

Abstract of thesis entitled

Optimization and Differential Geometry for Geometric Modeling

Submitted by

LIU Yang

for the degree of Doctor of Philosophy
at The University of Hong Kong
in September 2008

Nowadays “Geometry Modeling and Computer Graphics” has grown into a cross-disciplinary research area. It has shown its powerful applicability in computer aided design, scientific visualization, computer animation and other fields. One of important and interesting topics is “finding optimal geometric structures and shapes” which covers curve and surface approximation, shape reconstruction and shape modeling by polyhedral meshes, mesh generation, etc... , and mainly solved by geometrical and optimization techniques.

This thesis first investigates the problem of orthogonal distance fitting of a point cloud by parametric curves and surfaces. The fitting problem is considered as an optimization problem and the geometric aspects in the traditional Newton-like and Gauss-Newton methods are studied. The study shows how the better geometric understanding benefits the development of fast optimization algorithms. Based on the consideration of local geometric properties, several existing methods(PDM, TDM) are classified and novel methods(SDM, GTDM, CDM) are proposed. The demonstrated super-linear convergence and good performance of novel methods are very useful and efficient in parametric curve and surface fitting. Moreover, the geometric optimization analysis is successfully applied in constrained 3D shape reconstruction by combining enhanced fitting and registration techniques.

The second part of this thesis studies a well-known technique—“Centroidal Voronoi Tessellation”(CVT) from the numerical optimization point of view. A nice property is revealed and proved that the 2D/3D CVT function in a convex domain is actually C^2 . Based on this property, Newton-like or Quasi-Newton methods are applied to speed up CVT computation and achieve superlinear convergence comparing to the classical Lloyd’s method. Similar analysis and algorithms are also applied in constrained/restricted Centroidal Voronoi Tessellation for triangular meshes.

The geometric and aesthetical demands from architecture motivate the study of geometric modeling and processing. In architectural freeform design, the relation between

shape and fabrication poses new challenges and requires more sophistication from the underlying geometry. The last part of this thesis proposes new concepts: conical meshes, circular meshes and edge-parallel meshes in the context of parallel meshes. These new types of meshes provide a novel approach to approximate and model geometric shape with nice face/vertex/edge-offset properties and orthogonal support structure for fabrication. The geometric realization of these concepts involves quadrilateral meshing by principle curvature networks, face planarization of meshes by numerical optimization. Also mesh parallelism is the main ingredient in a novel discrete theory of curvatures, the proposed methods in this thesis are applied to the construction of quadrilateral, pentagonal and hexagonal meshes, discrete minimal surfaces, discrete constant mean curvature surfaces and their geometric transforms.

The methods and concepts introduced in this thesis are essential and effective. The unified parametric curve and surface fitting presents a clear way to understand geometric meaning of optimization techniques and develop fast algorithms. The fast computation of CVT has a good potential to replace Lloyd's method in geometry modeling and meshing generation. The proposed methods and concepts of parallel meshes are powerful tools for designing geometric shape in architecture applications.

[499 Words]

A handwritten signature in blue ink, reading "Lam Tang". The signature is written in a cursive style with a long, sweeping underline that extends to the right.

Optimization and Differential Geometry for Geometric Modeling

by

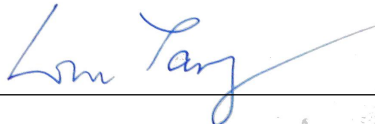
LIU Yang

A thesis submitted for
the Degree of Doctor of Philosophy
at The University of Hong Kong.

September 2008

Declaration

I declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree, diploma or other qualifications.

Signed:  _____

Date: 2008-09-07 _____

“Mighty is geometry; joined with art; resistless.”

Euripides (485-406 B.C)

Acknowledgements

It is my pleasure to acknowledge the people who made this thesis possible.

First I would like to thank my PhD supervisor Prof. Wenping Wang for his great guidance and continuous support on my research. His enthusiasm for research inspires me to work hard to achieve better results.

Part of my work is collaborated with Prof. Helmut Pottmann and Prof. Johannes Wallner. I wish to thank them for their excellent guidance in geometry and kind help when I was in Vienna.

Special thanks go to my Master supervisors Prof. Falai Chen and Prof. Yuyu Feng, who led me to computational mathematics and computer graphics fields, and support my research always.

I am very grateful to all my friends from HKU, USTC and colleagues at TU Wien. Without them, I could not enjoy the joyful life during these years. My thanks go to Bin Chan, Yi-King Choi, Cheng Kin Shing Dominic, Huaiping Yang, Kelvin Lee, Dayue Zheng, Qi Su, Chen Liang, Xi Luo, Dongming Yan, Pengbo Bo, Lin Lu, Hui Zhang, Feng Sun, Yufei Li, Ruotian Ling, Li Cao, Liyong Shen, Zuoqin Wang, Martin Peternell, Niloy Mitra, Heinz Schmiedhofer.

Finally I would like to dedicate this thesis to my parents and my wife for their love, never-ending support and long-time waiting.

Contents

Declaration	iii
Acknowledgements	v
List of Figures	xi
1 Introduction	1
1.1 Motivation	1
1.2 Outline	3
2 Fitting B-spline Curves to Point Clouds by SDM	5
2.1 Problem Formulation	5
2.2 Related Work	6
2.2.1 Spline curve fitting techniques	6
2.2.2 Second order approximation to squared distance function	10
2.3 Fitting a B-spline Curve to a Point Cloud Using SDM	11
2.3.1 A new quadratic approximation to the squared distance	12
2.3.2 Main steps of SDM	13
2.4 Experiments and Comparison	14
2.5 Implementation Issues	19
2.5.1 Initialization and adjustment of control points	19
2.5.2 Fast setup of error terms	20
2.5.3 Fitting an open B-spline curve	21
2.6 Discussion from Viewpoint of Optimization	24
2.6.1 PDM as an alternating method	25
2.6.2 TDM – Gauss-Newton iteration and its variants	26
2.6.3 SDM – a quasi-Newton method	29
2.6.4 Step size control	32
2.7 Concluding Remarks	33
3 Constrained 3D Shape Reconstruction	35
3.1 Introduction	35
3.1.1 Previous work	36
3.1.2 Contributions	36
3.2 Fundamentals of SDM	37

3.2.1	Squared distance function of a surface	37
3.2.2	Registration using SDM	38
3.2.3	SDM for B-spline surface fitting	40
3.2.4	TDM and PDM	41
3.3	Combination of Surface Fitting and Registration	42
3.4	Applications	44
3.4.1	Surfaces of revolution	44
3.4.2	Spiral surfaces	46
3.4.3	Constrained 3D shape reconstruction	48
3.4.3.1	Constrained fitting to a single set of data points	48
3.4.3.2	Constrained fitting to multiple views	49
3.4.4	Remarks on the implementation	49
3.5	Conclusions	54
4	Least Squares Orthogonal Distance Fitting of Parametric Curves and Surfaces	55
4.1	Introduction	55
4.2	Preliminary	56
4.2.1	Notation	56
4.2.2	Nonlinear least squares	57
4.2.3	Principal directions and curvatures of parametric curves and surfaces	58
4.3	Orthogonal Distance Fitting	58
4.3.1	Distance-based Gauss-Newton method	59
4.3.2	Coordinate-based Gauss-Newton method	61
4.3.3	SDM - modified Hessian approximation	65
4.3.4	Comparisons	65
4.4	Numerical Experiments	67
4.5	Conclusions	68
5	Computing Centroidal Voronoi Tessellation with Superlinear Convergence	71
5.1	Introduction	71
5.1.1	Problem setting and previous work	71
5.1.2	The variational point of view	73
5.2	Continuity Analysis	76
5.2.1	Variational formulation	76
5.2.2	Smoothness of F	77
5.2.3	Experimental evidence of continuity	77
5.3	Numerical Optimization	78
5.3.1	Numerical examples	80
5.4	CVT on Polyhedral Surfaces	83
5.4.1	Constrained and restricted CVT	83
5.4.2	C^2 smoothness	84
5.4.3	Implementation	85
5.4.4	Experiments	87
5.5	Appendix: Proof of Theorem 5.2	88

6	Geometric Modeling with Conical Meshes and Developable Surfaces	95
6.1	Introduction	95
6.1.1	Previous work	95
6.1.2	Contributions and overview	98
6.2	PQ Meshes and PQ Perturbation	99
6.2.1	PQ meshes	100
6.2.2	PQ perturbation	101
6.3	Subdividing Developables and PQ Meshes	105
6.4	Conical Meshes	105
6.5	Computing Conical Meshes	111
6.6	Results and Discussion	113
6.7	Conclusion and Future Work	116
6.8	Appendix: An Angle Criterion for Conical Mesh Vertices	117
6.8.1	Conical vertices	117
6.8.2	Convex spherical quadrilaterals	119
6.8.3	General spherical quadrilaterals	121
7	Geometry of Multi-layer Freeform Structures for Architecture	129
7.1	Introduction	129
7.2	Mesh Parallelism for Architecture	132
7.2.1	Motivation and introduction	132
7.2.2	Basics of mesh parallelism	136
7.3	Offset Meshes	138
7.3.1	Types of exact offset meshes	139
7.3.2	Meshes with edge offsets	140
7.3.3	Designing with EO meshes	142
7.4	Optimizing Support Structures	145
7.4.1	Approximate offsets	145
7.4.2	Offset meshes by optimization in $\mathcal{P}(\mathcal{M})$	145
7.4.3	A processing pipeline from shape to beam layout	146
7.4.4	Other ways of optimization	149
7.5	Curvatures in Meshes with Planar Faces	150
7.6	Discussion	154
8	Conclusion and Future Research	157
8.1	Principal Contributions	157
8.2	Future Research	158
	Bibliography	159

List of Figures

1.1	Parametric curve and surface fitting	1
1.2	Architecture design by planar quadrilateral and hexagonal meshes	3
2.1	Iso-values curves of the point distance (PD) error term.	8
2.2	Iso-values curves of the tangent distance (TD) error term.	8
2.3	TD error at high curvature point	9
2.4	The squared distance function	10
2.5	The SD error term $e_{SD,k}(\mathcal{D})$	13
2.6	Example 2.1	15
2.7	Example 2.2	16
2.8	The evolution surfaces of Example 2.2	17
2.9	Example 2.3	18
2.10	Example 2.4	18
2.11	Foot-point computation on a fitting curve.	21
2.12	Deriving an error term for an outer data point X	21
2.13	Fitting a Chinese character	22
2.14	Comparison for fitting an open curve	23
2.15	Reconstruction of a revolution surface	24
2.16	The alternating minimization steps of PDM near a local minimum	33
2.17	TDM using the Armijo rule for step size control	33
3.1	Approximation of an archeological finding by a rotational surface	45
3.2	Approximation of a broken archeological finding by a rotational surface	46
3.3	Shell reconstruction	47
3.4	Machine part	50
3.5	Comparison for the machine part	51
3.6	Variations of the parameters for the machine part model in Figure 3.4.	51
3.7	Multiple scans of the tesa model	52
3.8	Registration and reconstruction of the tesa model	52
3.9	Multiple scans of a CAD model	53
3.10	Registration and reconstruction of the CAD model	53
4.1	The initial ellipse and data points of Case 4.	69
4.2	Comparisons of the six methods on a set of 200 data points	69
5.1	Critical points in a square	76
5.2	A 2D non-convex CVT after several Lloyd's iterations	78
5.3	Illustration of 2D CVT smoothness	79
5.4	Illustration of 3D CVT smoothness.	80

5.5	Example 5.3	81
5.6	Example 5.4	82
5.7	Voronoi diagrams and their dual	86
5.8	CCVT of David model	88
5.9	CCVT of Homer model	89
5.10	Geometric analysis of the CVT energy function F 's continuity	92
6.1	Conical meshes	96
6.2	PQ strip for a developable surface	99
6.3	Visualization of conjugacy via shadow contours.	99
6.4	Various conjugate networks and their suitability for meshing purposes	100
6.5	PQ perturbation without a closeness term applied to a highly un-planar mesh consisting of only a few quads.	103
6.6	PQ perturbation acting on a quad-dominant mesh	104
6.7	(a)–(c): Hierarchy of PQ meshes obtained by iterative application of Catmull-Clark subdivision and PQ perturbation.	106
6.8	Developable subdivision surfaces	106
6.9	Developable Möbius band in the shape of a trefoil knot	107
6.10	Configuration of the faces of a conical mesh at a vertex	107
6.11	A conical mesh has conical offset meshes	109
6.12	A conical mesh discretizes the network of principal curvature lines	110
6.13	Principal curves computed with different kernel radii.	112
6.14	Catmull-Clark subdivision and conical optimization	113
6.15	Design studies with conical meshes and their offset meshes produced by subdivision and conical perturbation	114
6.16	Design studies with developable surfaces	115
6.17	Circular mesh generated by optimizing a mesh generated from principal curves (top right).	116
6.18	Conical vertex of valence four	118
6.19	A convex spherical quad	119
6.20	Elliptic, parabolic, hyperbolic vertices and their associated spherical quadrilaterals	122
6.21	Stereographic images of four oriented great circles on the unit sphere	123
6.22	Converting an elliptic configuration into a parabolic configuration	124
6.23	An admissible parabolic quad having an incircle.	125
6.24	A parabolic admissible quad satisfying the angle criterion.	126
7.1	Architectural free form structure	130
7.2	Multi-layer constructions	132
7.3	Meshes $\mathcal{M}, \mathcal{M}'$ with planar faces are parallel if they are combinatorially equivalent, and corresponding edges are parallel.	134
7.4	Nodes, supporting beams, and the underlying geometric support structure of a steel/glass construction	135
7.5	The set $\mathcal{P}(\mathcal{M})$ of meshes parallel to a given mesh \mathcal{M} is a linear space and can be explored by a linear blend of some of its elements.	136
7.6	This construction is based on an edge offset mesh and has beams of constant height	138

7.7	This geometric support structure is defined by two parallel meshes which are not of constant edge-edge distance	138
7.8	A mesh \mathcal{M} with an edge offset mesh \mathcal{M}' at distance d	140
7.9	A Koebe polyhedron and related circles and cones.	141
7.10	Creation of the mesh \mathcal{M}' which Figure 7.1 is based on	142
7.11	Edge offset mesh of negative curvature	144
7.12	Assign a geometric support structure to a given mesh \mathcal{M}	146
7.13	Finding a support structure	147
7.14	Meshing and construction of a support structure with optimized nodes . .	148
7.15	These two systems of ‘approximate normal vectors’ of a mesh \mathcal{M} are defined by a Gauss image \mathcal{S}	150
7.16	Parallel polygons with vertices \mathbf{p}_i , \mathbf{q}_i , and $\mathbf{p}_i + d\mathbf{q}_i$	151
7.17	Construction of simple minimal quad meshes via parallelity of diagonals .	153
7.18	A discrete minimal EO mesh	154
7.19	This design with convex faces is composed from pieces of discrete cmc surfaces obtained in different ways	155

Dedicated to my parents and my wife

Chapter 1

Introduction

1.1 Motivation

With the rapid development of 3D model acquisition and advanced demands from industry and entertainment, the study of geometric modeling and processing becomes a hot research area in computer graphics and applied geometry. The main topics include shape representation, shape reconstruction, mesh processing, differential geometry in discrete setting, optimal geometry design, etc.

