wall of the object are parallel, the ORTHO and ARBIT relations with the roof plane contradict each other. Here the orthogonality constraint simply overrules the ARBIT relation. In other cases such inconsistencies can cause significant defects in the final reconstruction.

As demonstrated in this section, the proposed reconstruction framework is flexible enough to be applied to various kinds of surface meshes. Problems occurred, where a low sampling density led to inaccurate triangulations or where the true surface is not describable with our model for man-made surface structures.

### 7.3.3 Sensitivity w.r.t. model assumptions

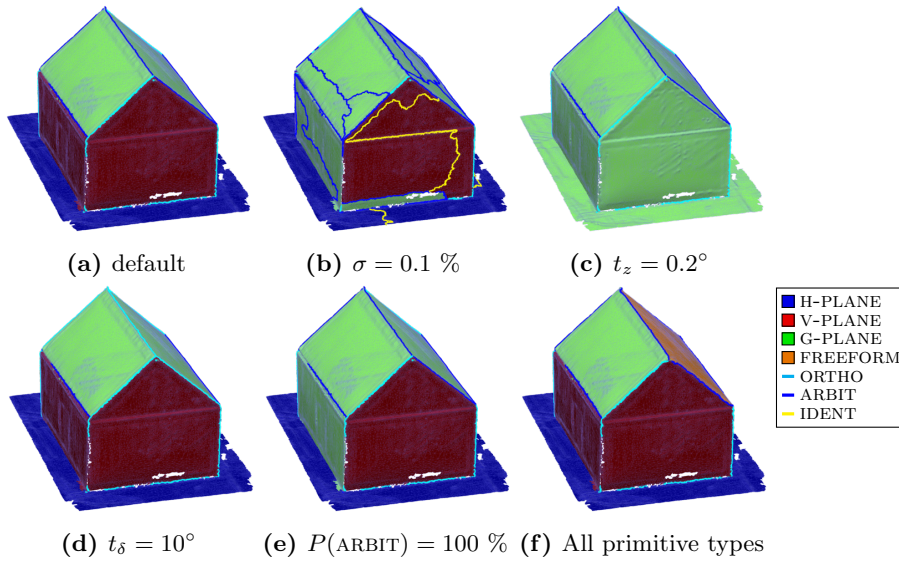
For synthetic meshes we analyzed the sensitivity w.r.t. control parameters and the set of surface classes in Section 7.2.3. Now we are interested in the effects on real data sets. Therefore, we reconstruct the PAPER HOUSE from Fig. 7.17 again, but with different model assumptions.

Fig. 7.18 shows the reconstruction results with six different parameter settings. Since most control parameters mainly affect the classification, we focus on the visualization of obtained surface and relation classes. Aside from the default parameters we try to use a smaller noise level  $\sigma$ , a smaller parameter  $t_z$  controlling the zenith distance of normal vectors, a larger tolerance  $t_\delta$  for the difference between adjacent parameter vectors, a larger prior probability for arbitrarily related surfaces  $P(\text{ARBIT})$  as well as reconstructing the surface with all primitive types instead of only planes.



**Figure 7.17:** Reconstruction of the PAPER HOUSE laser scan with  $V = 50\,854$  vertices and an assumed point uncertainty of  $\sigma = 0.4$  mm. The DijkstraFPS pre-segmentation yields an almost perfect segmentation with only the ground plane being split into two regions (c). This defect is corrected by the MDL-based HFC refinement (d). Apparently the object is not built accurately enough such that all four slanted roof edges are classified as being ORTHO; two of them seem to be ARBIT (e). But for all other surfaces and relations the classification is visually correct and the redundancy leads to orthogonalities at the two mentioned ARBIT relations as well (f).





**Figure 7.18:** Sensitivity analysis with the PAPER HOUSE laser scan. The surface mesh from Fig. 7.17 is reconstructed with different model parameters: Compared to the default parameters  $\sigma = 0.2\%$ ,  $t_z = 1^\circ$ ,  $t_\delta = 1^\circ$  and  $P(\text{ARBIT}) = 1\%$  with planar primitives only (a), we decrease the noise level  $\sigma$  (b), decrease the zenith angle tolerance  $t_z$  (c), increase the tolerance  $t_\delta$  for angular deviations between adjacent parameter vectors (d), increase the prior probability  $P(\text{ARBIT})$  for arbitrarily related surfaces (e) and reconstruct the surface with all eight primitive types. We can conclude from this experiment that adjusting the control parameters reasonably affects the level of detail and the amount of simplification of the resulting surface model.

**Noise level  $\sigma$ .** The assumed noise level  $\sigma$  controls the level of detail of the resulting reconstruction. A value of  $\sigma = 0.1\%$ , half as much as in Fig. 7.17, yields many surface parts being split into multiple regions, some of them later being classified as IDENT (Fig. 7.18b). This reconstruction is reasonable for such a low noise assumption, since the object parts built from paper are not perfectly planar. Thus, the noise level  $\sigma$  is a useful parameter to control the level of detail of the reconstructed surface model.

**Threshold on the zenith distance  $t_z$ .** The parameter  $t_z$  controls the deviation of a normal vector's zenith distance from the vertical or horizontal direction. Allowing a G-PLANE to have zenith distances of  $t_z = 0.2^\circ$  or smaller leads to all planes being classified as G-PLANE rather than H-PLANE or V-PLANE (Fig. 7.18c). Note that identified ORTHO relations amplify this effect: If the ground plane is slightly slanted with large probability and it is ORTHO to a vertical wall, then the wall will certainly not be classified as V-PLANE. Apparently the threshold  $t_z$  is an important parameter for controlling the amount of simplification of the idealized surface model.

**Threshold on the difference between adjacent parameters  $t_\delta$ .** Constraints between two adjacent parameters are either formulated as equality, orthogonality or parallelism. The angular deviation from this relation is con-

trolled by the parameter  $t_\delta$ . Increasing the value to  $t_\delta = 10^\circ$  not only yields the previously missing ORTHO relations at the slanted roof edges, but an ORTHO ridge as well (Fig. 7.18d). As shown in Fig. 1.1c, the ridge angle of  $96.2^\circ$  actually deviates from a right angle significantly. Thus, the parameter  $t_\delta$  yields another possibility to control the simplification of the idealized model compared to the real, data-driven model.

**Prior probability for arbitrarily related surfaces  $P(\text{ARBIT})$ .** In order to use restrictive relations like ORTHO or IDENT in favor of non-restrictive ARBIT relations, we usually use a smaller prior probability  $P(\text{ARBIT})$ . If not doing so, as in Fig. 7.18e, the ORTHO relation is less likely to occur in the final surface structure. On the left side of the object the relation between ground plane and vertical wall seems to support the V-PLANE classification of the wall with default parameters, since it becomes a G-PLANE when setting equal prior probabilities for all relations. Thus, the prior probabilities affect the reconstruction as well and might be particularly useful if learned from training data sets.

**Set of primitive types.** Last but not least, we modify the set of surface primitives that can be used for the surface model reconstruction. While it is favorable to restrict this set to expected primitives, in some cases such an expectation might not be available. When including quadrics and parameterless FREEFORM surfaces for reconstructing the piece-wise planar PAPER HOUSE, one part of the roof is classified as FREEFORM surface (Fig. 7.18e). Within the specified noise and model assumption, this is supposedly the most likely configuration of surface and relation classes. This reconstruction is indeed correct and complies with the declared assumption that FREEFORM surfaces are present in the data.

As shown in this section, the different adjustable parameters of the proposed model for man-made surface structures allow to control both the level of detail as well as the amount of simplification during the reconstruction procedure. Most parameters affect the classification and therefore the idealized surface model only. The noise assumption  $\sigma$ , however, has impact on all three major processing steps.

### 7.3.4 Computational complexity

In Section 7.2.4 we analyzed the computing times for four major processing steps: the DijkstraFPS pre-segmentation, the MDL-based HFC refinement, the global surface classification as well as the constraint parameter estimation. Clearly most of the time was spent for the pre-segmentation. In the following we will investigate, how well these findings transfer to real data sets.

Tab. 7.4 lists the computing times for all data sets processed throughout Sections 7.1 and 7.3. Some data sets like BRICKS and PAPER HOUSE indeed yield the same time pattern. Others, however, like FACADE and BUILDING take nearly as much time for the MDL-based HFC refinement as for the DijkstraFPS pre-segmentation. For the FAN DISK data set the refinement takes even four times as long. The effect is caused if the surface does not fully comply with the model for man-made surfaces, is heavily disturbed by noise and outliers or contains bad sampling and holes. Then the pre-segmentation might fail to

yield a correct and complete segmentation, causing significantly more iterations for the HFC refinement to converge. Especially the boundary vertex diffusion is expensive for moving boundaries over large distances to replace a missing boundary from the pre-segmentation. With the FAN DISK data set there is the additional problem of very flat quadrics, for which the initializations in Appendix A.1 yield unstable results. This further slows down the convergence or even causes oscillations.

**Table 7.4:** Computing times for all real data sets from Sections 7.1 and 7.3. The numbers indicate minutes and seconds required for each of the four major processing steps: DijkstraFPS pre-segmentation, MDL-based HFC refinement, classification and constraint parameter estimation. The computation is performed with a Matlab implementation on a 64-bit Linux machine with a  $2 \times 3.0$  GHz CPU and 8 GB RAM.

Data set	Fig.	$V$	FPS	HFC	Class.	Est.	Total
BRICKS	7.1	6 713	0:06.9	0:01.1	0:02.2	0:00.6	0:10.8
D. TORUS	7.11	2 686	0:01.7	0:01.2	0:00.7	0:00.1	0:03.7
FAN DISK	7.12	25 894	0:33.8	2:00.3	0:08.0	0:00.5	2:42.6
C. BLOCK	7.13	108 650	2:59.3	0:44.0	0:15.2	0:00.2	5:58.7
FACADE	7.14	44 184	1:06.8	0:53.7	0:05.6	0:00.5	2:06.6
KINECT	7.15	137 487	3:09.7	1:16.3	0:33.5	0:00.1	4:59.6
BUILDING	7.16	195 154	5:44.3	5:18.3	0:31.3	0:00.2	11:34.1
P. HOUSE	7.17	50 854	2:00.3	0:08.5	0:06.3	0:00.1	2:15.2

Aside from a slower convergence of the HFC refinement we observe larger computing times for the surface classification. This processing step involves the local parameter estimation for each surface region. Therefore, it depends on the number of vertices  $V$ , the number of regions  $L$  as well as the number of surface and relation classes  $S$  and  $R$ .

The total computing time of several minutes for data sets with up to  $V \approx 200\,000$  vertices is satisfactory. Nevertheless, the current implementation – especially of the first processing steps DijkstraFPS and MDL-based HFC – does hold potential for improvements. The use of parallel computing hardware like graphics processing units might be a promising approach to speed up both segmentation algorithms.

This concludes the experimental evaluation on real data sets. Most findings from the previous section are confirmed. Some data artefacts, however, as they only appear with real triangulations of complex surfaces, lead to unexpected results and increased computing times.

For many real-world data sets the point uncertainty  $\sigma$  needed to be chosen significantly larger than the sensor noise in order to take into account the *object noise*. The latter involves all deviations from geometrical primitives that we are not able or not willing to reconstruct. The reconstructable level of detail depends on the point density – or more specifically: on the number of points per surface region. Underestimating the noise and thus increasing the level of detail not only causes problems at poorly sampled areas. Often it affects surrounding, well sampled regions as well. Therefore, we need to choose a rather large uncertainty  $\sigma$  for data sets that do not fully comply with our model for man-made surface structures.

This chapter demonstrated the proposed reconstruction framework on various types of data sets and analyzed the performance w.r.t. prespecified criteria on both synthetic and real surface meshes. We conclude this thesis with a summary and an outlook on future work.

# CHAPTER 8

---

## Conclusion

---

This work proposed a reconstruction framework for man-made surfaces. Industrial production control and building reconstruction for 3D city models are possible application areas.

Given a noisy triangulated point cloud, the reconstruction procedure determines the underlying 3D structure. This includes a partitioning into planar, quadratic and freeform regions. The proposed two-step segmentation approach is free of control parameters and mainly data-driven based on the MDL model selection strategy. To infer the most probable configuration of watertightly connected surface types and inter-regional relations, we represent the preliminary surface structure as graphical model. We incorporate data-driven likelihood factors as well as model-driven parameter distributions and prior probabilities. Constraints induced by the global classification guide a final estimation procedure to obtain an accurate surface parametrization.

---

<b>8.1 Summary</b>	<b>137</b>
<b>8.2 Outlook</b>	<b>139</b>

---

## 8.1 Summary

Goal of this thesis was to reconstruct surface structures of man-made objects given a triangulated point cloud with a corresponding uncertainty information. In the following we will summarize the proposed surface model, the proposed reconstruction framework as well as the experimental evaluation.

The proposed framework is designed for reconstructing man-made surfaces. These are defined as a compound of geometric primitives. We introduced eight possible primitives: three types of planes, four quadrics and a non-parametric freeform surface. Furthermore, we defined five inter-regional relations with corresponding constraints on the two adjacent surface parametrizations. These

constraints guide the surface classification as well as the subsequent parameter estimation.

The reconstruction procedure consists of three processing steps. DijkstraFPS (Chapter 4), a data-driven low-level pre-segmentation based on farthest point sampling and a curvature-adaptive distance transform with Dijkstra's algorithm, yields a fast segmentation into planar regions. Using an incremental-decremental strategy we obtain robust results even on noisy surface meshes. Given an estimate of the underlying data noise, the algorithm automatically stops at an optimal number of regions without the need for any control parameters.

The pre-segmentation is refined within a mid-level hierarchical face clustering (HFC, Chapter 5) based on the MDL model selection strategy. It considers different types of primitives, which are fitted into the data points. By hierarchically merging adjacent regions the overall description length is minimized. Thus, it yields an optimal trade-off between approximation error and model complexity according to an assumed point uncertainty. Additionally we introduced a diffusion operation for relabeling boundary vertices, which significantly improves the resulting segmentation boundaries.

Finally, a global classification using a graphical model representation yields the most probable configuration of surfaces and inter-surface relations (Chapter 6). The latter induce constraints on a joint parameter estimation. This model-driven high-level processing step consistently combines the data points with model knowledge about man-made surfaces: Factors of the graphical model involve the likelihood, knowledge about surface parameter distributions as well as prior probabilities of surface and relation classes. It leads to a simplification of the real, data-driven surface model towards an ideal parametrization in accordance with the classification result.

We analyzed the performance of the proposed reconstruction method based on several previously defined evaluation criteria in Chapter 7. Besides correctness and accuracy of the surface structure we demanded insensibility w.r.t. noise and outliers, clear semantics of involved control parameters, a reasonable computational complexity and flexibility w.r.t. different acquisition methods and scenarios. While most quantitative evaluations were performed on synthetically generated meshes, we processed several real data sets of different type and scope as well. We achieved the following results:

- The experiments showed that our reconstruction method succeeds with up to 10 % Gaussian noise w.r.t. the size of each primitive, 15 % outliers and with surfaces violating the assumed model to a certain extend. The obtained accuracies correspond to the theoretical expectations.
- Sampling density in terms of local feature size is essential for most triangulation algorithms, which are required for preprocessing unordered point clouds, but turned out to play a secondary role for reconstructing the surface structure.
- The effect of control parameters is semantically clear and reflects the results obtained via statistical error propagation. For realistic amounts of Gaussian noise the reconstruction is insensitive w.r.t. extending the set of surface classes. For complex meshes, however, it is advisable to match the

possible classes to the ones expected to be present in the respective data set.