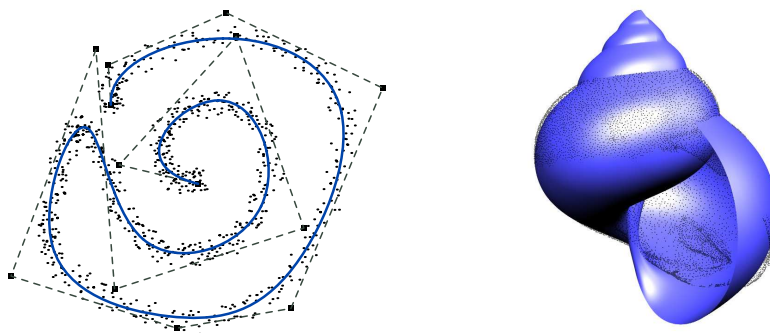


FIGURE 1.1: Left: Fitting an open B-spline Curve to a given point cloud [82]; right: Fitting a parametric surface to a point cloud scanned from a seashell.

Shape approximation and registration

A common subject among the various topics is finding an optimal solution in the consideration of geometry and economy, for instance, locating the best positions of control points of a spline surface to approximate a point cloud, determining the model parameters for shape reconstruction (See examples in Figure 1.1). One important issue is the efficiency and accuracy of optimization algorithms employed in geometric modeling and processing. Good understandings of the relationship between underlying geometry and

optimization theory would benefit the development of powerful geometry processing algorithms and tools. However, related study is somehow lacking in computer graphics community and traditional methods [67, 68] which only exhibits linear convergence, are still employed in most applications. In this thesis, a novel study of parametric shape reconstruction and model registration is presented clearly in the numerical optimization framework, and several super-linear algorithms are proposed and demonstrated. The concepts and algorithms can be easily integrated into most applications in computer graphics and geometric modeling.

Centroidal Voronoi Tessellation

Centroidal Voronoi Tessellation (CVT) is a powerful technique in many scientific and engineering fields, such as mesh generation and image segmentation. In practice the classical Lloyd's method [86] is quite popular but only presents linear convergence [48]. A major impediment to the development of superlinear methods for CVT computation is the popular, but wrong, perception that the CVT energy function, as a piecewise function, is not smooth enough. However, we prove that the energy function of the 2D/3D CVT in a convex domain is C^2 . Motivated and justified by this surprising result, we propose quasi-Newton optimization techniques for computing CVT with superlinear convergence.

Geometry in architecture

Nowadays geometric modeling is a key component in industrial and architecture design. Many geometric, aesthetical and economical demands rise in the design and manufacture process. In architectural freeform design, the relation between shape and fabrication poses new challenges and requires more sophistication from the underlying geometry, such as designing planar quadrilateral meshes for freeform glass structures [56]. But there is only little attempt in previous work of mesh processing and discrete differential geometry, where the study focuses on triangular meshes [40]. In this thesis, several novel geometric concepts and practical algorithms in the context of discrete differential geometry are proposed, including conical meshes, edge-offset meshes, and polygonal meshes planarization. A new concept – “parallel meshes” which include conical meshes, circular meshes and edge-offset meshes provides a unified framework for modeling and computation. Moreover, parallel meshes are the main ingredient in a novel discrete theory of curvatures, which is used to design discrete mean-curvature surfaces, minimal surfaces and polygonal meshes with geometrically optimal properties (see examples in Figure 1.2).

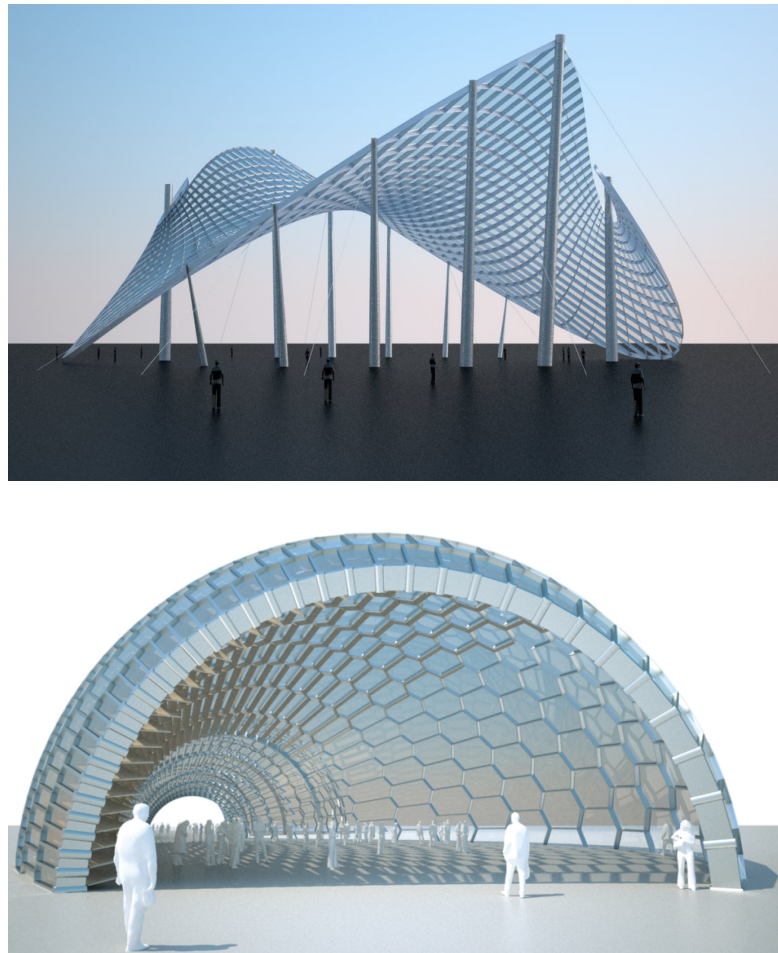


FIGURE 1.2: Architecture design by a planar quadrilateral mesh [113] and a planar hexagonal mesh [121].

1.2 Outline

This thesis addresses several problems of geometry and optimization in geometric modeling and it is organized as follows:

Chapter 2 presents a novel and efficient method, called *squared distance minimization* (SDM), for computing a planar B-spline curve to approximate a target shape defined by a point cloud. The theory and experiments show that SDM significantly outperforms other existing methods; (The presented material has been published in [151].)

Chapter 3 investigates 3D shape reconstruction from measurement data in the presence of constraints. The constraints may fix the surface type or set geometric relations between parts of an object's surface, such as orthogonality, parallelity. It is proposed to use a combination of surface fitting and registration within the geometric optimization framework of squared distance minimization (SDM). (The presented material has been published in [84].)

Chapter 4 systematically analyzes existing least squares orthogonal distance fitting

techniques in a general numerical optimization framework. The connections between the underlying geometry and Newton/Gauss-Newton methods are revealed. Two geometric variant methods (GTDM and CDM) are proposed. (The presented material has been published in [85].)

Chapter 5 studies the fast computation of “Centroidal Voronoi Tessellation” (CVT). The 2D/3D CVT function is proved to be C^2 and Quasi-Newton methods are applied to speed up CVT computation.

Chapter 6 proposes a new concept – “conical meshes” which are suitable for the design of freeform glass structure. They are tightly connected with principal curvature network which provides an initialization tool for creating conical meshes. With the perturbation and subdivision techniques, a powerful modeling tool for designing planar quadrilateral meshes is presented. (The presented material has been published in [83] and [152].)

Chapter 7 deals with the geometric challenges in the architectural design of freeform shapes come mainly from the physical realization of beams and nodes via the concepts of parallel meshes and geometric optimization methods. (The presented material has been published in [121].)

Chapter 8 summarizes the thesis and concludes possible future research.

Chapter 2

Fitting B-spline Curves to Point Clouds by Curvature-Based Squared Distance Minimization

2.1 Problem Formulation

We consider the following problem: Given a set of unorganized data points X_k , $k = 1, 2, \dots, n$, in the plane, compute a planar B-spline curve to approximate the points X_k . The data points X_k are assumed to represent the shape of some unknown planar curve, which can be open or closed, but not self-intersecting; this curve is called a *target curve* or *target shape*. We suppose that unorganized data points, often referred to as a *point cloud* or *scattered data points* in literature, may have non-uniform distribution with considerable noise; this assumption makes it difficult or impossible to order data points along the target curve. Hence, we assume that such an ordering is not available.

The above problem can be formulated as a nonlinear optimization problem as follows. Consider a B-spline curve $P(t) = \sum_{i=1}^m P_i B_i(t)$ with control points P_i . We assume throughout that the order and the knots of the B-spline curve are fixed, so they are not subject to optimization. This simplifying assumption allows us to give a clear explanation of the general idea of our new optimization scheme. The very same idea of optimization presented here can also be extended to the more general setting of fitting a NURBS curve with free weights and knots to be optimized, by variable linearization and inequality constraints. ¹

¹More general analysis is studied in Chapter 4.

Given data points X_k , $k = 1, 2, \dots, n$, we want to find the control points P_i , $i = 1, 2, \dots, m$, such that the objective function

$$f = \frac{1}{2} \sum_{k=1}^n d^2(P(t), X_k) + \lambda f_s \quad (2.1)$$

is minimized, where $d(P(t), X_k) = \min_t \|P(t) - X_k\|$ is the distance of the data point X_k to the curve $P(t)$, f_s is a regularization term to ensure a smooth solution curve and λ is a positive constant to modulate the weight of f_s . Here the distance $d(P(t), X_k)$ is measured orthogonal to the curve $P(t)$ from X_k . The exceptional case where the shortest distance from X_k to an open curve $P(t)$ occurs at an endpoint of $P(t)$ will be discussed separately in Section 2.5.

We present a novel method that approximates unorganized data points with a B-spline curve that starts with some properly specified initial curve and converges through iterative optimization towards the target shape of data points. One of our contributions is the introduction of a novel error term defined by a curvature-based quadratic approximant of squared distances from data points to the fitting curve. For brevity, this new error term is called the *squared distance error term* or *SD error term*, and the resulting iterative minimization scheme will be referred to as *squared distance minimization* or *SDM*. Because the SD error term measures faithfully the geometric distance between data points and the fitting curve, SDM converges fast and stably, in comparison with other commonly used error terms, as will be discussed shortly.

2.2 Related Work

2.2.1 Spline curve fitting techniques

Fitting a curve to a set of data points is a fundamental problem in graphics (e.g., [58, 104, 107, 122, 149]) and many other application areas. Instead of attempting a comprehensive review, we will only discuss some main results in the literature to provide a background for our work.

Let $X_k \in \mathbb{R}^2$, $k = 1, 2, \dots, n$, be unorganized data points representing a target shape, which is to be approximated by a closed or open planar B-spline curve $P(t) = \sum_{i=1}^m B_i(t)P_i$, where the $B_i(t)$ are the B-spline basis functions of a fixed order and knots, and the P_i are the control points. Since f in Equation. (2.1) is a nonlinear objective function, iterative minimization comes as a natural approach. Suppose that we have a specific B-spline curve $P_c(t) = \sum_{i=1}^m B_i(t)P_{c,i}$ with control points $\mathcal{P}_c = (P_{c,1}, P_{c,2}, \dots, P_{c,m})$, which can be an initial fitting curve or the current fitting generated from the last iteration.

Let $\mathcal{D} = (D_1, D_2, \dots, D_m)$ be the variable updates to \mathcal{P}_c to give the new control points $\mathcal{P}_+ = \mathcal{P}_c + \mathcal{D}$. Let $P_+(t) = \sum_{i=1}^m B_i(t)(P_{c,i} + D_i)$ denote the B-spline curve with updated control points \mathcal{P}_+ .

Many existing B-spline curve fitting methods invoke a data parameterization procedure to assign a parameter value t_k to each data point X_k . In some methods dealing with ordered data points, the chord length method or the centripetal method [52, 68, 79] is used for data parameterization. Then, with the fixed t_k , the function

$$\hat{f} = \frac{1}{2} \sum_k \|P_+(t_k) - X_k\|^2 + \lambda f_s, \quad (2.2)$$

which is a local quadratic model of f in (2.1), is minimized by solving a linear system of equations to yield updated control points \mathcal{P}_+ , and hence the updated fitting curve $P_+(t)$.

A commonly used data parameterization method is to choose t_k such that $P_c(t_k)$ is the closest point from the current fitting curve to the data point X_k ; ($P_c(t_k)$ is also called the *foot point* of X_k on the curve $P_c(t)$). Then an iterative method can be developed by interleaving this step of foot point computation with the minimization of the local quadratic model \hat{f} in (2.2) to compute the updated control points \mathcal{P}_+ . Note that the error term $\|P_+(t_k) - X_k\|^2$ in (2.2) measures the squared distance between the data point X_k and a particular point $P_+(t_k)$ on the fitting curve to be determined; therefore, we will call this error term the *point distance error term* or the *PD error term*, and denote it by

$$e_{PD,k} = \|P_+(t_k) - X_k\|^2. \quad (2.3)$$

(The PD error term is illustrated in Figure 2.1.) This minimization scheme will be called the *point distance minimization* or *PDM* for short. Note that, since the ordering of data points is not required for data parameterization via foot point projection, PDM is applicable to fitting a curve to a point cloud.