- The computational complexity is linearithmic  $O(V \log V)$  with  $V$  being the number of vertices. For meshes that do not well comply with the model for man-made surfaces, the DijkstraFPS pre-segmentation might yield poor results with some missing region boundaries, which is corrected via vertex diffusion within the MDL-based HFC refinement. This extra effort can drastically slow down the convergence or cause oscillations.
- Experiments on popular CAD meshes, terrestrial multi-view stereo reconstructions, structured-light reconstructions of a small desktop scene as well as terrestrial and desktop laser scans demonstrate the flexibility of the proposed approach w.r.t. different acquisition techniques. Difficulties occur where the topology is not clearly defined by the given triangulation, e.g. due to very sparse sampling, blurred object edges and large proportions of non-parametric freeform surface regions.

Thus, the proposed reconstruction framework is an accurate, robust and computationally tractable approach to determine an underlying surface structure from triangulated point clouds of man-made objects. Nevertheless, there is potential for improvement, which we will briefly outline in the following section.

## 8.2 Outlook

Although the proposed method shows promising results throughout our experiments, several aspects give rise for future work. After collecting minor, rather trivial suggestions for improvement, we add more profound initiations for further research.

The list of surfaces and inter-surface relations proposed in this thesis is rather limited. It can be easily extended with additional primitive types like other types of quadrics or tori. Additional combinations of adjacent surfaces and their corresponding relation are possible as well and are to be chosen depending on the respective application. Common relations not yet defined within this work might be a plane *tangent* to a cylinder or a sphere *touching* the ground plane. When giving up the requirement of adjacent regions to be watertightly connected, one might introduce *parallel* related planes, e.g. for indented facade elements.

The defined primitive types require direct solutions for determining initial parameters as well as a functional model for a possibly iterative parameter estimation. The direct solutions for cylinders and cones yield unstable results for flat surface regions representing small sections of the rotational primitive. An alternative solution based on determining a graph surface within a local tangent coordinate system might be used in such cases.

For all experiments on real data we manually specified the noise level to control the level of detail depending on the expected coordinate accuracy. This control parameter could be automatically determined from the given point cloud by robustly estimating the average residual w.r.t. local tangent planes.

The computational complexity is shown to be linearithmic  $O(V \log V)$  with the number of data points  $V$ , but could be further reduced using more efficient

implementations for the individual reconstruction steps. There are, e.g., fast and even parallel versions of distance transforms, which would significantly speed up the pre-segmentation process. Incremental parameter updates could accelerate the merge and diffusion operations during the hierarchical face clustering (HFC). Last but not least, large meshes could be split into multiple blocks to reduce the required computing time and memory consumption. For the application of city models this could be guided by a prior large-scale building detection.

So far we assumed a surface mesh to be given. Alternatively, we generated the triangulation ourselves with common algorithms within a preprocessing step. A more advanced approach might be to integrate the triangulation with the determination of the surface structure. For, e.g., the Poisson surface reconstruction one might keep a correspondence between original 3D points and the resulting mesh vertices. While the segmentation is performed on the triangular mesh, which only approximates the original points, the classification and estimation could incorporate the original points.

For the relatively small data sets with up to  $V \approx 200\,000$  data points used in this thesis we specified all control parameters based on manual inspections of the given surface mesh and knowledge about the acquired object. For reconstructing many large data sets of similar type, e.g. in the field of 3D city models, one might learn these parameters from annotated data. Existing city models could serve as training data and newly captured test areas were automatically reconstructed.

Another approach to improve the applicability is to combine the proposed framework with higher-level application-specific semantics, like door and window detection or grammars for roof structures. The detection of repetitive structures and verifying the topological consistency of detected inter-regional constraints would further stabilize the result.

Altogether, the proposed reconstruction framework is a powerful strategy for recognizing geometric structures in 3D point clouds of man-made surfaces. Several aspects like improving computing time and scalability as well as extending the provided surface model reveal potential for future research.



# APPENDIX A

---

## Gauss-Helmert model for estimating surface primitives

---

In Tab. 3.1 we defined eight types of surface primitives that are used for fitting primitives in Chapters 5 and 6. This appendix collects relevant formulas for deriving primitive parameters given a set of 3D points. First it presents direct solutions for approximate initializations. Furthermore, it yields the derivatives of the functional model, i.e. the incidence constraint between points and surface, w.r.t. observations and unknown parameters used for iteratively updating the estimation within a Gauss-Helmert model.

---

<b>A.1 Initialization . . . . .</b>	<b>141</b>
<b>A.2 Derivatives of the functional model. . . . .</b>	<b>143</b>

---

### A.1 Initialization

This section lists direct solutions for each primitive in Tab. 3.1. The solutions are either based on a singular value decomposition (SVD)

$$UDV^T \stackrel{\text{SVD}}{=} M \tag{A.1}$$

yielding a matrix

$$V = \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots \\ v_{2,1} & v_{2,2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{A.2}$$

or based on the Moore-Penrose pseudoinverse

$$M^+ = (M^T M)^{-1} M^T \tag{A.3}$$

using a matrix  $M$  with one row per vertex  $v = 1, \dots, V$ :

$$M = \begin{bmatrix} \mathbf{m}_1^\top \\ \mathbf{m}_2^\top \\ \vdots \\ \mathbf{m}_v^\top \\ \vdots \\ \mathbf{m}_V^\top \end{bmatrix}. \quad (\text{A.4})$$

### Horizontal plane

$$\theta = \frac{1}{V} \sum_v X_{v,z} \quad (\text{A.5})$$

### Vertical plane

$$\mathbf{m}_v^\top = [X_{v,x} \quad X_{v,y} \quad 1] \quad (\text{A.6})$$

$$UDV^\top \stackrel{\text{SVD}}{=} M \quad (\text{A.7})$$

$$\boldsymbol{\theta} = \frac{1}{\sqrt{v_{1,3} + v_{2,3}}} \begin{bmatrix} v_{1,3} \\ v_{2,3} \\ v_{3,3} \end{bmatrix} \quad (\text{A.8})$$

### General plane

$$\mathbf{m}_v^\top = [X_{v,x} \quad X_{v,y} \quad X_{v,z} \quad 1] \quad (\text{A.9})$$

$$UDV^\top \stackrel{\text{SVD}}{=} M \quad (\text{A.10})$$

$$\boldsymbol{\theta} = \frac{1}{\sqrt{v_{1,4} + v_{2,4} + v_{3,4}}} \begin{bmatrix} v_{1,4} \\ v_{2,4} \\ v_{3,4} \\ v_{4,4} \end{bmatrix} \quad (\text{A.11})$$

### Vertical cylinder

$$\mathbf{m}_v^\top = \left[ -2X_{v,x} \quad -2X_{v,y} \quad 1 \quad \sqrt{X_{v,x}^2 + X_{v,y}^2 + X_{v,z}^2} \right] \quad (\text{A.12})$$

$$UDV^\top \stackrel{\text{SVD}}{=} M \quad (\text{A.13})$$

$$\boldsymbol{\theta} = \frac{1}{v_{4,4}} \begin{bmatrix} v_{1,4} \\ v_{2,4} \\ \sqrt{v_{1,4}^2 + v_{2,4}^2 - v_{3,4}} \end{bmatrix} \quad (\text{A.14})$$

### General cylinder

As a direct solution for the G-CYLINDER we use the minimal five-point solution from Beder and Förstner (2006). The authors formulate algebraic constraints on the general quadric and obtain sixth order polynomials, which are solved using 2D-Bernstein polynomials. The interested reader is referred to the original publication for further details.

### Vertical cone

$$\mathbf{m}_v^\top = [X_{v,z}^2 \quad 2X_{v,x} \quad 2X_{v,y} \quad 2X_{v,z} \quad -1] \quad (\text{A.15})$$

$$b_v = \sqrt{X_{v,x}^2 + X_{v,y}^2} \quad (\text{A.16})$$

$$\mathbf{q} = M^+ \mathbf{b} \quad (\text{A.17})$$

$$\boldsymbol{\theta} = \begin{bmatrix} q_2 \\ q_3 \\ -q_4/q_1 \\ q_1 \end{bmatrix} \quad \text{if } q_1 > 0 \quad \text{and} \quad q_2^2 + q_3^2 - q_5 > 0 \quad (\text{A.18})$$

### Sphere

$$\mathbf{m}_v^\top = [-2X_{v,x} \quad -2X_{v,y} \quad -2X_{v,z} \quad 1 \quad \sqrt{X_{v,x}^2 + X_{v,y}^2 + X_{v,z}^2}] \quad (\text{A.19})$$

$$UDV^\top \stackrel{\text{SVD}}{=} M \quad (\text{A.20})$$

$$\boldsymbol{\theta} = \frac{1}{v_{5,5}} \begin{bmatrix} v_{1,5} \\ v_{2,5} \\ v_{3,5} \\ \sqrt{v_{1,5}^2 + v_{2,5}^2 + v_{3,5}^2} - v_{4,5} \end{bmatrix} \quad (\text{A.21})$$

### Freeform primitive

The parameterless FREEFORM surface does not require any initialization.

## A.2 Derivatives of the functional model

In order to use a Gauss-Helmert model to find parameters  $\boldsymbol{\theta}$  that minimize the sum of squared distances to all points  $\{\mathcal{X}_v\}$  with  $v = 1, \dots, V$ , we need derivatives of the functional model  $\mathbf{g}(\mathbf{y}, \mathbf{p}) = \mathbf{0}$  w.r.t. unknown parameters  $\mathbf{p} = \boldsymbol{\theta}$

$$A = \frac{\partial \mathbf{g}(\mathbf{y}, \mathbf{p})}{\partial \mathbf{p}} = \begin{bmatrix} \mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \vdots \\ \mathbf{a}_v^\top \\ \vdots \\ \mathbf{a}_V^\top \end{bmatrix} \quad (\text{A.22})$$

and w.r.t. observed point coordinates  $\mathbf{y} = [\mathbf{X}_v]$

$$\mathbf{B}^\top = \frac{\partial \mathbf{g}(\mathbf{y}, \mathbf{p})}{\partial \mathbf{y}} = \begin{bmatrix} \mathbf{b}_1^\top & & & & & \\ & \mathbf{b}_2^\top & & & & \\ & & \ddots & & & \\ & & & \mathbf{b}_v^\top & & \\ & & & & \ddots & \\ & & & & & \mathbf{b}_V^\top \end{bmatrix}. \quad (\text{A.23})$$

In the following we will collect these derivatives for each type of surface primitive relevant to this thesis (Tab. 3.1). For a more compact representation we introduce

$$\Delta \mathbf{X}_v = \mathbf{X}_v - \boldsymbol{\theta}_c \quad (\text{A.24})$$

as the coordinate differences between an observed point  $\mathbf{X}_v$  and the primitive center  $\boldsymbol{\theta}_c$  as well as the 3D and 2D radii  $r_v$  and  $r'_v$ , respectively:

$$r_v = \sqrt{\Delta X_{v,x} + \Delta X_{v,y} + \Delta X_{v,z}}, \quad (\text{A.25})$$

$$r'_v = \sqrt{\Delta X_{v,x} + \Delta X_{v,y}}. \quad (\text{A.26})$$

### Horizontal plane

$$a_v = -1 \quad (\text{A.27})$$

$$\mathbf{b}_v^\top = [0 \quad 0 \quad 1] \quad (\text{A.28})$$

### Vertical plane

$$\mathbf{a}_v^\top = [X_{v,x} \quad X_{v,y} \quad -1] \quad (\text{A.29})$$

$$\mathbf{b}_v^\top = [\theta_{n_x} \quad \theta_{n_y} \quad 0] \quad (\text{A.30})$$

### General plane

$$\mathbf{a}_v^\top = [X_{v,x} \quad X_{v,y} \quad X_{v,z} \quad -1] \quad (\text{A.31})$$

$$\mathbf{b}_v^\top = [\theta_{n_x} \quad \theta_{n_y} \quad \theta_{n_z} \quad 0] \quad (\text{A.32})$$

### Vertical cylinder

$$\mathbf{a}_v^\top = \left[ \frac{-\Delta X_{v,x}}{r'_v} \quad \frac{-\Delta X_{v,y}}{r'_v} \quad -1 \right] \quad (\text{A.33})$$

$$\mathbf{b}_v^\top = \left[ \frac{\Delta X_{v,x}}{r'_v} \quad \frac{\Delta X_{v,y}}{r'_v} \right] \quad (\text{A.34})$$

### General cylinder

$$\mathbf{a}_v^\top = \begin{bmatrix} \frac{(\boldsymbol{\theta}_{L_h} \times \mathbf{X}_v + \boldsymbol{\theta}_{L_0}) \times \mathbf{X}_v}{|\boldsymbol{\theta}_{L_h}| |\boldsymbol{\theta}_{L_h} \times \mathbf{X}_v + \boldsymbol{\theta}_{L_0}|} - \frac{|\boldsymbol{\theta}_{L_h} \times \mathbf{X}_v + \boldsymbol{\theta}_{L_0}| \boldsymbol{\theta}_{L_h}}{|\boldsymbol{\theta}_{L_h}^3|} \\ \frac{\boldsymbol{\theta}_{L_h} \times \mathbf{X}_v + \boldsymbol{\theta}_{L_0}}{|\boldsymbol{\theta}_{L_h}| |\boldsymbol{\theta}_{L_h} \times \mathbf{X}_v + \boldsymbol{\theta}_{L_0}|} \\ -1 \end{bmatrix}^\top \quad (\text{A.35})$$

$$\mathbf{b}_v^\top = \left( \frac{(\boldsymbol{\theta}_{L_h} \times \mathbf{X}_v + \boldsymbol{\theta}_{L_0}) \times \boldsymbol{\theta}_{L_h}}{|\boldsymbol{\theta}_{L_h}| |\boldsymbol{\theta}_{L_h} \times \mathbf{X}_v + \boldsymbol{\theta}_{L_0}|} \right)^\top \quad (\text{A.36})$$

### Vertical cone

$$\mathbf{a}_v^\top = \frac{1}{\sqrt{\theta_A + 1}} \begin{bmatrix} \frac{\Delta X_{v,x}}{r_v} & \frac{\Delta X_{v,y}}{r_v} & \text{sgn}(\Delta X_{v,z}) \sqrt{\theta_A} & \frac{\sqrt{\theta_A} |\Delta X_{v,z}| - r'_v}{2(\theta_A + 1)} - \frac{|\Delta X_{v,z}|}{2\sqrt{\theta_A}} \end{bmatrix} \quad (\text{A.37})$$

$$\mathbf{b}_v^\top = \frac{1}{\sqrt{\theta_A + 1}} \begin{bmatrix} \frac{\Delta X_{v,x}}{r_v} & \frac{\Delta X_{v,y}}{r_v} & \text{sgn}(-\Delta X_{v,z}) \sqrt{\theta_A} \end{bmatrix} \quad (\text{A.38})$$

### Sphere

$$\mathbf{a}_v^\top = -\frac{1}{r_v} [\Delta X_{v,x} \quad \Delta X_{v,y} \quad \Delta X_{v,z} \quad r_v] \quad (\text{A.39})$$

$$\mathbf{b}_v^\top = \frac{1}{r_v} [\Delta X_{v,x} \quad \Delta X_{v,y} \quad \Delta X_{v,z}] \quad (\text{A.40})$$

### Freeform primitive

The derivatives of the parameterless FREEFORM surface are empty matrices.