Hoschek [67] proposes an iterative scheme, called *intrinsic parameterization*, which also uses the PD error term but performs parameter correction using a formula that is a first-order approximation to exact foot point computation. Bercovier and Jacob [12] prove that the intrinsic parameterization method is equivalent to Uzawa's method for solving a constrained minimization problem, but they do not establish the convergence rate of the intrinsic parameterization method or that of PDM. Higher order approximation or accurate computation of foot points for data parameterization are discussed in [68, 69, 127].

PDM, or its simple variants, are the most commonly applied method for curve fitting in computer graphics and CAD [58, 67, 107, 127]. The same idea of PDM is also widely used for surface fitting [41, 54, 59, 60, 65, 66, 88, 90, 140, 153, 154], with B-spline surfaces as well as other types of surfaces. The popularity of PDM might be explained by its simplicity — the error term $e_{PD,k}$ in (2.3) is derived by simply substituting t_k in the squared distance $d^2(P(t), X_k)$ in the original objective function f in (2.1). However, considering that $P(t_k)$ is a variable point depending on the variable control points, $e_{PD,k}$ is a rather poor approximation to $d^2(P(t), X_k)$, thus causing slow convergence.

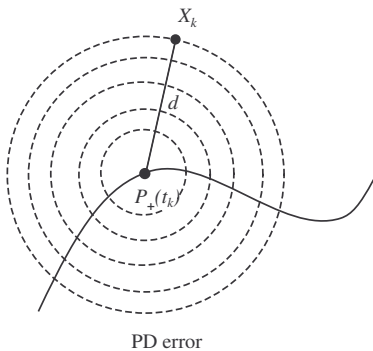


FIGURE 2.1: Iso-values curves of the point distance (PD) error term.

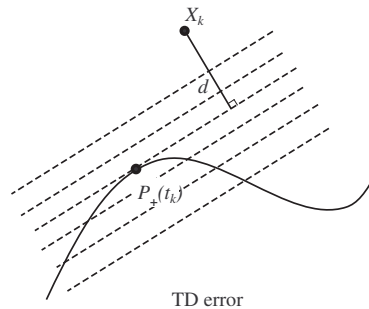


FIGURE 2.2: Iso-values curves of the tangent distance (TD) error term.

Another error term, often used in the computer vision community (e.g., [15]), is defined by

$$e_{TD,k} = [(P_+(t_k) - X_k)^T N_k]^2, \quad (2.4)$$

where N_k is a unit normal vector of the current fitting curve $P_c(t)$ at the point $P_c(t_k)$. We will call $e_{TD,k}$ the *tangent distance error term* or the *TD error term*, since $e_{TD,k}$ gives the squared distance from X_k to the tangent of $P_c(t)$ at the foot point $P_c(t_k)$ when $P_+(t)$ is $P_c(t)$. The TD error term is illustrated in Figure 2.2.

The TD error term $e_{TD,k} = [(P_+(t_k) - X_k)^T N_k]^2$ can also be combined with data parameterization via foot point projection to yield a B-spline curve fitting method, as used in [15] for boundary extraction in motion tracking. We will call this method *tangent distance minimization* or *TDM*. TDM minimizes in each iteration the function

$$f_{TD} = \frac{1}{2} \sum_k e_{TD,k} + \lambda f_s. \quad (2.5)$$

Treating the control points \mathcal{P}_+ as variables to be optimized, the TD error term measures the squared distance from the point X_k to a moving straight line L_k that has the fixed normal vector N_k and passes through the moving point $P_+(t_k)$. See Figure 2.3. The

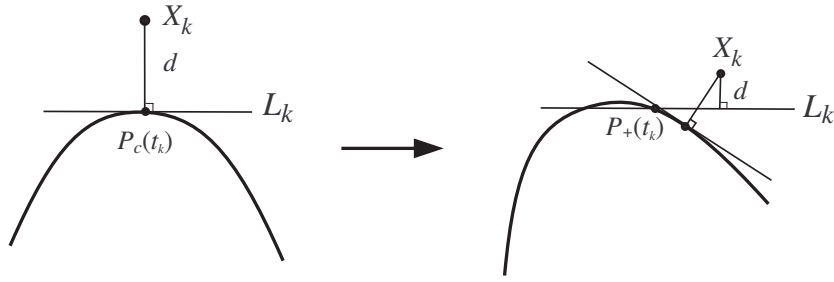


FIGURE 2.3: In a neighborhood of a high curvature point, the true approximation error can be rather big even when the TD error d is small.

TD error $e_{TD,k} = [(P_+(t_k) - X_k) \cdot N_k]^2$ becomes zero if the point X_k is contained in the line L_k . On the other hand, the line L_k is a relatively poor approximation to the curve $P_+(t)$ in a neighborhood of a high-curvature point $P_+(t_k)$ or if X_k is far from $P_+(t_k)$. Hence, in these cases, the point X_k may still be poorly approximated by the curve $P_+(t)$ even if X_k is nearly on L_k , i.e., when the TD error $[(P_+(t_k) - X_k) \cdot N_k]^2$ is nearly zero (see Figure 2.3).

This disparity between approximation quality and error measurement is the cause of the instability of TDM near a high-curvature part of the target shape, as will be illustrated later with test examples. This unstable behavior of TDM is, in fact, the consequence of using an inappropriately large step size to solve a nonlinear optimization problem; indeed, we will show that TDM uses Gauss-Newton iteration for solving a nonlinear least-squares problem and its excessively large step size is due to omission of important curvature related parts in the true Hessian.

Now let us consider the geometric interpretations of the PD error term and the TD error term. Since $P_c(t_k)$ is the fixed foot point on the current fitting curve $P_c(t)$ of X_k , if $P_+(t)$ is the same as $P_c(t)$ then both $e_{PD,k}$ and $e_{TD,k}$ give the same value of $d^2(P_c(t_k), X_k)$. However, for optimization purpose, we need to regard $d^2(P_+(t_k), X_k)$ as a function of variable control points \mathcal{P}_+ , and in this sense $e_{PD,k}$ and $e_{TD,k}$ give very different approximations to $d^2(P_+(t_k), X_k)$.

If treating X_k as a free point, for any constant $c > 0$, the iso-value curve $e_{PD,k} \equiv \|P_+(t_k) - X_k\|^2 = c$ of the PD error term is a circle (see Figure 2.1), and the iso-value curve $e_{TD,k} \equiv [(P_+(t_k) - X_k) \cdot N_k]^2 = c$ of the TD error term is a pair of parallel lines, which can be regarded as a degenerate ellipse (see Figure 2.2). Since PDM has relatively slow convergence and TDM tends to have fast but unstable convergence, one may speculate whether or not a new error term with ellipse-shaped iso-value curves can be devised to yield a more balanced performance between efficiency and stability. We

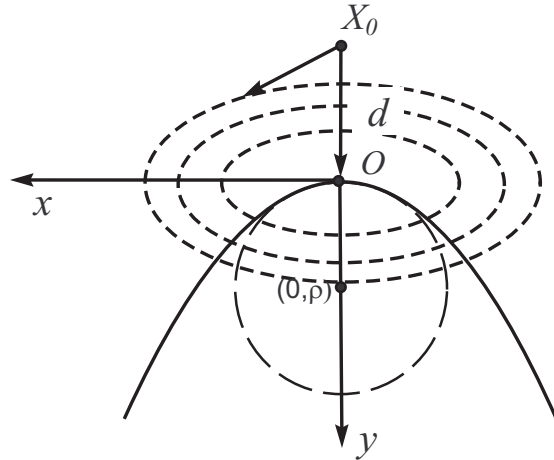


FIGURE 2.4: A second order approximant of the squared distance function to the curve C at X_0 .

will see that such an error term is naturally provided by a curvature-based quadratic approximation to the squared distance function.

2.2.2 Second order approximation to squared distance function

Given a curve C in \mathbb{R}^2 , one may define the squared distance function that assigns to each point X in \mathbb{R}^2 the squared distance from X to C . The second order approximation to this distance function is given in [6]. This approximation is then studied in detail and applied to solving a number of shape fitting problems in [111, 116]. Below we review this work briefly.

Let O be the closest point on a twice differentiable curve C to a fixed point X_0 (see Figure 2.4). Consider the local Frenet frame of C with its origin at O and its two coordinate axes being the tangent vector and the normal vector of the curve C at O . Let $\rho > 0$ be the curvature radius of C at O . We use an orientation of the curve normal such that $K = (0, \rho)^T$ is the curvature center of C at O . Let d be the signed distance from X_0 to O , i.e., $|d| = \|X_0 - O\|$ with $d < 0$ if X_0 and K are on opposite sides of the curve C , and $d > 0$ if X_0 and K are on the same side of C . We note that there is always $d < \rho$ when $d > 0$, for otherwise O cannot be the closest point on the curve C to X_0 .

Consider a point $X = (x, y)^T$ in a neighborhood of X_0 . The second order approximant of the squared distance from X to the curve C is [6, 111]

$$g(x, y) = \frac{d}{d - \rho} x^2 + y^2. \quad (2.6)$$

Geometrically, the conic section $g(x, y) = d^2$ has second order contact with the offset curve of the target curve C that passes through X_0 .

Since $g(x, y)$ in (2.6) is indefinite when $0 < d < \rho$, the unified expression

$$\hat{g}(x, y) = \frac{|d|}{|d| + \rho} x^2 + y^2 \quad (2.7)$$

is used in [116] as a positive semi-definite quadratic error term for solving geometric optimization problems. An alternative to this modification is to replace the first coefficient $d/(d - \rho)$ by $\max\{0, d/(d - \rho)\}$, thus also making the distance measurement positive semi-definite in all cases.

The above approximation to the squared distance of a smooth target curve is used in [116] for fitting a B-spline curve to the target curve as follows. Given a target curve to be approximated and the current B-spline fitting curve $P_c(t)$ with control points $\mathcal{P}_c = \{P_{c,i}\}$, a set of densely distributed points S_k , called *sensor points*, are first sampled on $P_c(t)$. Then the approximate squared distance $f_k(S_k)$ defined by (2.7) from each sensor point S_k to the fixed target curve is computed. Let $\mathcal{P}_+ = \mathcal{P}_c + \mathcal{D}$ denote the updated control points of the B-spline fitting curve, where \mathcal{D} are the incremental updates to the current control points \mathcal{P}_c . The error term associated with each sensor point is defined as

$$\hat{e}_k = \hat{g}(S_k(\mathcal{P}_+)), \quad (2.8)$$

which is quadratic in the control points \mathcal{P}_+ , since each S_k is a linear combination of the control points. Then the local error function $f = \frac{1}{2} \sum_k \hat{e}_k + \lambda f_s$, where f_s is a regularization term that is quadratic in \mathcal{P}_+ , is minimized to find the updated control points \mathcal{P}_+ by solving a linear system of equations. This minimization step is iterated to make the fitting curve $P(t)$ move towards the target curve.

The superior efficiency of the above curve fitting scheme comes from the fact that the error function \hat{e}_k in (2.8) is an accurate approximation to the true squared distance function from S_k to the target curve \mathcal{C} , in terms of the incremental updates \mathcal{D} . However, this method assumes that the target shape is a smooth curve whose tangent and curvature information can easily be evaluated or estimated accurately, thus it is not applicable to a target point defined by a point cloud because of the difficulty in computing accurate tangent and curvature from noisy or sparsely distributed data points.

2.3 Fitting a B-spline Curve to a Point Cloud Using SDM

In this section we introduce a new *SD error term* for fitting a B-spline curve to a point cloud. We emphasize that *this new SD error term is defined by a quadratic approximation to the squared distance function of the B-spline fitting curve, rather than that of the fixed target shape*. In other words, we measure the fitting error as defined in Equation. (2.1),

namely orthogonal to the fitting curve, in contrast to the method in [116] (or see Section 2.2.2) in which errors are measured orthogonal to the fixed target curve and therefore also a different objective function is minimized.

2.3.1 A new quadratic approximation to the squared distance

Given the current B-spline fitting curve $P_c(t) = \sum_{i=1}^m B_i(t)P_{c,i}$, let $P_+(t)$ denote the fitting curve with updated control points $\mathcal{P}_+ = \mathcal{P}_c + \mathcal{D}$, where $\mathcal{P}_c = \{P_{c,i}\}$ and \mathcal{D} are incremental updates to \mathcal{P}_c . Suppose that $P_c(t_k)$ is the foot point of the data point X_k on $P_c(t)$. Let T_k and N_k be the unit tangent vector and the unit normal vector of the current fitting curve $P_c(t)$ at the foot point $P_c(t_k)$, $\rho > 0$ is the curvature radius of $P_c(t)$ at $P_c(t_k)$ and $|d| = \|P_c(t_k) - X_k\|$, with the same convention on the sign of d as made in Section 2.2.2. When the control points \mathcal{P}_+ change, with the same parameter t_k , the foot point $P_c(t_k)$ becomes a variable point $P_+(t_k)$, and the unit tangent vector \tilde{T}_k , the unit normal vector \tilde{N}_k and curvature radius $\tilde{\rho}$ of the curve $P_+(t)$ at the point $P_+(t_k)$ all vary with \mathcal{P}_+ .

To obtain a quadratic approximation to the squared distance from X_k to the curve $P_+(t)$, we assume that \tilde{T}_k , \tilde{N}_k and $\tilde{\rho}$ are fixed to be T_k , N_k and ρ , i.e., they do not vary with \mathcal{P}_+ . This assumption implies that, locally at the point $P_c(t_k)$, we will only consider a differential translation of the curve $P_c(t)$ into the curve $P_+(t)$. Since this translation is relative to the data point X_k , we may view, in a relative sense, that $P_+(t)$ is a fixed curve and X_k undergoes a translation. Therefore, we may use the formula (2.6) to approximate the squared distance from X_k to the curve $P_+(t)$, expressed in the global coordinate system, as

$$h_k(\mathcal{D}) = \frac{d}{d-\rho} [(P_+(t_k) - X_k)^T T_k]^2 + [(P_+(t_k) - X_k)^T N_k]^2. \quad (2.9)$$

Note that $h_k(\mathcal{D})$ may take a negative value when $0 < d < \rho$. In order to obtain a positive semi-definite error metric, based on $h_k(\mathcal{D})$, we define the new error term as

$$e_{SD,k}(\mathcal{D}) = \begin{cases} \frac{d}{d-\rho} [(P_+(t_k) - X_k)^T T_k]^2 + [(P_+(t_k) - X_k)^T N_k]^2, & \text{if } d < 0, \\ [(P_+(t_k) - X_k)^T N_k]^2, & \text{if } 0 \leq d < \rho, \end{cases} \quad (2.10)$$

Clearly, $e_{SD,k}(\mathcal{D})$ is a positive semi-definite quadratic function of \mathcal{D} in all cases. Since $e_{SD,k}(\mathcal{D})$ is derived from a direct attempt to accurately approximate the squared distance function, we will call $e_{SD,k}(\mathcal{D})$ the *squared distance error term* or *SD error term* for short. We stress that, due to the simplifications we have made, $e_{SD,k}(\mathcal{D})$ is, in general, no longer a second order approximation to the squared distance function, but rather a

first order approximation that is more accurate than the PD error term or the TD error term.

When $d < 0$, the level-set curve of $e_{SD,k}(\mathcal{D}) = c$ is an ellipse centered at the point $P_+(t_k)$, if the X_k is treated as a variable point. When the control points \mathcal{P}_+ change, the ellipse is translated by keeping its center at $P_+(t_k)$ but with its shape, size and orientation remaining unchanged (see Figure 2.5). The SD error term $e_{SD,k}$ becomes the TD error term $e_{TD,k}$ when $0 \leq d < \rho$, i.e., when (i) the data point X_k is sufficiently near $P_c(t_k)$ relative to the magnitude of ρ ; and (ii) X_k is on the convex side of the curve $P_c(t)$, i.e., X_k and the curvature center K are on the same side of the curve $P_c(t)$. The use of the TD error term here will not cause instability, since in this case the tangent line is a relatively good approximation to the curve $P_c(t)$ in a neighborhood of $P_c(t_k)$.

Iterative minimization of the squared distance using the SD error term (2.10), interleaved with foot point computation for data parameterization, will be called *squared distance minimization* or *SDM*. Although the tangent and normal vectors T_k and N_k , as well as ρ , are kept fixed during an iteration, they are updated at the beginning of each iteration to reflect the continual change of the shape of the fitting curve.

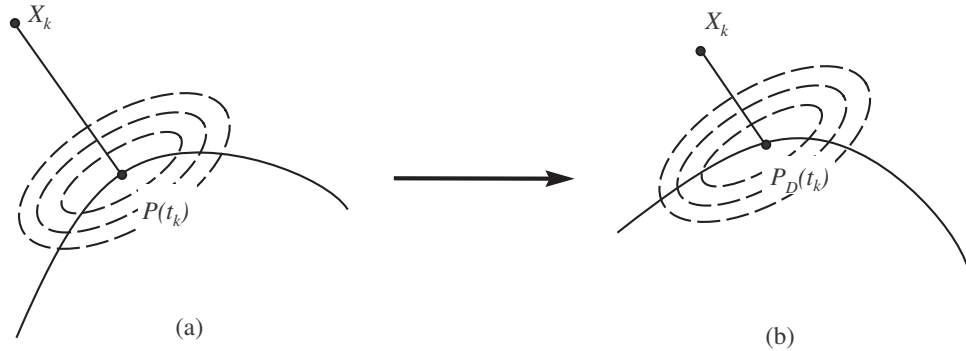


FIGURE 2.5: The SD error term $e_{SD,k}(\mathcal{D})$ defined by (2.10) is shown via its iso-value curves in the case of $d < 0$. (a) Before updating the control points \mathcal{P} . (b) The translation of $e_{SD,k}(\mathcal{D})$ after updating \mathcal{P} .

The above derivation of the SD error term is based on a geometric argument. Using a second order Taylor expansion, we will reveal the relationship between this SD error term and the Newton iteration later in Section 2.6.3, thus providing another derivation of the SD error term.

2.3.2 Main steps of SDM

The main steps of the SDM method are as follows.

- (1) Specify a proper initial shape of a B-spline fitting curve.
- (2) Compute SD error terms for all data points to obtain a local quadratic approximation f_{SD} of the objective function f , defined by

$$f_{SD} = \frac{1}{2} \sum_k e_{SD,k} + \lambda f_s.$$

- (3) Solve a linear system of equations to minimize f_{SD} to obtain an updated B-spline fitting curve.
- (4) Repeat steps 2 and 3 until convergence, i.e., until a pre-specified error threshold is satisfied or the incremental change of the control points falls below a preset threshold.

2.4 Experiments and Comparison

In this section we use some test examples to compare SDM with PDM and TDM for fitting a closed B-spline curve to unorganized data points in the plane. The quadratic function to be optimized in each iteration has the form

$$f = \frac{1}{2} \sum_{k=1}^n e_k + \alpha F_1 + \beta F_2, \quad (2.11)$$

where e_k is a particular error term (PD, TD or SD) for the data point X_k , F_1 and F_2 are energy terms defined as

$$F_1 = \int \|P'(t)\|^2 dt, \quad F_2 = \int \|P''(t)\|^2 dt, \quad (2.12)$$

and $\alpha, \beta \geq 0$ are constants. In our implementation F_1 and F_2 are integrated explicitly without numerical approximation.

For a fixed B-spline fitting curve $P(t)$, the Euclidean distance from data point X_k to $P(t)$ is denoted by $d_k = \|P(t_k) - X_k\|$, where $P(t_k)$ is the foot point of X_k on $P(t)$. Then, for evaluating the approximation error, we define the average error, which is the *root mean square error*, as

$$Error_Ave = \left[\frac{1}{n} \sum_{k=1}^n d_k^2 \right]^{1/2},$$

and the maximum error as

$$Error_Max = \max_{k=1}^n \{d_k\}.$$

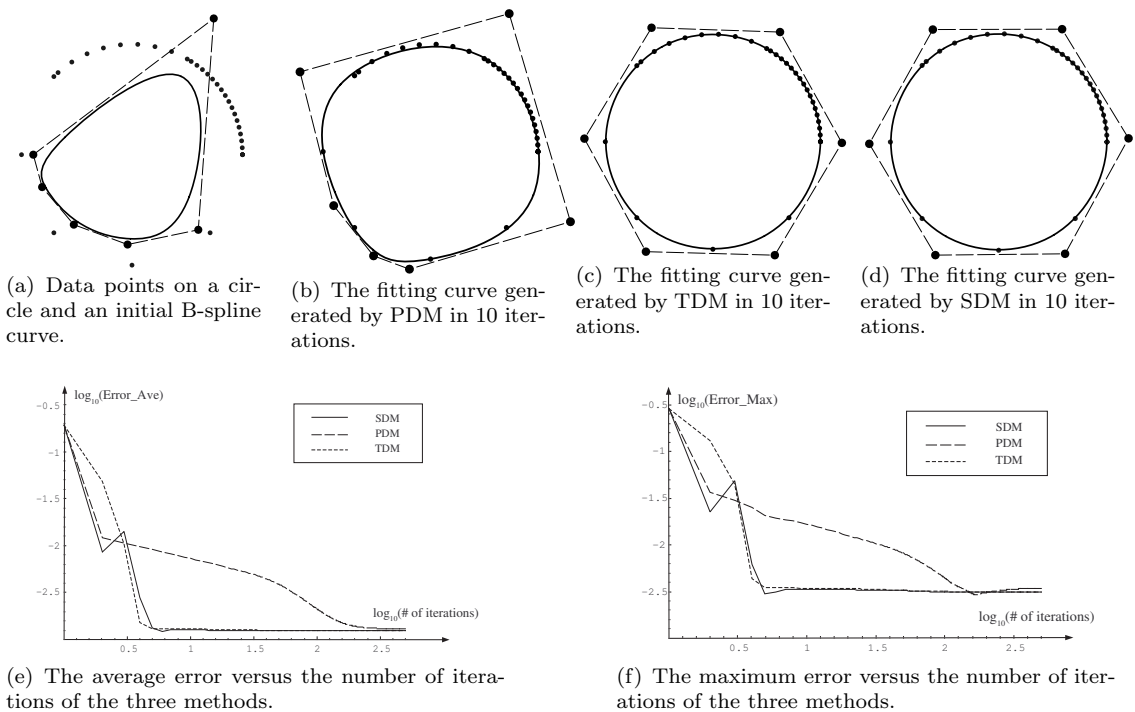


FIGURE 2.6: (Example 2.1) Comparison of the three methods on a set of 32 sparse data points on a circle. In this figure log scale (base 10) is used for the iteration axis in Figures (e) and (f) in order to distinguish the error curves of TDM and SDM. (Note that the sudden increase in SDM is due to the rapid change of the distribution of control points.)

We present below the results of applying the three methods — PDM, TDM and SDM — to fitting a cubic B-spline curve with uniform knots to several sets of unorganized data points. The same values of energy coefficients $\alpha = 0$ and $\beta = 0.001$ are used for all the examples in this section, unless specified otherwise.

Example 2.1. *Non-uniform data points on a circle.* (See Figure 2.6.) TDM and SDM converge with roughly the same speed, and both converge much faster than PDM does, as shown in Figures 2.6(e) and (f). PDM takes about 100 iterations to reach the same small error values produced by SDM in fewer than 10 iterations.

Example 2.2. (See Figure 2.7.) For this set of data points, SDM again converges much faster than PDM does, while TDM is trapped in a local minimum, producing a curve with self-intersection. To visualize the evolution of an iterative optimization process (PDM or SDM), we place the fitting curves generated in successive iterations at successive heights to form an *evolution surface* (see Figure 2.8). Data points or a subset of them are displayed at the top of an evolution surface. A striped texture is used to depict the trajectories of points of fixed parameter values on the fitting curve. The trajectories of evolving control points are shown by white curves in space. Log scale (base 10) is used for the height axis in these figures to accommodate for the large

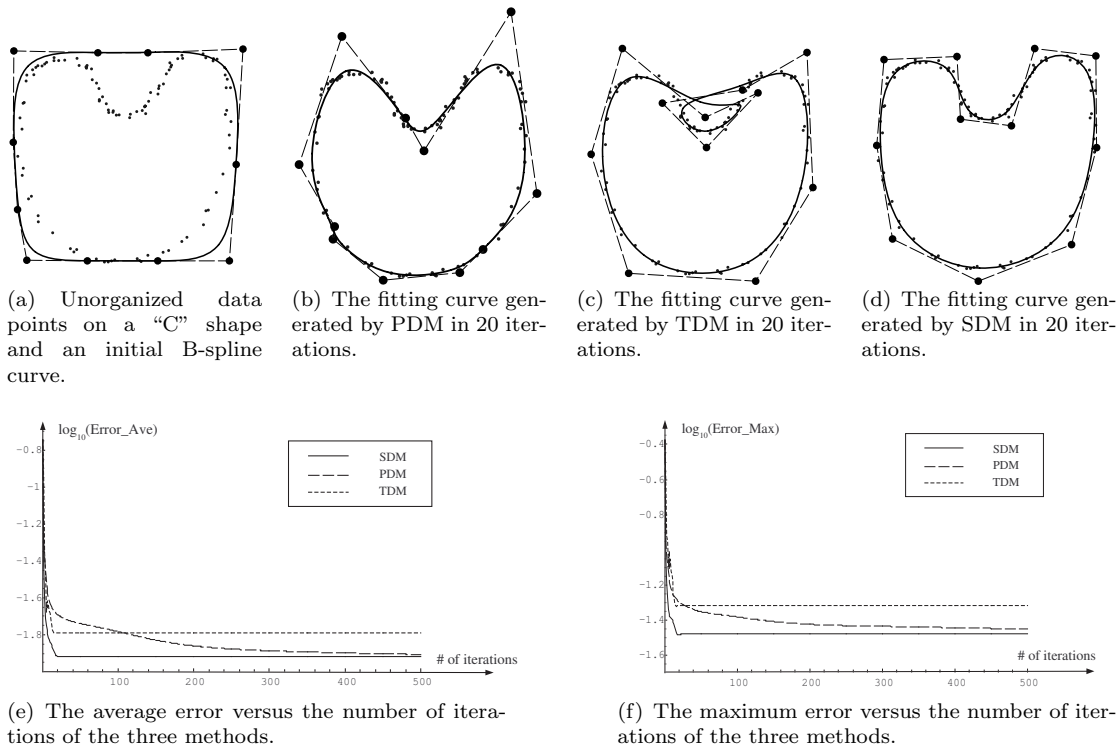


FIGURE 2.7: (Example 2.2) Comparison of the three methods on a set of 102 data points. SDM converges faster than PDM does. TDM is trapped in a local minimum with self-intersection in the fitting curve.

number of iterations needed by PDM. For size reference, a base square of size 2.2×2.2 is shown along with these evolution surfaces.

The evolution surfaces generated by PDM and SDM (Figure 2.8), viewed from two different directions, show that PDM experiences a slow convergence process, while SDM converges quickly with conspicuous tangential flow of the control points in the first 10 iterations.

Example 2.3. (See Figure 2.9.) This set of data points is extremely noisy. After 50 iterations, SDM has already produced an acceptable result, while PDM converges slowly and TDM becomes unstable. Here strong tangential flow of control points is observed in SDM to move some control points at the bottom of the initial curve to the top in (d).

Example 2.4. (See Figure 2.10.) The difficulty with this test lies in the corner points of the target shape and the highly non-uniform distribution of the control points of the initial B-spline curve. After 20 iterations, PDM gets trapped in an unacceptable local minimum and TDM becomes divergent, while SDM converges successfully. Again SDM exhibits strong tangential flow responsible for re-distributing control points initially clustered at one side on the initial curve over the target shape to well approximate the four corner points.