---

## List of Figures

---

1.1	Surface structure of a small object. . . . .	14
2.1	3D shape acquisition techniques . . . . .	26
2.2	Surface normal and principal curvatures . . . . .	30
2.3	Vertex and triangle normals on a small example mesh . . . . .	33
2.4	Vertex curvature of a small example mesh . . . . .	33
2.5	Comparison of two distance transforms . . . . .	34
2.6	Estimated plane through a set of 3D points . . . . .	38
2.7	Model selection on a 2D data set . . . . .	42
2.8	Graphical model and factor graph of a Markov chain . . . . .	46
3.1	Local feature size and $\epsilon$ -sampling . . . . .	53
3.2	Two possible concepts for segmenting a meshed 2D manifold . . . . .	54
3.3	Model for man-made surface structures . . . . .	57
3.4	Three levels of the proposed reconstruction framework . . . . .	62
4.1	Farthest point sampling . . . . .	64
4.2	Intrinsic and extrinsic distances . . . . .	64
4.3	Uniform and curvature-adaptive sampling . . . . .	65
4.4	Surface segmentation with DijkstraFPS . . . . .	67
4.5	Incremental-decremental surface segmentation . . . . .	68
4.6	Incremental-decremental segmentation, automatic stopping criterion . . . . .	70
4.7	Number of regions depending on shape and noise . . . . .	73
4.8	DijkstraFPS surface segmentation with Lloyd iterations . . . . .	74
4.9	Different curvature-adaptive metrics . . . . .	75
4.10	DijkstraFPS with intrinsic and extrinsic distance metrics . . . . .	77
5.1	Merging adjacent triangles (Garland et al., 2001) . . . . .	80
5.2	Hierarchical face clustering . . . . .	81
5.3	MDL-based HFC with and without boundary vertex diffusion . . . . .	84
5.4	MDL-based HFC with and without boundary regularization . . . . .	85
5.5	MDL-based HFC with different classes of planes . . . . .	87
5.6	MDL-based HFC with quadratic surfaces . . . . .	87
5.7	MDL-based HFC with FREEFORM surfaces . . . . .	88
5.8	Combination of DijkstraFPS and MDL-based HFC . . . . .	89
6.1	Factor graph . . . . .	92
6.2	Potential representing the likelihood of the data . . . . .	95

6.3	Parameter distribution for the zenith distance $z$ . . . . .	96
6.4	Parameter distribution for the radius $\theta_r$ . . . . .	96
6.5	Parameter distribution for the opening angle $\alpha$ . . . . .	97
6.6	Parameter distribution for the contradiction angle $\delta$ . . . . .	100
6.7	Impact of rotation and translation on the contradiction angle $\delta$ .	100
6.8	Factor graph with all messages involved . . . . .	102
6.9	Results of the max-sum inference . . . . .	104
6.10	Joint parameter estimation with constraints . . . . .	105
7.1	Reconstruction of the multi-view stereo mesh BRICKS . . . . .	109
7.2	Reconstruction accuracy of synthetically generated meshes . . . .	111
7.3	Sensitivity analysis w.r.t. noise . . . . .	112
7.4	Sensitivity analysis w.r.t. outliers . . . . .	113
7.5	Sensitivity analysis w.r.t. model violation . . . . .	114
7.6	Sensitivity analysis w.r.t. sampling density . . . . .	115
7.7	Triangulation of a sparsely sampled surface . . . . .	115
7.8	Sensitivity analysis w.r.t. control parameters . . . . .	117
7.9	Sensitivity analysis w.r.t. the set of surface classes . . . . .	118
7.10	Computing times for synthetically generated cubes . . . . .	119
7.11	Reconstruction of the DOUBLE TORUS CAD data set . . . . .	122
7.12	Reconstruction of the FAN DISK CAD data set . . . . .	123
7.13	Reconstruction of the multi-view stereo mesh CONCRETE BLOCK	125
7.14	Reconstruction of the multi-view stereo mesh FACADE . . . . .	126
7.15	Reconstruction of the KINECT data set . . . . .	129
7.16	Reconstruction of the BUILDING laser scan . . . . .	130
7.17	Reconstruction of the PAPER HOUSE laser scan . . . . .	132
7.18	Sensitivity analysis with the PAPER HOUSE laser scan . . . . .	133



---

## List of Tables

---

2.1	Three outlier re-weighting schemes . . . . .	41
3.1	Surface primitives for man-made surface structures . . . . .	59
3.2	Surface relations for man-made surface structures . . . . .	61
6.1	Components of unary and ternary factors . . . . .	93
7.1	Default values for model parameters . . . . .	108
7.2	Reconstruction accuracy of a laser scanned BALL . . . . .	120
7.3	Reconstruction accuracy of a laser scanned CONE . . . . .	121
7.4	Computing times for all real data sets . . . . .	135



---

## List of Algorithms

---

2.1	Dijkstra's algorithm . . . . .	35
4.1	Incremental DijkstraFPS surface segmentation . . . . .	66
4.2	Decremental DijkstraFPS surface segmentation . . . . .	68
4.3	Incremental-decremental DijkstraFPS segmentation strategy . . . . .	71
4.4	DijkstraFPS Lloyd iteration . . . . .	74
5.1	Hierarchical face clustering (HFC, Garland et al., 2001) . . . . .	80
5.2	MDL-based hierarchical face clustering . . . . .	84



---

## Bibliography

---

- Aganj, E., R. Keriven, and J.-P. Pons (2009). Photo-consistent surface reconstruction from noisy point clouds. In *IEEE International Conference on Image Processing*, pp. 505–508. 28
- Amenta, N. and M. Bern (1998). Surface reconstruction by voronoi filtering. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pp. 39–48. 28
- Amenta, N. and M. Bern (1999, December). Surface reconstruction by voronoi filtering. *Discrete and Computational Geometry* 22(4), 481–504. 28
- Amenta, N., M. Bern, and D. Eppstein (1998, March). The crust and the  $\beta$ -skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing* 60, 125–135. 28
- Amenta, N., M. Bern, and M. Kamvysselis (1998). A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 415–421. 28
- Amenta, N., S. Choi, T. K. Dey, and N. Leekha (2000). A simple algorithm for homeomorphic surface reconstruction. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*. 28, 114, 115
- Amenta, N., S. Choi, and R. Kolluri (2001). The power crust. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*. 28
- Andres, B., C. Kondermann, U. Köthe, and F. A. Hamprecht (2008). An extensible c++ template library for statistical inference in probabilistic factor graph models with higher order functions. Technical report, University of Heidelberg. 49, 103
- Atkar, P. N., A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi (2005). Hierarchical segmentation of surfaces embedded in  $r^3$  for auto-body painting. In *IEEE International Conference on Robotics and Automation*. 19
- Attali, D. (1997).  $r$ -regular shape reconstruction from unorganized points. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, pp. 248–253. 27
- Attene, M. (2010, November). A lightweight approach to repairing digitized polygon meshes. *The Visual Computer* 26(11), 1393–1406. 17

- Attene, M. and B. Falcidieno (2006). ReMESH: An interactive environment to edit and repair triangle meshes. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications*. 17
- Attene, M., B. Falcidieno, and M. Spagnuolo (2006, March). Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer* 22(3), 181–193. 17
- Attene, M., S. Katz, M. Mortara, G. Patane, M. Spagnuolo, and A. Tal (2006). Mesh segmentation – a comparative study. In *IEEE International Conference on Shape Modeling and Applications*. IEEE. 16
- Attene, M. and G. Patanè (2010). Hierarchical structure recovery of point-sampled surfaces. In *Computer Graphics Forum*, pp. 1905–1920. 17
- Attene, M. and M. Spagnuolo (2000, September). Automatic surface reconstruction from point sets in space. *Computer Graphics Forum* 19(3), 457–465. 27
- Baillard, C. and A. Zisserman (2000). A plane-sweep strategy for the 3d reconstruction of buildings from multiple images. In *XIXth ISPRS Congress - Technical Commission II: Systems for Data Processing, Analysis and Representation*. 21
- Becker, S. (2009, November). Generation and application of rules for quality dependent facade reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing* 64(6), 640–653. 55
- Beder, C. and W. Förstner (2006). Direct solutions for computing cylinders from minimal sets of 3d points. In A. Leonardis, H. Bischof, and A. Pinz (Eds.), *Computer Vision – ECCV 2006*, Volume 3951 of *Lecture Notes in Computer Science*, pp. 135–146. Springer Berlin Heidelberg. 143
- Belton, D. and D. D. Lichti (2006). Classification and segmentation of terrestrial laserscanner point clouds using local variance information. In *Proceedings of the ISPRS Commission V Symposium "Image Engineering and Vision Metrology"*. 75
- Beucher, S. and C. Lantuejoul (1979). Use of watersheds in contour detection. In *Proceedings of the International Workshop on Image Processing, Real-time Edge and Motion Detection*. 19
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. 45, 47, 48
- Boissonnat, J.-D. (1984, October). Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics* 3(4), 266–286. 27
- Boykov, Y. and V. Kolmogorov (2004, September). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26(9), 1124–1137. 18, 20