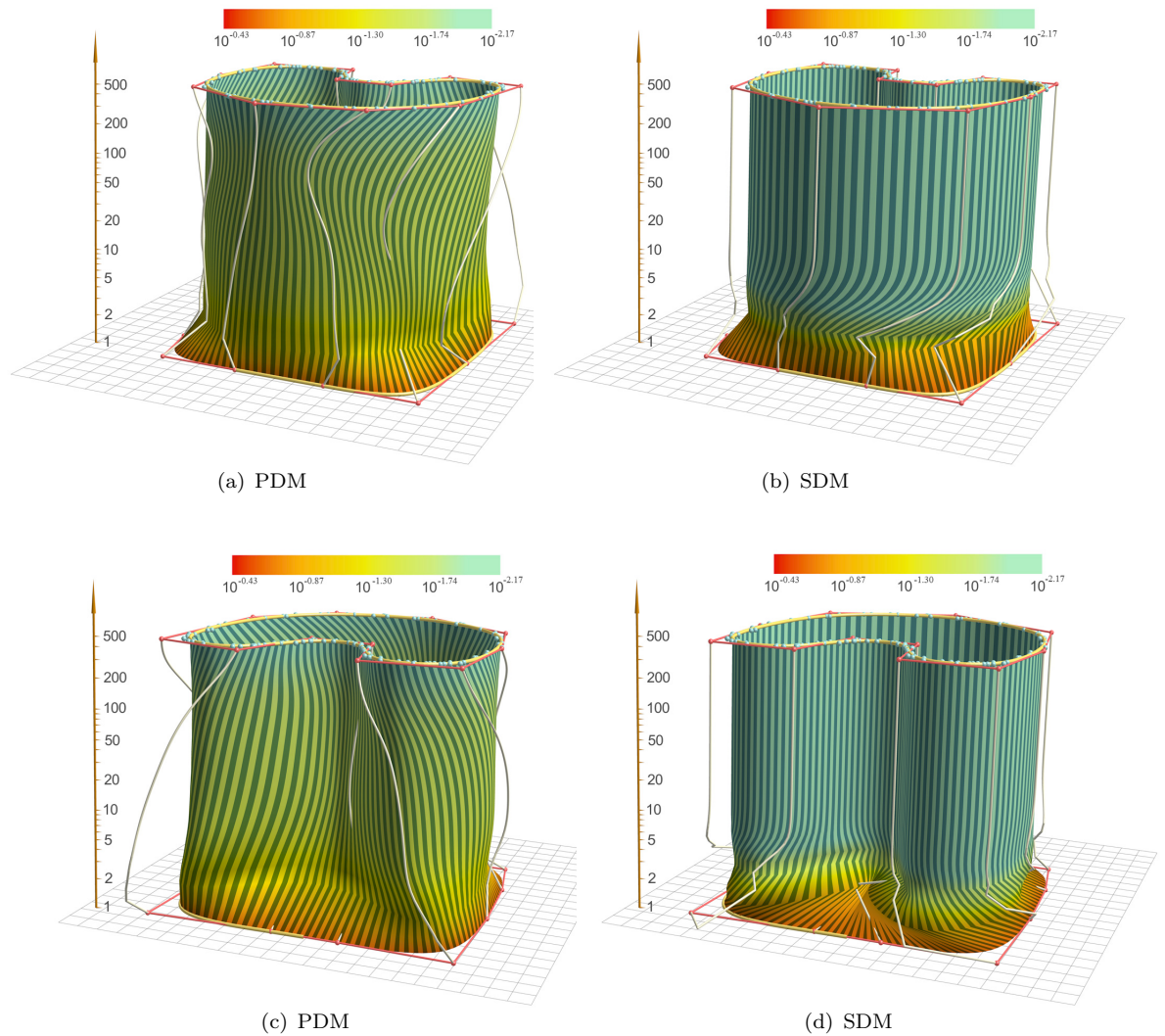


FIGURE 2.8: The evolution surfaces of PDM and SDM for the data in Example 2.2 (Figure 2.7) — viewed from two different angles. (a) The evolution surface of PDM; (b) The evolution surface of SDM with the same view angle as in (a); (c) The evolution surface of PDM from another view angle; (d) The evolution surface of SDM with the same view angle as in (c).

The four examples above are from numerous examples with which we have experimented. The following observations can be made from our experiments.

- 1) PDM exhibits the slowest convergence among the three methods, and is often trapped at a poor local minimum. Our experiments confirm the theoretical conclusion that PDM has, in general, only linear convergence. This is further explained in Section 2.6.
- 2) TDM demonstrates fast convergence when the target shape is not so noisy (i.e., representing a small-residue problem) and the initial fitting curve is relatively near

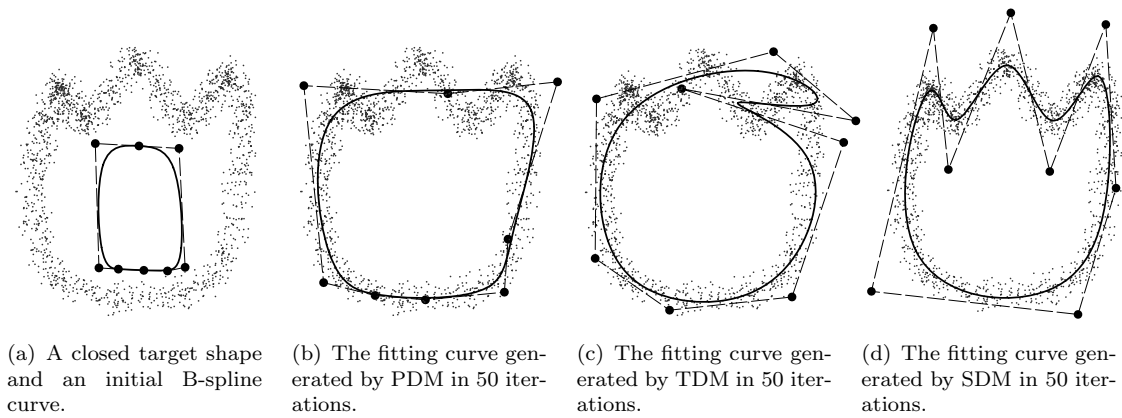


FIGURE 2.9: (Example 2.3) Comparison of the three methods for fitting an extremely noisy data set of 1,630 points. After 50 iterations, SDM generates a satisfactory fitting curve (d), but TDM becomes unstable (c), and PDM is still improving at a slow rate (b); PDM needs about 400 iterations to produce a fitting curve similar to the one by SDM shown in (d).

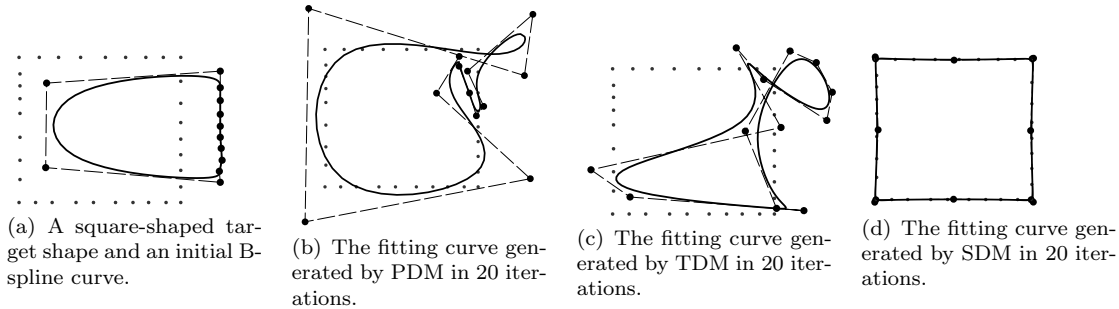


FIGURE 2.10: (Example 2.4) Comparison of the three methods for approximating a square-shaped target shape consisting of 33 data points. PDM is trapped in a poor local minimum (b), TDM eventually becomes divergent (c), and SDM converges successfully in 20 iterations.

the target shape (i.e., $|d| \leq \rho$), but often becomes unstable or even develops self-intersection in a high curvature region of the target shape or if the initial fitting curve is relatively far from the target shape. Increasing the value of the energy coefficient β in (2.11) usually improve the stability of TDM, as well as the fairness of the fitting curve, but often at the expense of a larger fitting error.

- 3) SDM exhibits much faster convergence than PDM does. The convergence of SDM is about as fast as that of TDM; moreover, SDM is more stable than TDM, since TDM often does not converge for target shapes with shape features. This is mainly due to the fact that TDM is a Gauss-Newton method without step size control. We will discuss this in more detail in Section 2.6.
- 4) The iso-value curves of the SD error term are ellipses aligned with the tangent at a point of the fitting curve. Therefore, at a low-curvature region of a B-spline fitting curve, the control points of the fitting curve, as well as points on the fitting

curve, can flow in the tangential direction to attain a better distribution without causing much penalty from the SD error term. Meanwhile, as desired, such a flow is dampened at a high-curvature region due to the role played by the curvature radius ρ and distance d in the SD error term. In contrast, tangential flow of control points is inhibited by the PD error term, causing stagnant improvement. Meanwhile, this tangential flow is checked nowhere by the TD error term, since the TD error term ignores curvature variation on the fitting curve, thus leading to unstable convergence in the presence of corner points in the target shape.

The ease of implementation and per-iteration computation time of SDM are nearly the same as those of PDM and TDM, since the three methods share the same framework but use different quadratic error terms. The per-iteration computation time of SDM is mainly determined by the number of data points. The dominant part of computation time is the computation of the foot points of all data points in each iteration. For example, for the set of 1,630 data points used in Example 2.3, computation of each iteration takes about 0.15 seconds on a PC with Pentium IV 2.4GHz CPU and 256 MB RAM, with over 95% of this time spent on foot point computation.

2.5 Implementation Issues

In this section we discuss the following implementation issues for facilitating the convergence or improving the computational efficiency of SDM: 1) initialization and adjustment of control points; 2) fast setup of error terms; and 3) adapting SDM to fit an open B-spline curve to data points.

2.5.1 Initialization and adjustment of control points

All the three methods we have discussed so far, PDM, TDM and SDM, are local minimization schemes; that is to say, their convergence depends on the initial value, i.e., the initial fitting curve. We would like to point out several possibilities of specification of the initial fitting curve, though this is not a focus of this chapter. The first obvious option is to let the user specify an initial B-spline fitting curve that is sufficiently close to the target shape and has an appropriate number of control points.

For a target shape defined by a set of dense points, an alternative is to compute a quadtree partition of the data points and then extract a sequence of points approximating the target shape from non-empty cells, i.e., those cells containing at least one data point. These extracted points can then be used as the control points of an initial B-spline fitting

curve. Our experience shows that this method tends to produce too many control points at the beginning, so control point deletion is normally required during the fitting process in order to obtain a fitting curve with a minimal number of control points while still meeting a prescribed error threshold.

Another approach under our current investigation is based the active contour model. In this method, a simple initial fitting curve is specified either automatically or by the user. Then some external energy/force is used in combination with SDM to drive the fitting curve to converge to a complex target shape in a manner that ensures global convergence. The key issues here are (i) proper design of the external force so that features and concavities of the target shape can be captured; and (ii) progressive insertion of control points at appropriate locations and stages so as to provide increasing shape flexibility to cope with the complexity of the target shape. We refer to [158] for discussions on control point insertion and deletion in the context of a local B-spline curve fitting procedure.

2.5.2 Fast setup of error terms

Efficient computation of foot points on the fitting curve of data points is important, especially when they are a large number of data points, since an error term needs to be computed for each data point in every iteration. We use the following speedup method consisting of two phases: *preprocessing* and *query*. In preprocessing we first compute a uniform spatial partition of the data points with a proper cell size and record those nonempty cells. Next we sample a sufficient number of points on the fitting curve and compute the normal lines of the fitting curve at these sample points. Then we record the intersections between these normal lines and all the non-empty cells.

In the query phase, for each data point X_k , we find its containing cell and the two normal lines such that X_k is between the two lines and they are closest to X_k (see Figure 2.11). Let the two normal lines be associated with parameter values \bar{t}_1 and \bar{t}_2 of the fitting curve. Let d_1 and d_2 denote the distances from X_k to the two lines. Then, supposing that the current fitting curve is sufficiently close to the target shape, a good estimate $P(\tilde{t}_k)$ of the foot-point of X_k is given by linear interpolation $\tilde{t}_k = (d_2\bar{t}_1 + d_1\bar{t}_2)/(d_1 + d_2)$. The point $P(\tilde{t}_k)$ is then used as an initial point in a Newton-like iterative procedure to find the foot point $P(t_k)$ of X_k .

Figure 2.13 shows an example of using SDM and PDM to fit a B-spline curve to the contour of a Chinese character Tian, meaning sky. In this example, the procedures described in Section 2.5.1 are used for initial fitting curve specification and control point insertion. Again we see that, to achieve the same level of fitting quality, PDM needs about the same number of control points but much more iterations than SDM does.

TDM without step size control fails to converge for this example, because the font outline has a number of high curvature feature points.

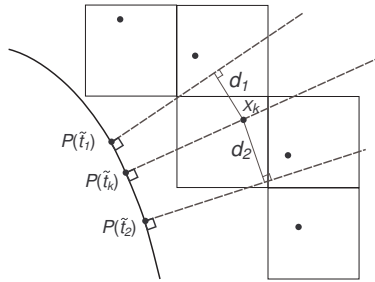


FIGURE 2.11: Foot-point computation on a fitting curve.

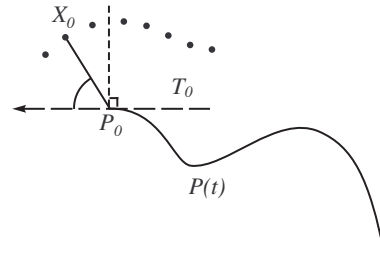


FIGURE 2.12: Deriving an error term for an outer data point X .

2.5.3 Fitting an open B-spline curve

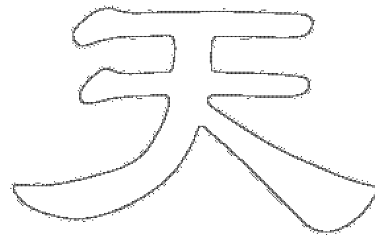
SDM can also be used to fit an open curve to a point cloud that represents an open target curve, with some necessary modifications to ensure that the endpoints of the fitting curve are properly determined. We assume that the target curve is not self-intersecting and that a proper initial shape of an open fitting curve is provided. The data points near an end of the target curve are called *target endpoints*. There are two cases to consider: *Case 1*: some data points cannot be projected to inner points of the fitting curve; such points are called *outer data points* with respect to the fitting curve under consideration. *Case 2*: all data points can be projected to inner points of the fitting curve.

In the first case, the error term associated with an outer data point is derived by blending the SD error term and the PD error term. Specifically, referring to Figure 2.12, let T_0 be the unit tangent vector of the fitting curve $P(t)$ at its endpoint P_0 . Let X_0 be an outer point such that P_0 is the closest point from the curve $P(t)$ to X_0 . Let θ denote the angle between the tangent line of the curve $P(t)$ at P_0 and the vector $X_0 - P_0$, with $|\theta| < \pi/2$ (since X_0 is an outer point). Then the error term $e_{outer,0}$ to be used for X_0 is given by the following interpolation of the PD error term and SD error term,

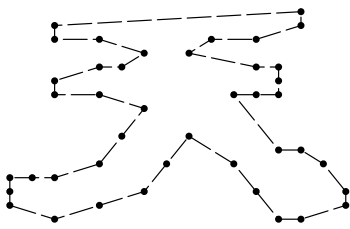
$$e_{outer,0} = \cos \theta e_{PD,0} + (1 - \cos \theta) e_{SD,0}. \quad (2.13)$$

Here P_0 is regarded as a function of the control points.

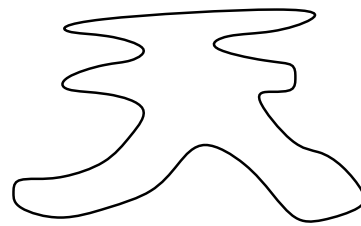
The rationale behind the interpolation in (2.13) is to use the PD error term partially for outer data points so that, through iterative optimization, the endpoint P_0 of the fitting curve is pulled towards the target endpoints; of course, the SD error terms are still used for all other non-outer points. Note that the outer points in a target shape are identified

(a) A Chinese character, *tian*.

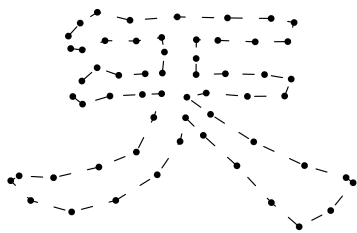
(b) The contour of the character (2,656 points).



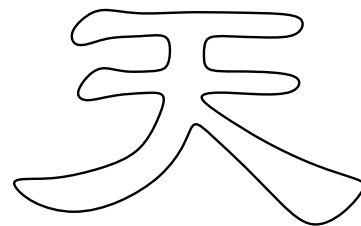
(c) 42 control points of an initial B-spline curve.



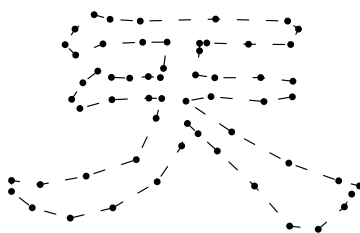
(d) The initial B-spline curve.



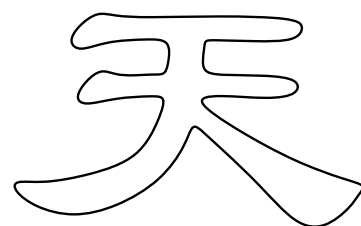
(e) Control points generated by SDM.



(f) The B-spline fitting curve from (e).



(g) Control points generated by PDM.



(h) The B-spline fitting curve from (g).

FIGURE 2.13: SDM and PDM are applied to fitting a B-spline curve to the contour of a Chinese character in (a). The initial fitting curve in (d) has the control points in (c) that are extracted from a quad-tree partition of the contour data points in (b). The coefficient of the smoothing term is $\lambda = 0.005$. SDM produces the fitting curve in (f) with the 59 control points shown in (e) after 54 iterations and PDM produces the fitting curve in (h) with the 60 control points in (g) after 352 iterations.

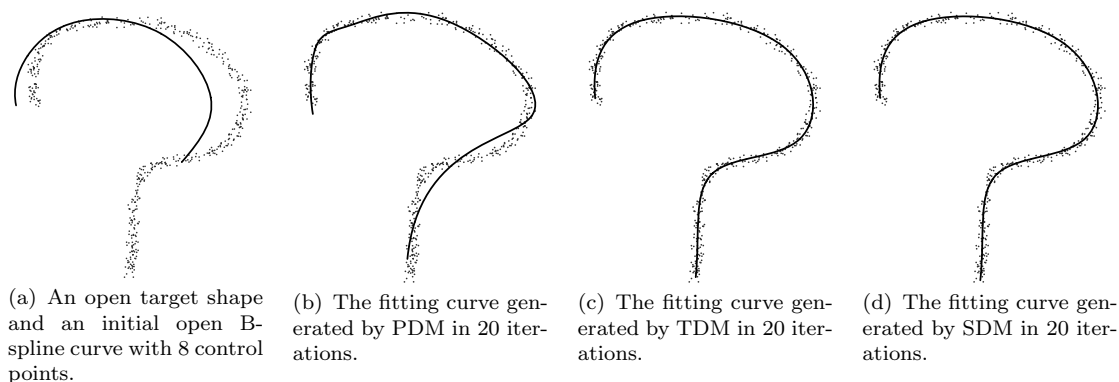


FIGURE 2.14: Comparison of the three methods for fitting an open curve to a set of 472 data points. PDM needs about 600 iterations to reach the small approximation error of SDM as shown in (c).

relative to the current fitting curve; therefore we may have different data points as outer points in every iteration.

In the second case where none of the data points is an outer point, we just use the standard SDM method — that is, use the SDM error term for each data point, to make the fitting curve to contract to fit the target shape. If there are some control points which are not constrained, we sample some points on those B-spline segments determined by them and find the corresponding nearest data points, add PDM error term into the objective function. A non-zero but small value of α for the energy term F_1 in (2.12) may be used to speed up the speed of contraction.

We are going to present two examples of fitting open B-spline curves to data points, using the technique described above, in combination with SDM and PDM. The first example is shown in Figure 2.14. We note that, in this example, PDM takes about 600 iterations to reach the same approximation error that is achieved by SDM in 20 iterations. This is again due to the strong tangential flow of B-spline control points that is accommodated by the SDM error term. We note that TDM works effectively for this example as well.

The second example, shown in Figure 2.15, is an application to reconstructing a revolution surface from a point cloud scanned in by a laser range scanner, following a method proposed in [115]. The basic idea is as follows. First, the rotation axis of the revolution surface is estimated from the data points. Then this axis is used to rotate the input data points in 3D (Figure 2.15(a)) into data points lying on a 2D plane (Figure 2.15(b)), from which the B-spline profile curve is reconstructed using SDM. Then this profile curve is used to generate a revolution surface (Figure 2.15(c)) approximating the input 3D data points.

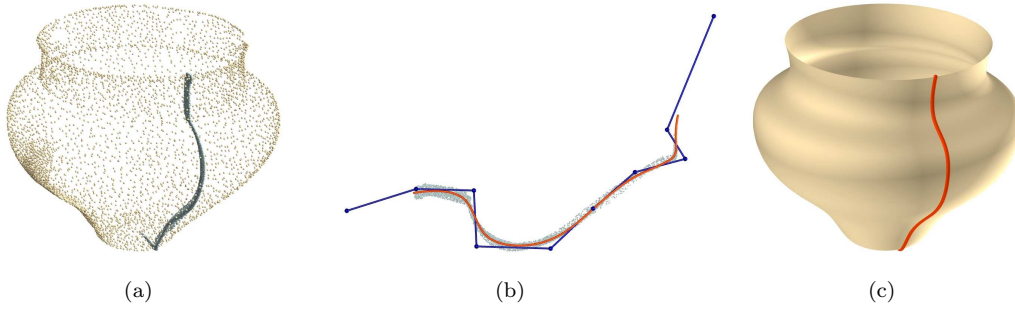


FIGURE 2.15: Reconstruction of a revolution surface from a point cloud using a B-spline profile curve computed by SDM. (a) A revolution surface represented by a set of 5,077 points sampled from an original scan data of 423,697 points; (b) The thick planar point cloud is fitted with a uniform cubic B-spline curve computed by SDM; (c) The reconstructed revolution surface.

2.6 Discussion from Viewpoint of Optimization

In this section we discuss SDM, as well as PDM and TDM, for B-spline curve approximation from the viewpoint of optimization. The B-spline curve fitting problem, as formulated in (2.1), can also be seen as the nonlinear optimization problem of minimizing

$$f = \frac{1}{2} \sum_{k=1}^n \|P(t_k) - X_k\|^2 + \lambda f_s, \quad (2.14)$$

where $P(t_k)$ is a normal foot point of X_k , i.e.,

$$(P(t_k) - X_k)^T P'_t(t_k) = 0, \quad k = 1, 2, \dots, n. \quad (2.15)$$

We note that, treating $\mathcal{P} = \{P_i\}$ and $\mathcal{T} = \{t_k\}$ as two separable groups of variables and Equation. (2.15) as constraints, this curve fitting problem is an instance of a *separable and constrained nonlinear least squares problem* for which the variable projection method using Gauss-Newton iteration often provides an efficient algorithm [14]. This viewpoint will be helpful below in the computation of gradient and Hessian of the objective function f . For simplicity of discussion, in the following we will ignore the regularization term f_s ; our conclusion is still applicable with f_s being taken into consideration, since f_s is independent of the t_k and quadratic in the P_i , assuming that λ is a fixed constant throughout all iterations.

In the rest of this section we will first explain why PDM has linear convergence, by viewing it as an alternating method. We will then investigate the standard algorithms for nonlinear least squares problems, namely Gauss-Newton iteration and the Levenberg-Marquart method [74], in connection with TDM. We will show that the Gauss-Newton method based on variable projection [14] is exactly the same as TDM, which is used

in [15]. From this we conclude on scenarios where TDM works well: a good initial position of the fitting curve and a small residual problem (i.e., data points are close to the final fitting curve); for a zero residual problem, optimization theory tells us that this method exhibits even quadratic convergence. The Levenberg-Marquart method is seen as a regularized version of TDM.

Our SDM scheme is finer than these standard methods (i.e., gradient descent and Gauss-Newton) for curve fitting as a nonlinear least squares problem. Although SDM is not a full Newton method because we do not use the complete Hessian, it comes close to it; in SDM we approximate the complete Hessian by a quadratic approximation to the squared distance to a fitting curve in a manner that makes SDM adaptable to local curvature variation.

In fact, all the methods above can be seen as gradient descent schemes in some appropriate metric. Whereas SDM chooses carefully the metric and TDM does this at least close to the target shape, PDM uses a metric which is not well adapted at all. Finally, we mention step size control [74] as a means of global convergence improvement of the three methods.

2.6.1 PDM as an alternating method

We now explain why PDM is a variant of the steepest descent method, and therefore has a linear convergence rate. Given a planar B-spline fitting curve $P(t)$ with control points P_i and data points X_k , PDM minimizes the error function $f(\mathcal{P}, \mathcal{T})$ defined by (2.14), which is a function in the Euclidean space \mathbb{R}^{2m+n} spanned by \mathcal{P} and \mathcal{T} , where $\mathcal{P} = \{P_i\}_{i=1}^m$ are the control points of the fitting curve and $\mathcal{T} = \{t_k\}_{k=1}^n$ are the parameter values associated the data points X_k .

PDM has the following two steps that are carried out in each iteration (see Figure 2.16): (1) For fixed parameter values $\mathcal{T}_0 = \{t_{k,0}\}$ and current control points $\mathcal{P}_0 = \{P_{i,0}\}$, find new control points $\mathcal{P}_1 = \{P_{i,1}\}$ by minimization of the quadratic function $f(\mathcal{P}, \mathcal{T}_0)$; (2) Considering the control points \mathcal{P}_1 produced in step 1 as fixed, find new parameter values $\mathcal{T}_1 = \{t_{k,1}\}$ by minimization of the error function $f(\mathcal{P}_1, \mathcal{T})$; this is done by computing the foot points $P(t_k)$ of the data points X_k on the fitting curve $P(t)$ with the control points \mathcal{P}_1 . Thus, PDM minimizes the objective function in two subspaces parallel to \mathcal{P} and \mathcal{T} , respectively, leading to a zigzag path near a local minimum as shown in Figure 2.16, which is reminiscent of the crawling behavior of the gradient descent method.

Due to the separate and alternate minimization of the \mathcal{P} and \mathcal{T} variables in each iteration, PDM is an *alternating method*, which is a typical optimization technique for

solving a separable nonlinear least squares problem and is known to have only linear convergence [14].

2.6.2 TDM – Gauss-Newton iteration and its variants

First we will derive an expression of the gradient of the objective function in (2.14), which can also be regarded as a function of the m control points $\mathcal{P} = (P_1, P_2, \dots, P_m)$, i.e., a function $f : R^{2m} \rightarrow R$; the dependence of \mathcal{T} on \mathcal{P} is built through the constraints in (2.15). For a fixed k , to indicate the dependence of t_k on \mathcal{P} , we write $t_k = t(\mathcal{P})$, omitting the subscript for simplicity. Denote $\mathcal{F} = P(t_k) - X_k$ and $f_k = \|\mathcal{F}\|$. Then the constraints (2.15) can be re-written, for each k , as

$$\mathcal{F}_t^T \mathcal{F} = 0. \quad (2.16)$$

Here and in the sequel we will denote partial derivatives with a subscript, e.g., $\mathcal{F}_t := \partial \mathcal{F} / \partial t$, $\mathcal{F}_{tt} := \partial^2 \mathcal{F} / \partial t^2$.

We first compute the gradients of f_k^2 and f_k . Since $f_k^2 = \mathcal{F}^T \mathcal{F}$, by (2.16), we have

$$\nabla f_k^2 = \nabla(\mathcal{F}^T \mathcal{F}) = 2(\mathcal{F}_\mathcal{P}^T + \nabla t \mathcal{F}_t^T) \mathcal{F} = 2\mathcal{F}_\mathcal{P}^T \mathcal{F}. \quad (2.17)$$

Here, ∇t is the gradient of t_k with respect to \mathcal{P} , and $\mathcal{F}_\mathcal{P}$ is the matrix representing the partial derivative of \mathcal{F} with respect to \mathcal{P} , not taking into account the dependency of t_k on \mathcal{P} . Since $\nabla f_k^2 = 2f_k \nabla f_k$, we obtain

$$\nabla f_k = \mathcal{F}_\mathcal{P}^T \frac{\mathcal{F}}{f_k} = \mathcal{F}_\mathcal{P}^T \frac{\mathcal{F}}{\|\mathcal{F}\|} = -\mathcal{F}_\mathcal{P}^T N_k, \quad (2.18)$$

where $N_k = -\mathcal{F} / \|\mathcal{F}\|$ is a unit normal vector of the curve $P(t)$ at $P(t_k)$. Then the gradient of f is found to be

$$\nabla f = \frac{1}{2} \sum_k \nabla f_k^2 = \left(\sum_{k=1}^n B_1(t_k)(P(t_k) - X_k), \dots, \sum_{k=1}^n B_m(t_k)(P(t_k) - X_k) \right)^T.$$

Each component of the above gradient vector, i.e.,

$$(\nabla f)_i = \sum_{k=1}^n B_i(t_k)(P(t_k) - X_k),$$

stands for a 2D vector associated with the i -th control point P_i , which is a weighted sum of the error vectors $P(t_k) - X_k$, where the weights are given by the i -th basis function

B_i , evaluated at the parameter t_k of the foot point; of course, only those error vectors in the support of B_i have influence.