- Brandenburger, W., M. Drauschke, W. Nguatem, and H. Mayer (2013). Detektion von fensterverdachungen und gesims in fassadenbildern und punktwolken. In *33. Wissenschaftlich-Technische Jahrestagung der DGPF*. 55
- Brenner, C. (2000). *Dreidimensionale Gebäuderekonstruktion aus digitalen Oberflächenmodellen und Grundrissen*. Ph. D. thesis, Fakultät für Bauingenieur- und Vermessungswesen, Universität Stuttgart. 21
- Brenner, C. (2003). Building reconstruction from laser scanning and images. In *Proceedings of the ITC Earth Observation Science Department Workshop on Data Quality in Earth Observation Techniques*. 21
- Brenner, C., N. Haala, and D. Fritsch (2001). Towards fully automated 3d city model generation. In *Proceedings of the third International Workshop on Automatic Extraction of Man-Made Objects from Aerial and Space Images*. 21
- Brenner, C. and N. Ripperda (2006, September). Extraction of facades using rjmc and constraint equations. In W. Förstner and R. Steffen (Eds.), *Photogrammetric Computer Vision*, Volume 36, pp. 155–160. 55
- Brunn, A. (2000). *Semantik-basierte Gebäudeerfassung mit verkoppelten Markoff-Zufallsfeldern*. Ph. D. thesis, Institute of Photogrammetry, University of Bonn. 21
- Cazals, F. and J. Giesen (2004, November). Delaunay triangulation based surface reconstruction: Ideas and algorithms. Technical Report 5393, Institut National De Recherche En Informatique Et En Automatique (INRIA). 29
- Chauve, A., P. Labatut, and J. Pons (2010). Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1261–1268. IEEE. 20
- Chen, X., A. Golovinskiy, and T. Funkhouser (2009, August). A benchmark for 3d mesh segmentation. *ACM Transactions on Graphics* 28(3), 73:1–73:12. 16
- Cignoni, P., C. Montani, and R. Scopigno (1998, February). A comparison of mesh simplification algorithms. *Computers & Graphics* 22(1), 37–54. 16
- Cohen-Steiner, D., P. Alliez, and M. Desbrun (2004, August). Variational shape approximation. *ACM Transactions on Graphics* 23(3), 905–914. 17, 72
- Dey, T. K., X. Ge, Q. Que, I. Safa, L. Wang, and Y. Wang (2013, August). Feature-preserving reconstruction of singular surfaces. *Computer Graphics Forum* 31(5), 1787–1796. 28
- Dey, T. K., J. Giesen, and J. Hudson (2001). Delaunay based shape reconstruction from large data. In *Proceedings of the IEEE 2001 Symposium on Parallel and Large-data Visualization and Graphics*. 28
- Dey, T. K. and S. Goswami (2003). Tight cocone: A water-tight surface reconstructor. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*. 28

- Dey, T. K. and S. Goswami (2006, August). Provable surface reconstruction from noisy samples. *Computational Geometry: Theory and Applications* 35(1), 124–141. 28
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1), 269–271. 17, 34, 66
- Dorninger, P. and C. Nothegger (2007). 3d segmentation of unstructured point clouds for building modelling. In *Photogrammetric Image Analysis*. 21
- Douglas, D. H. and T. K. Peucker (1973, December). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization* 10(2), 112–122. 17
- Edelsbrunner, H. (1998). Shape reconstruction with delaunay complex. In C. L. Lucchesi and A. V. Moura (Eds.), *LATIN'98: Theoretical Informatics*, Volume 1380 of *Lecture Notes in Computer Science*, pp. 119–132. Springer Berlin Heidelberg. 27, 52
- Edelsbrunner, H. and E. P. Mücke (1994, January). Three-dimensional alpha shapes. *ACM Transactions on Graphics* 13(1), 43–72. 27
- Elberink, S. O. (2009). Target graph matching for building reconstruction. In *Laserscanning*. 20
- Elberink, S. O. and G. Vosselman (2009, July). Building reconstruction by target based graph matching on incomplete laser data: Analysis and limitations. *Sensors* 9(8), 6101–6118. 20
- Eldar, Y., M. Lindenbaum, M. Porat, and Y. Y. Zeevi (1997, September). The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing* 6(9), 1305–1315. 16
- Fischler, M. A. and R. C. Bolles (1981, June). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24(6), 381–395. 18
- Förstner, W. (1989). Image analysis techniques for digital photogrammetry. In *Photogrammetrische Woche*, pp. 205–221. 41, 42, 69, 88
- Förstner, W. (2012, June). Minimal representations for testing and estimation in projective spaces. *Photogrammetrie Fernerkundung Geoinformation* 2012(3), 209–220. 38, 99
- Förstner, W. and B. Wrobel (2004). Mathematical concepts in photogrammetry. In J. McGlone, E. Mikhail, J. Bethel, and R. Mullen (Eds.), *Manual of Photogrammetry* (5th ed.), Chapter 2, pp. 15–180. American Society for Photogrammetry and Remote Sensing. 36, 37, 40, 58
- Frank, A. U. and W. Kuhn (1986). Cell graphs : A provable correct method for the storage of geometry. In *2nd International Symposium on Spatial Data Handling (SDH)*, pp. 411–436. 20



- Frey, B. J. and D. J. C. MacKay (1998). A revolution: Belief propagation in graphs with cycles. *Advances in Neural Information Processing Systems 10*, 479–485. 45, 101
- Furukawa, Y. and J. Ponce (2010, August). Accurate, dense, and robust multi-view stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*(8), 1362–1376. 108, 124
- Gallup, D., J.-M. Frahm, and M. Pollefeys (2010). Piecewise planar and non-planar stereo for urban scene reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*. 18, 55
- Garland, M. (1999a). Multiresolution modeling: Survey & future opportunities. In *Eurographics, State of the Art Report*. 16
- Garland, M. (1999b). *Quadric-based Polygonal Surface Simplification*. Ph. D. thesis, School of Computer Science, Carnegie Mellon University. 17
- Garland, M. and P. S. Heckbert (1997). Surface simplification using quadric error metrics. In *SIGGRAPH '97 Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 17, 131
- Garland, M., A. Willmott, and P. S. Heckbert (2001, March). Hierarchical face clustering on polygonal surfaces. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 49–58. 17, 76, 79, 80, 81, 82, 147, 151
- Georgeff, M. P. and C. S. Wallace (1984). A general selection criterion for inductive inference. In *European Conference on Artificial Intelligence*, pp. 219–228. 41
- Glanvill, M. and K. Broughan (1997). Curve and surface reconstruction in  $r^2$  and  $r^3$ . In *High Performance Computing on the Information Superhighway*, pp. 395–400. 27
- Goldman, R. (2005, October). Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design 22*(7), 632–658. 29, 30
- Gool, L. V., G. Zeng, F. V. den Borre, and P. Müller (2007). Towards mass-produced building models. In U. Stilla (Ed.), *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 209–220. 55
- Hoppe, H., T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle (1992). Surface reconstruction from unorganized points. In *Proceedings of the nineteenth Annual Conference on Computer Graphics and Interactive Techniques*, pp. 71–78. 29
- Huang, H., C. Brenner, and M. Sester (2013, May). A generative statistical approach to automatic 3d building roof reconstruction from laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing 79*, 29–43. 20
- Jancosek, M. and T. Pajdla (2011). Hallucination-free multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*. 28