At some places, it will be convenient to represent the B-spline curve in matrix form,

$$P(t) = B(t)\mathcal{P}, \quad (2.19)$$

where $B(t)$ is the $2 \times 2m$ matrix $(B_1(t)I_2, \dots, B_m(t)I_2)$ with I_2 being the 2×2 identity matrix. Then the gradient vector $\nabla f \in R^{2m}$ can be written as

$$\nabla f = \sum_{k=1}^n B^T(t_k)(P(t_k) - X_k) = \sum_{k=1}^n B^T(t_k)B(t_k)\mathcal{P} - \sum_{k=1}^n B^T(t_k)X_k. \quad (2.20)$$

Now let us consider the Gauss-Newton method. A Newton method minimizes the second-order approximant of the objective function at the current position x_c to obtain the next iterate x_+ . To find this quadratic approximant for a nonlinear least squares problem with $f = \frac{1}{2} \sum_k f_k^2$, one needs to compute the Hessian of f , which is

$$\nabla^2 f = \sum_{k=1}^n \nabla f_k \cdot (\nabla f_k)^T + \sum_{k=1}^n f_k \nabla^2 f_k. \quad (2.21)$$

Since the computation of $\nabla^2 f_k$ is usually too costly, the Gauss-Newton method uses only the first part in (2.21) to approximate the Hessian $\nabla^2 f$. This amounts to computing the minimizer x_+ of the linear least squares problem

$$\min \frac{1}{2} \sum_k [f_k(x_c) + \nabla(f_k(x_c))^T \cdot (x - x_c)]^2.$$

That is, a linear approximation of f_k is used in the Gauss-Newton method.

In the B-spline curve fitting problem, the update step $x_+ - x_c$ is given by the displacement vectors $\mathcal{D} = (D_1, \dots, D_m)$ of the m control points. From Equation. (2.18) and Equation. (2.19), since $\mathcal{F} = P(t_k) - X_k$, we have

$$\nabla f_k = -\mathcal{F}_{\mathcal{P}}^T N_k = -B^T(t_k)N_k.$$

Therefore the Gauss-Newton iteration for B-spline curve fitting performs iterative minimization of

$$f_{GN} = \frac{1}{2} \sum_k [f_k - \sum_i B_i(t_k)D_i^T N_k]^2,$$

which is interleaved with the step of foot point computation in order to satisfy the constraints (2.15). Since $N_k = (X_k - P(t_k))/\|P(t_k) - X_k\|$, we have $f_k = (X_k - P(t_k))^T N_k$.

Noting that $P(t_k) = \sum_i B_i(t_k)P_i$, we obtain

$$\begin{aligned}
 f_{GN} &= \frac{1}{2} \sum_k [(X_k - P(t_k))^T N_k - \sum_i B_i(t_k) D_i^T N_k]^2 \\
 &= \frac{1}{2} \sum_k [(X_k - \sum_i B_i(t_k)(P_i + D_i))^T N_k]^2 \\
 &= \frac{1}{2} \sum_k [(X_k - P_+(t_k))^T N_k]^2 = \frac{1}{2} \sum_k e_{TD,k}, \tag{2.22}
 \end{aligned}$$

where $e_{TD,k}$ is defined in Equation. (2.4). Hence, the minimization of the TD error function in (2.5) in TDM is equivalent to Gauss-Newton iteration.

The TD error term $e_{TD,k}$, unlike the SDM error term, counts for neither the distance from the data point X_k to the curve $P(t)$ nor the curvature of the curve $P(t)$, reflecting the fact that the Gauss-Newton method omits the term $f_k \nabla^2 f_k$ in (2.21).

Strictly speaking, TDM is not the standard Gauss-Newton method, since there is a step of foot point computation interleaved with Gauss-Newton iteration. In TDM the Gauss-Newton step is applied in the tangent plane to the constraint surface defined by the constraints (2.16). Such an algorithm is called a *variable projection method using a Gauss-Newton method* for solving a separable and constrained nonlinear least squares problem and can be shown [14, 124] to have the same asymptotic convergence behavior as the full Gauss-Newton method applied to all variables (i.e., \mathcal{P} and \mathcal{T} in our case).

TDM also shares the same framework of the so called *generalized reduced gradient (GRG) method* [87] for solving a nonlinear constraint problem with two separable groups of variables; the steepest gradient direction is used in the tangent plane of the constraint surface in the GRG method, while a Gauss-Newton step is used in TDM.

It is well known [74] that, in a Gauss-Newton method, if x_c is sufficiently close to the minimizer x^* of f , the distance $\|e_c\| = \|x_c - x^*\|$ of the current iterate to x^* is related to the error $\|e_+\|$ in the next iterate by

$$\|e_+\| \leq K(\|e_c\|^2 + \|R(x^*)\| \|e_c\|), \tag{2.23}$$

where $R(x^*) = (f_1, \dots, f_n)(x^*)$ is the residual at x^* , and K is a constant which involves the Jacobian of $R(x)$. It follows from (2.23) that for a zero residual problem Gauss-Newton iteration converges quadratically and the data points can be fitted exactly. Furthermore, Gauss-Newton iteration has fast convergence for good initial data and a small residual problem. For a large residual problem, the Gauss-Newton iteration may not converge at all.

Some variants of the Gauss-Newton method are possible. If only a scalar multiple of the Gauss-Newton step, $s(x_+ - x_c)$, usually with $0 < s < 1$, is used for stepping to the next solution, then one obtains the *damped Gauss-Newton method* [74].

Another way to modify Gauss-Newton is a regularization with the *Levenberg-Marquart method* [74], in which a scalar multiple of the identity matrix is added to the approximate Hessian. In our setting, this method requires the minimization of

$$f_{LM} = \frac{1}{2} \sum_k [(X_k - P_+(t_k))^T N_k]^2 + \nu_c \sum_i \|D_i\|^2.$$

Thus, the regularization term penalizes large changes D_i in the control points. It can be shown that using a regularization parameter ν_c of the order of the norm of the residual, i.e., $O(\|R(x_c)\|)$, one obtains still quadratic convergence for a zero residual problem. A drawback of the Levenberg-Marquart method in the setting of curve fitting is that the same magnitude of regularization is applied to every control point, without taking into account the curvature variation at different locations.

By writing the fitting error as a function of the parameter values t_k , the Levenberg-Marquart method is used in [125] to iteratively update the t_k , and faster convergence of this method than a variant of PDM is reported. However, although the foot point computation is avoided, it is noted in [125] that this variant of the L-M method is about 10 times slower than PDM per iteration.

A global Gauss-Newton method is implemented in [138] to update the control points P_i and the parameter values t_k together, therefore avoiding the costly step of computing foot points of data points. However, a relatively large linear system of equations needs to be solved, since now the parameter values of a large number of data points also enter optimization.

2.6.3 SDM – a quasi-Newton method

In this section we will derive the expression of the Newton method and then reveal the difference between our SDM scheme and the Newton method to show that SDM is, in fact, a quasi-Newton method. The key to this analysis is deriving a suitable expression of the Hessian of the objective function.

For a fixed k , consider the term $f_k^2 = \mathcal{F}^T \mathcal{F}$. Denote $\mathcal{F} = (\mathcal{F}_x, \mathcal{F}_y)^T$. By (2.17), we have $\nabla f_k^2 = 2\mathcal{F}_{\mathcal{P}}^T \mathcal{F}$. The derivative of ∇f_k^2 yields the Hessian

$$\begin{aligned} \frac{1}{2} \nabla^2 f_k^2 &= \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_{\mathcal{P}} + (\mathcal{F}_{\mathcal{P}t}^T \mathcal{F}_t + \mathcal{F}_{\mathcal{P}t}^T \mathcal{F}) \nabla t^T + \\ &\quad \nabla t (\mathcal{F}_t^T \mathcal{F}_{\mathcal{P}} + \mathcal{F}^T \mathcal{F}_{\mathcal{P}t}) + (\mathcal{F}_{tt}^T \mathcal{F} + \mathcal{F}_t^T \mathcal{F}_t) \nabla t \nabla t^T \\ &\quad + \mathcal{F}_x \mathcal{F}_{x\mathcal{P}\mathcal{P}} + \mathcal{F}_y \mathcal{F}_{y\mathcal{P}\mathcal{P}} \\ &= \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_{\mathcal{P}} + (\mathcal{F}_{\mathcal{P}t}^T \mathcal{F}_t + \mathcal{F}_{\mathcal{P}t}^T \mathcal{F}) \nabla t^T + \\ &\quad \nabla t (\mathcal{F}_t^T \mathcal{F}_{\mathcal{P}} + \mathcal{F}^T \mathcal{F}_{\mathcal{P}t}) + (\mathcal{F}_{tt}^T \mathcal{F} + \mathcal{F}_t^T \mathcal{F}_t) \nabla t \nabla t^T \end{aligned} \quad (2.24)$$

Here we used the fact that $\mathcal{F}_{x\mathcal{P}\mathcal{P}} = 0, \mathcal{F}_{y\mathcal{P}\mathcal{P}} = 0$, since \mathcal{F} is linear in \mathcal{P} .

Again, ∇t stands for the gradient of $t_k = t(\mathcal{P})$ with respect to \mathcal{P} .

On the other hand, we need to find the relationship between ∇t and $\mathcal{F}_{\mathcal{P}}$. Differentiating the constraint (2.16), we obtain

$$(\mathcal{F}_{\mathcal{P}t}^T + \nabla t \mathcal{F}_{tt}^T) \mathcal{F} + (\mathcal{F}_{\mathcal{P}}^T + \nabla t \mathcal{F}_t^T) \mathcal{F}_t = 0. \quad (2.25)$$

Solving for ∇t yields

$$\nabla t = -\frac{\mathcal{F}_{\mathcal{P}t}^T \mathcal{F} + \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_t}{\mathcal{F}_{tt}^T \mathcal{F} + \mathcal{F}_t^T \mathcal{F}_t}.$$

Substituting this expression of ∇t in (2.24), we obtain the complete Hessian

$$\frac{1}{2} \nabla^2 f_k^2 = \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_{\mathcal{P}} - \frac{(\mathcal{F}_{\mathcal{P}t}^T \mathcal{F} + \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_t)(\mathcal{F}_t^T \mathcal{F}_{\mathcal{P}} + \mathcal{F}^T \mathcal{F}_{\mathcal{P}t})}{\mathcal{F}_{tt}^T \mathcal{F} + \mathcal{F}_t^T \mathcal{F}_t}.$$

We will now make a simplification by neglecting the term $\mathcal{F}_{\mathcal{P}t}^T \mathcal{F}$, i.e., setting it to zero. This results in an approximate Hessian $\tilde{\nabla}^2 f_k^2$. To interpret this approximate Hessian geometrically, we introduce the arc length parameter s of the B-spline curve $P(t)$. Then we have

$$\begin{aligned} \frac{1}{2} \tilde{\nabla}^2 f_k^2 &= \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_{\mathcal{P}} - \frac{\mathcal{F}_{\mathcal{P}}^T \mathcal{F}_t \mathcal{F}_t^T \mathcal{F}_{\mathcal{P}}}{\mathcal{F}_{tt}^T \mathcal{F} + \mathcal{F}_t^T \mathcal{F}_t} = \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_{\mathcal{P}} - \frac{(s_t)^2 \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_s \mathcal{F}_s^T \mathcal{F}_{\mathcal{P}}}{[\mathcal{F}_{ss}^T (s_t)^2 + \mathcal{F}_s^T s_{tt}] \mathcal{F} + \mathcal{F}_s^T \mathcal{F}_s (s_t)^2} \\ &= \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_{\mathcal{P}} - \frac{(s_t)^2 \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_s \mathcal{F}_s^T \mathcal{F}_{\mathcal{P}}}{\mathcal{F}_{ss}^T \mathcal{F} (s_t)^2 + \mathcal{F}_s^T \mathcal{F}_s (s_t)^2} = \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_{\mathcal{P}} - \frac{\mathcal{F}_{\mathcal{P}}^T \mathcal{F}_s \mathcal{F}_s^T \mathcal{F}_{\mathcal{P}}}{(\mathcal{F}_{ss}^T \mathcal{F} + \mathcal{F}_s^T \mathcal{F}_s)}. \end{aligned}$$

In the above, the term $\mathcal{F}_s^T \mathcal{F}_s s_{tt}$ drops out due to the constraint (2.16) and the fact that \mathcal{F}_s^T and \mathcal{F}_t^T are collinear.