- Kang, S.-J., H.-H. Trinh, D.-N. Kim, and K.-H. Jo (2010). Entrance detection of buildings using multiple cues. In *Intelligent Information and Database Systems*, Volume 5990 of *Lecture Notes in Computer Science*, pp. 251–260. Springer Berlin Heidelberg. 55
- Kazhdan, M., M. Bolitho, and H. Hoppe (2006). Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, Aire-la-Ville, Switzerland, Switzerland, pp. 61–70. Eurographics Association. 29, 108
- Kazhdan, M. and H. Hoppe (2013, June). Screened poisson surface reconstruction. *ACM Transactions on Graphics* 32(3), 29:1–29:13. 29, 124
- Kimmel, R. and J. A. Sethian (1998, July). Computing geodesic paths on manifolds. In *Proceedings of the National Academy of Sciences*, Volume 95, pp. 8431–8435. 35
- Kindermann, R. and J. L. Snell (1980). *Markov random fields and their applications*, Volume 1 of *Contemporary Mathematics*. American Mathematical Society. 45
- Koch, K.-R. (1997). *Parameterschätzung und Hypothesentests* (3 ed.). Dümmler. 36, 37, 94
- Kolbe, T. H. (2001). *Identifikation und Rekonstruktion von Gebäuden in Luftbildern mittels unscharfer Constraints*. Ph. D. thesis, Institut für Umweltwissenschaften der Hochschule Vechta. 21
- Kolluri, R., J. R. Shewchuk, and J. F. O'Brien (2004). Spectral surface reconstruction from noisy point clouds. In *Eurographics Symposium on Geometry Processing*. 28
- Kschischang, F. R., B. J. Frey, and H.-A. Loeliger (2001, February). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 47(2), 498–519. 44
- Labatut, P. (2009). *Labeling of data-driven complexes for surface reconstruction*. Ph. D. thesis, Ecole Doctorale de Sciences Mathématiques de Paris Centre, Laboratoire IMAGINE. 20
- Labatut, P., J.-P. Pons, and R. Keriven (2007). Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *IEEE International Conference on Computer Vision*. 28
- Labatut, P., J.-P. Pons, and R. Keriven (2009, September). Hierarchical shape-based surface reconstruction for dense multi-view stereo. In *IEEE International Workshop on 3-D Digital Imaging and Modeling*, pp. 1598–1605. IEEE. 20
- Lafarge, F., R. Keriven, and M. Brédif (2010, July). Insertion of 3-d-primitives in mesh-based representations: towards compact models preserving the details. *IEEE Transactions on Image Processing* 19(7), 1683–1694. 18
- Lafarge, F., R. Keriven, M. Brédif, and H.-H. Vu (2012, January). A hybrid multi-view stereo algorithm for modeling urban scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(1), 5–17. 18

- Lafarge, F. and C. Mallet (2012). Creating large-scale city models from 3d-point clouds: A robust approach with hybrid representation. *International Journal of Computer Vision* 99(1), 69–85. 21
- Lai, Y.-K., S.-M. Hu, R. R. Martin, and P. L. Rosin (2008). Fast mesh segmentation using random walks. In *Solid and physical modeling*, pp. 183–191. 16
- Lari, Z., A. Habib, and E. Kwak (2011). An adaptive approach for segmentation of 3d laser point cloud. In *ISPRS Workshop Laser Scanning 2011, Calgary, Canada*. 21
- Leclerc, Y. G. (1989, May). Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision* 3(1), 73–102. 85
- Leonardis, A., A. Gupta, and R. Bajcsy (1995, April). Segmentation of range images as the search for geometric parametric models. *International Journal of Computer Vision* 14(3), 253–277. 41
- Li, Y., X. Wu, Y. Chrysanthou, A. Sharf, D. Cohen-Or, and N. J. Mitra (2011, July). Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics* 30(4), 52:1–52:11. 18
- Lloyd, S. P. (1982, March). Least squares quantization in pcm. *IEEE Transactions on Information Theory* 28(2), 129–137. 72
- Loch-Dehbi, S. and L. Plümer (2009). Geometric reasoning in 3d building models using multivariate polynomials and characteristic sets. In *Proceedings of the ISPRS WG II/2+3+4 and Cost Workshop*. 104
- Loch-Dehbi, S. and L. Plümer (2011). Automatic reasoning for geometric constraints in 3d city models with uncertain observations. *ISPRS Journal of Photogrammetry and Remote Sensing* 66(2), 177–187. 104
- Lorensen, W. E. and H. E. Cline (1987, July). Marching cubes: A high resolution 3d surface reconstruction algorithm. *Computer Graphics* 21(4), 163–169. 28
- Mada, S. K., M. L. Smith, L. N. Smith, and P. S. Midha (2003). Overview of passive and active vision techniques for hand-held 3d data acquisition. In *Proceedings of the Society of Photo-Optical Instrumentation*, Volume 4877, pp. 16–27. 26
- Mahalanobis, P. C. (1936, January). On the generalised distance in statistics. *Proceedings of the National Institute of Sciences of India* 2(1), 49–55. 99
- Mangan, A. P. and R. T. Whitaker (1999). Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics* 5(4), 308–321. 19
- Meidow, J., C. Beder, and W. Förstner (2009, March). Reasoning with uncertain points, straight lines, and straight line segments in 2d. *ISPRS Journal of Photogrammetry and Remote Sensing* 64(2), 125–139. 38
- Meine, H. and U. Köthe (2005). Image segmentation with the exact watershed transform. In *Visualization, Imaging, and Image Processing*. 19

- Memoli, F. and G. Sapiro (2001). Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces. *Journal of Computational Physics* 173, 730–764. 16
- Memoli, F. and G. Sapiro (2002, December). Distance functions and geodesics on points clouds. Technical Report 2002/12, Digital Technology Center, University of Minnesota. 16
- Milde, J. and C. Brenner (2009). Graph-based modeling of building roofs. In *AGILE Conference on GIScience*. 20
- Milde, J., Y. Zhang, C. Brenner, L. Plümer, and M. Sester (2008). Building reconstruction using a structural description based on a formal grammar. In *XXIst ISPRS Congress Technical Commission III*. 20
- Minka, T. P. (2001). *A family of algorithms for approximate Bayesian inference*. Ph. D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. 94, 98
- Moening, C. and N. A. Dodgson (2003a). Fast marching farthest point sampling. In *Eurographics*. 16, 64, 66
- Moening, C. and N. A. Dodgson (2003b, May). Fast marching farthest point sampling for point clouds and implicit surfaces. Technical Report 565, University of Cambridge, Computer Laboratory. 16, 64, 66
- Müller, P., G. Zeng, P. Wonka, and L. V. Gool (2007, July). Image-based procedural modeling of facades. *ACM Transactions on Graphics* 26(3), 85:1–85:10. 55
- Ohtake, Y., A. Belyaev, and A. Pasko (2001). Dynamic meshes for accurate polygonization of implicit surfaces with sharp features. In *Proceedings International Conference on Shape Modeling and Applications*, pp. 74–81. IEEE Computer Society. 29
- O’Rourke, J., H. Booth, and R. Washington (1987, August). Connect-the-dots: A new heuristic. *Computer Vision, Graphics, and Image Processing* 39(2), 258–266. 27
- Page, D. L. (2003, May). *Part Decomposition of 3D Surfaces*. Ph. D. thesis, The University of Tennessee, Knoxville. 19
- Page, D. L., A. F. Koschan, and M. A. Abidi (2003, June). Perception-based 3d triangle mesh segmentation using fast marching watersheds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 27–32. 19
- Pan, H.-P. (1994, April). Two-level global optimization for image segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing* 49(2), 21–32. 85
- Pan, H.-P. and W. Förstner (1994). Generalization of linear patterns based on mdl criterion. Technical report, Institute of Photogrammetry, University of Bonn. 17

- Pan, Q., G. Reitmayr, and T. Drummond (2009, September). Proforma: Probabilistic feature-based on-line rapid model acquisition. In *British Machine Vision Conference*. 28
- Pearl, J. (1982). Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the American Association of Artificial Intelligence National Conference on AI*. 45
- Persson, P.-O. and G. Strang (2004, June). A simple mesh generator in matlab. *SIAM Review* 46(2), 329–345. 114
- Peternell, M. and T. Steiner (2004). Reconstruction of piecewise planar objects from point clouds. *Computer-Aided Design* 36(4), 333–342. 21
- Peyré, G. and L. Cohen (2004, September). Surface segmentation using geodesic centroidal tessellation. In *Second International Symposium on 3D Data Processing, Visualization and Transmission*, pp. 995–1002. 64, 72
- Peyré, G. and L. Cohen (2006). Geodesic remeshing using front propagation. *International Journal of Computer Vision* 69, 145–156. 64, 72
- Pu, S. and G. Vosselman (2006). Automatic extraction of building features from terrestrial laser scanning. In *Proceedings of the ISPRS Commission V Symposium "Image Engineering and Vision Metrology"*. 21
- Pu, S. and G. Vosselman (2007). Extracting windows from terrestrial laser scanning. In *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*. 55
- Pu, S. and G. Vosselman (2009a, June). Building facade reconstruction by fusing terrestrial laser points and images. *Sensors* 9(6), 4525–4542. 20
- Pu, S. and G. Vosselman (2009b). Knowledge based reconstruction of building models from terrestrial laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing* 64, 575–584. 20
- Pulla, S., A. Razdan, and G. Farin (2001, April). Improved curvature estimation for watershed segmentation of 3-dimensional meshes. Manuscript. 19, 29, 30
- Ramer, U. (1972, November). An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing* 1(3), 244–256. 17
- Reem, D. (2011, June). The geometric stability of voronoi diagrams with respect to small changes of the sites. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pp. 254–263. 68
- Ripperda, N. (2008, July). Determination of facade attributes for facade reconstruction. In *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 285–290. 55
- Rottensteiner, F. (2010). Roof plane segmentation by combining multiple images and point clouds. In *Proceedings of Photogrammetric Computer Vision and Image Analysis Conference*. 21

- Rottensteiner, F., G. Sohn, J. Jung, M. Gerke, C. Baillard, S. Benitez, and U. Breitkopf (2012). The isprs benchmark on urban object classification and 3d building reconstruction. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. 21
- Schindler, F. and W. Förstner (2011). Fast marching for robust surface segmentation. In U. Stilla, F. Rottensteiner, H. Mayer, B. Jutzi, and M. Butenuth (Eds.), *Photogrammetric Image Analysis*, Volume 6952 of *Lecture Notes in Computer Science*, pp. 147–158. Springer Berlin Heidelberg. 15, 17, 66, 67, 72
- Schindler, F. and W. Förstner (2013, August). DijkstraFPS: Graph partitioning in geometry and image processing. *Zeitschrift für Photogrammetrie, Fernerkundung und Geoinformation 2013(4)*, 285–296. 15, 17, 35
- Schindler, F., W. Förstner, and J.-M. Frahm (2011). Classification and reconstruction of surfaces from point clouds of man-made objects. In *IEEE International Conference on Computer Vision, Workshop on Computer Vision for Remote Sensing of the Environment*. 15
- Schnabel, R., P. Degener, and R. Klein (2009, March). Completion and reconstruction with primitive shapes. *Computer Graphics Forum 28(2)*, 503–512. 18
- Schnabel, R., R. Wahl, and R. Klein (2007, June). Efficient ransac for point-cloud shape detection. *Computer Graphics Forum 26(2)*, 214–226. 18
- Sedlacek, D. and J. Zara (2009). Graph cut based point-cloud segmentation for polygonal reconstruction. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, Y. Kuno, J. Wang, R. Pajarola, P. Lindstrom, A. Hinkenjann, M. L. Encarnacao, C. T. Silva, and D. Coming (Eds.), *Advances in Visual Computing*, Volume 5876 of *Lecture Notes in Computer Science*, pp. 218–227. Springer Berlin Heidelberg. 55
- Seong, J.-K., W.-K. Jeong, and E. Cohen (2008, June). Anisotropic geodesic distance computation for parametric surfaces. In *Shape Modeling and Applications*. 75
- Sethian, J. A. (1996, February). A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Sciences*, Volume 93, pp. 1591–1595. 16, 34, 66
- Snively, N., S. M. Seitz, and R. Szeliski (2006, July). Photo tourism: Exploring image collections in 3d. *ACM Transactions on Graphics 25(3)*, 835–846. 108, 124
- Spanier, E. H. (1994). *Algebraic Topology* (reprint ed.). Mathematics subject classifications. Springer. 27
- Strubecker, K. (1969). *Theorie der Flächenkrümmung*. Number 3 in Differential-geometrie. De Gruyter. 30
- Tcherniavski, L. and P. Stelldinger (2008). A thinning algorithm for topologically correct 3d surface reconstruction. In *International Conference on Visualization, Imaging and Image Processing*, pp. 119–124. 28, 53

- Tian, Y., M. Gerke, G. Vosselman, and Q. Zhu (2010). Knowledge-based building reconstruction from terrestrial video sequences. *ISPRS Journal of Photogrammetry and Remote Sensing* 65, 395–408. 20
- Tian, Y., Q. Zhu, M. Gerke, and G. Vosselman (2009). Knowledge-based topological reconstruction for building facade surface patches. In *Proceedings of the 3rd ISPRS International Workshop 3D-ARCH 2009: "3D Virtual Reconstruction and Visualization of Complex Architectures"*. 20
- Tishchenko, I. (2010). Surface reconstruction from point clouds. Bachelor's thesis, Autonomous Systems Lab, ETH Zürich. 29
- Tuttas, S. and U. Stilla (2013). Rekonstruktion von fenstern aus schrägsicht-als punktwolken zur anreicherung von gebäudemodellen. In *33. Wissenschaftlich-Technische Jahrestagung der DGPF*. 55
- Weber, O., Y. S. Devir, A. M. Bronstein, M. M. Bronstein, and R. Kimmel (2008, October). Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Transactions on Graphics* 27(4), 104:1–16. 34
- Weidner, U. and W. Förstner (1995). Towards automatic building extraction from high-resolution digital elevation models. *ISPRS Journal of Photogrammetry and Remote Sensing* 50(4), 38–49. 21
- Werner, T. and A. Zisserman (2002a). Model selection for automated architectural reconstruction from multiple views. In *British Machine Vision Conference*. 55
- Werner, T. and A. Zisserman (2002b, May). New techniques for automated architectural reconstruction from photographs. In *European Conference on Computer Vision*, pp. 541–555. 21
- Wilke, W. (2002). *Segmentierung und Approximation großer Punktwolken*. Ph. D. thesis, Technische Universität Darmstadt. 16, 29
- Xu, S.-G., Y.-X. Zhang, and J.-H. Yong (2010, February). A fast sweeping method for computing geodesics on triangular manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(2), 231–241. 34
- Yang, M. Y. and W. Förstner (2010, January). Plane detection in point cloud data. Technical Report 1, Department of Photogrammetry, University of Bonn, Bonn. 41
- Zhou, Q.-Y. and U. Neumann (2010). 2.5d dual contouring: A robust approach to creating building models from aerial lidar. In *European Conference on Computer Vision*. 21
- Zhou, Q.-Y. and U. Neumann (2011). 2.5d building modeling with topology control. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2489–2496. 21
- Zhou, Q.-Y. and U. Neumann (2012). 2.5d building modeling by discovering global regularities. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 326–333. 21