Clearly, $\mathcal{F}_s^T \mathcal{F}_s = 1$, since $\mathcal{F}_s = T_k$ is the unit tangent vector of $P(t)$. Since \mathcal{F}_{ss} is the curvature vector of $P(t)$ at $P(t_k)$, we have $\mathcal{F}_{ss}^T \mathcal{F} = -d\kappa$, where $\kappa \geq 0$ is the curvature

and d is the signed distance defined in Section 2.3. Hence, we obtain

$$\frac{1}{2}\tilde{\nabla}^2 f_k^2 = \mathcal{F}_{\mathcal{P}}^T \mathcal{F}_{\mathcal{P}} - \frac{\mathcal{F}_{\mathcal{P}}^T T_k T_k^T \mathcal{F}_{\mathcal{P}}}{-d\kappa + 1}. \quad (2.26)$$

Noting that $T_k T_k^T + N_k N_k^T = I$, this equation is further rewritten as

$$\begin{aligned} \frac{1}{2}\tilde{\nabla}^2 f_k^2 &= \mathcal{F}_{\mathcal{P}}^T (I - T_k T_k^T) \mathcal{F}_{\mathcal{P}} - \frac{d\kappa \mathcal{F}_{\mathcal{P}}^T T_k T_k^T \mathcal{F}_{\mathcal{P}}}{-d\kappa + 1} \\ &= \mathcal{F}_{\mathcal{P}}^T N_k N_k^T \mathcal{F}_{\mathcal{P}} + \frac{d}{d - \rho} \mathcal{F}_{\mathcal{P}}^T T_k T_k^T \mathcal{F}_{\mathcal{P}}. \end{aligned} \quad (2.27)$$

Now we consider the relationship between SDM and the quasi-Newton method obtained above by replacing the Hessian $\nabla^2 f_k^2$ by $\tilde{\nabla}^2 f_k^2$. Note that

$$\mathcal{F}_{\mathcal{P}}^T N_k N_k^T \mathcal{F}_{\mathcal{P}} = \nabla f_k (\nabla f_k)^T,$$

which is the first term in (2.21) that is used by Gauss-Newton iteration to approximate the true Hessian. Thus, replacing the Hessian $\nabla^2 f_k^2$ by $\tilde{\nabla}^2 f_k^2$ in the Newton method is equivalent to adding the second term in (2.27) to the Gauss-Newton method to yield a quasi-Newton method. Recall that the Gauss-Newton method is the same as TDM. Therefore, noting that $(P(t_k) - X_k)^T T_k = 0$ (by (2.15)), the above quasi-Newton method minimizes the quadratic function

$$\begin{aligned} f_{QN} &= f_{GN} + \frac{1}{2} \sum_{k=1}^n \mathcal{D} \left(\frac{d_k}{d_k - \rho_k} \mathcal{F}_{\mathcal{P}}^T T_k T_k^T \mathcal{F}_{\mathcal{P}} \right) \mathcal{D}^T \\ &= f_{GN} + \frac{1}{2} \sum_{k=1}^n \frac{d_k}{d_k - \rho_k} \left[\left(\sum_i B_i(t_k) D_i \right)^T T_k \right]^2 \\ &= \frac{1}{2} \sum_{k=1}^n \left\{ \left[(P_+(t_k) - X_k)^T N_k \right]^2 + \frac{d_k}{d_k - \rho_k} \left[(P_+(t_k) - P(t_k))^T T_k \right]^2 \right\} \\ &= \frac{1}{2} \sum_{k=1}^n \left\{ \left[(P_+(t_k) - X_k)^T N_k \right]^2 + \frac{d_k}{d_k - \rho_k} \left[(P_+(t_k) - X_k)^T T_k \right]^2 \right\} = \frac{1}{2} \sum_{k=1}^n h_k(\mathcal{D}), \end{aligned}$$

where $h_k(\mathcal{D})$ is defined in (2.9) and d_k, ρ_k are the corresponding distance and curvature radius for the k -th term. Hence, the quasi-Newton method uses a local quadratic model f_{QN} that is the same as the quadratic approximant $h_k(\mathcal{D})$ used by the SD error term before it is turned into the semi-definite form in (2.10). Hence, we have shown that the SDM method is a quasi-Newton method obtained by discarding the term $\mathcal{F}^T \mathcal{F}_{\mathcal{P}t}$, which amounts to disregarding the change $\mathcal{F}_{\mathcal{P}t}$ of the tangent vector $P'_t(t_k)$ caused by the change of the control points.

SDM does not fall into the category of the quasi-Newton methods that fulfill the so-called secant equation [74]. Instead, SDM uses another positive definite approximant of the Hessian, based on geometric considerations. Although SDM is not a standard optimization procedure, it is a computationally attractive and effective compromise between a full Newton scheme and Gauss-Newton – it picks up more contributions of the true Hessian than the Gauss-Newton iteration (or TDM) does, but ignores the remaining part for reasons of computational efficiency and simplicity. Indeed, SDM is an optimization scheme that is particularly suited for solving shape fitting problems, because it uses an intuitively simple error metric that is adaptable to local curvature variation of a target shape.

Each term f_k^2 in the objective function in (2.14) is the squared distance function. Because the exact Hessian in the second order Taylor expansion of f_k^2 is modified in order to derive the SD error term, we conclude that, in general, the SD error term is *not* a second order approximation to the squared distance, even without the modification in (2.10) to make the error term positive semi-definite. This is in contrast to the previous squared distance minimization method by Pottmann et al. [116] where the quadratic approximant in (2.6) is always a second order approximation to the squared distance function, before the modification to turn it into the positive semi-definite error term in (2.7).

2.6.4 Step size control

We have tested step size control on PDM, TDM, and SDM, using the Armijo rule [74]. It is found that step size control does not help much with PDM and SDM — PDM still converges slowly, and the per-iteration computation of SDM becomes much longer with moderate degree of improvement in stability. It is found that the stability of TDM improves greatly with the help of step size control, however, at the cost of much longer per-iteration time, especially when approaching a local minimum, since, due to the “flat” gradient near a local minimum, it normally gets more time-consuming to select an appropriate step size via repetitive evaluations of the fitting error.

Figure 2.17 shows the result of applying step size control (the Armijo rule) to TDM on the same data points and initial B-spline curves as shown in Figures 2.7(a) and 2.10(a); now both data sets are satisfactorily approximated by B-spline curves computed with TDM. For comparison, refer to Figures 2.7(c) and 2.10(c) to see the unacceptable fitting curves generated by TDM without step size control.

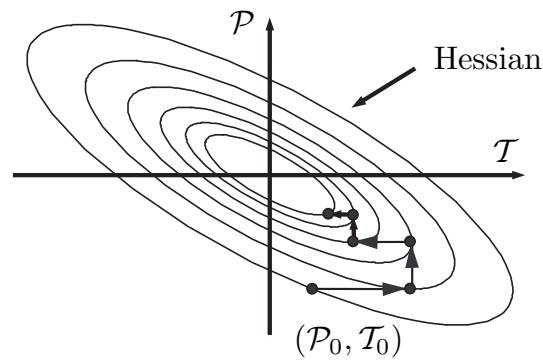


FIGURE 2.16: The alternating minimization steps of PDM near a local minimum. \mathcal{P} and \mathcal{T} stand for the linear subspaces spanned by the control points and the data parameters, respectively.

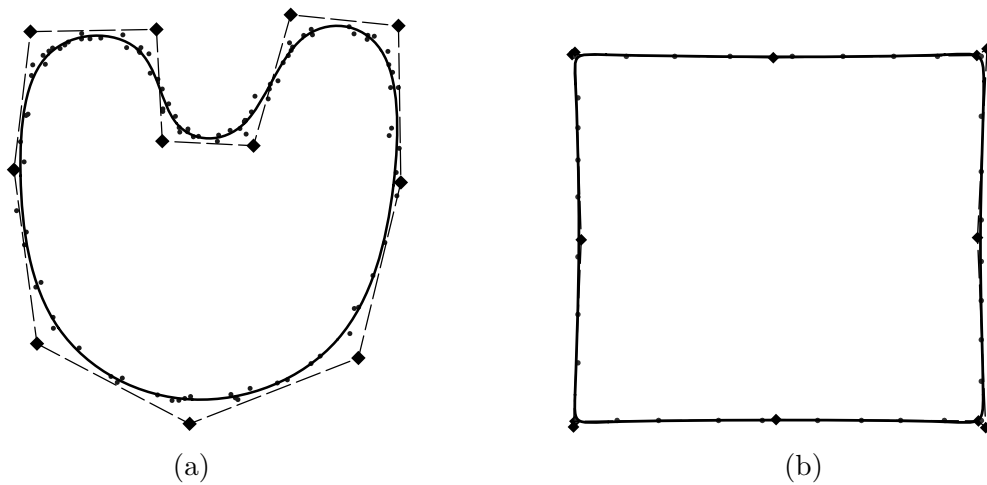


FIGURE 2.17: TDM using the Armijo rule for step size control. (a) The fitting curve by TDM in 20 iterations for the data set in Figure 2.7(a); (b) The fitting curve by TDM in 20 iterations for the data set in Figure 2.10(a).

2.7 Concluding Remarks

PDM is widely used in practice for parametric curve and surface fitting [52]. As we have shown, PDM has linear convergence in theory, converges slowly in practice, and is often trapped in a poor local minimum. TDM is a method often used for curve fitting in computer vision (e.g., [15]). TDM converges faster than PDM, but its convergence is highly unstable. Against this backdrop, we have proposed a novel and efficient method, called SDM, for fitting B-spline curves to point cloud data. We have shown empirically that SDM converges faster than PDM and that SDM is more stable than TDM. In addition, SDM is easy to implement and has similar per-iteration computation time to PDM and TDM, since they share the same framework. All this suggests that SDM is a favorable alternative to PDM or TDM for B-spline curve fitting.

In order to gain a better understanding of the above geometrically motivated schemes, we have studied the B-spline curve fitting problem from the optimization viewpoint. We note that PDM is a gradient descent method in a metric that is not well chosen. We have also shown that TDM uses a Gauss-Newton step for solving a nonlinear least squares problem, and its instability at high curvature regions is thus due to its omission of important parts in the true Hessian of the objective function and the lack of step size control. Finally, we have shown that our proposed SDM scheme is a quasi-Newton method using a carefully chosen approximate Hessian, and thus its superior performance in both convergence and stability does not come as a surprise. Interestingly, unlike most other quasi-Newton methods, the approximate Hessian used by SDM is not explicitly computed; it arises naturally as the consequence of using the simple SDM error term devised out of entirely geometric considerations, i.e., making use of curvature information to give a close approximation of the squared distance function. This contributes to the simplicity and efficiency of SDM. One of direct applications is subdivision surface fitting [31], which shows the superiority and effectiveness of SDM.

Claim: the presented material in this chapter has been published in [151].

Chapter 3

Constrained 3D Shape Reconstruction Using a Combination of Surface Fitting and Registration

3.1 Introduction

The motivation for the present research comes from reconstruction of objects from 3D scanner data, where special kinematic surfaces (cones, cylinders, general surfaces of revolution, helical surfaces) appear frequently. Many reconstruction algorithms for the more general representatives of these surface classes require estimated surface normals [115, 146]. Although these methods are quite efficient when good normal estimates are available, they lack the desired precision if it is difficult to obtain accurate normal estimation or the deviation of the data from the ideal shape model is relatively large; an example is the reconstruction of vessels from archeological findings. Moreover, in these methods the computation of the sweeping motion is separated from the computation of the swept profile, which is a further source of errors.

In this chapter, we extend recent work on improved reconstruction of surfaces of revolution [155] with a more generally applicable concept arising from the geometric optimization framework of *squared distance minimization (SDM)* (cf. Chapter 2). Our new method combines the two types of optimization problems that have been solved so far with SDM, namely *curve/surface fitting* and *registration*. This new approach is not only

applicable to surfaces of revolution but also to other classes of surfaces and to a number of surface reconstruction problems in reverse engineering in the presence of constraints.

3.1.1 Previous work

Since the focus of the present work is on *constrained* 3D shape reconstruction, we only review research in this direction. A constraint may fix the surface type: there have been a considerable number of contributions to fitting with special surfaces and thus we refer to [146] for a detailed survey. Existing methods are mainly taken from geometry (Gaussian image, line geometry and kinematical geometry), image processing (methods in extension of the Hough transform) and optimization (non-linear least squares problems). They are also used for surface type recognition (shape filters).

Fitting data with a surface of a given type that is determined with appropriate shape filters, while maintaining constraints between the individual elements of the surface, is a challenging problem [146]. Not only do we need to check the consistency of the constraints, we also need to fit the data simultaneously under these constraints. The work of Benkő et al. [11], Fisher [53], and Karniel et al. [73] can be considered to constitute the state of the art in this area. In the actual fitting part of the problem, most authors use a least squares formulation which embeds the constraints via penalty terms.

Our research is based on a combination of registration and fitting, and in this sense closely related to the work on knowledge based image segmentation via a combination of registration and active contours [96, 103, 147] and to deformable models introduced by Terzopoulos and Fleischer [141]. We also present a new solution for the simultaneous treatment of multiple view registration and model fitting, which extends prior work by Jin et al. [71] and Tubic et al. [143].

3.1.2 Contributions

Our contributions are:

- the extension of the SDM method to surface approximation with error measurement orthogonal to the fitting surface;
- the combination of registration and surface fitting within the SDM framework;
- refined algorithms for fitting with kinematic surfaces (rotational, helical and spiral surfaces) plus a demonstration of their efficiency for shape reconstruction from measurement data of archeological pottery, shells and engineering objects;

- a new way of incorporating constraints into 3D surface reconstruction for applications in reverse engineering of CAD models;
- an efficient optimization algorithm which combines multiple view registration and model fitting and in this way achieves higher accuracy than the traditional approach which first registers the data and then fits a model to it.

3.2 Fundamentals of SDM

Here we summarize a few basic facts about squared distance minimization (SDM). For more details and issues of efficient implementation we refer to [6, 112, 117, 158] and Chapter 2. Before entering this discussion we would like to point out that many authors have used the distance field [38, 78] for registration and fitting; in fact, the concept of the distance field is so closely tied to the problem that it must occur in some way. However, most papers do not use the distance function in the same way as we are doing it: we use local quadratic approximants of the squared distance function and in this way obtain fast local convergence via algorithms of the Newton or quasi-Newton type.

3.2.1 Squared distance function of a surface

Given a surface $\Phi \subset \mathbb{R}^3$, the squared distance function d^2 assigns to each point $\mathbf{x} \in \mathbb{R}^3$ the square of its shortest distance to Φ . The importance of this function for our algorithms lies in the fact that we want to compute a surface which minimizes the sum of squared distances to the data point cloud. Since several important optimization concepts require second order approximants of the objective function, we need to derive second order approximants of d^2 .

Let us fix the notation. We consider a surface Φ with unit normal vector field $\mathbf{n}(\mathbf{s}) = \mathbf{n}_3(\mathbf{s})$, attached to points $\mathbf{s} \in \Phi$. At each point \mathbf{s} , we have a local Cartesian frame $(\mathbf{n}_1, \mathbf{n}_2, \mathbf{n})$, where the first two vectors $\mathbf{n}_1, \mathbf{n}_2$ determine the principal curvature directions. We will refer to this local frame as the *principal frame* $\Pi(\mathbf{s})$. Let κ_j be the (signed) principal curvature in the principal curvature direction \mathbf{n}_j , $j = 1, 2$, and let $\rho_j = 1/\kappa_j$.

Let $\mathbf{s} \in \Phi$ be the normal foot point of a point $\mathbf{p} \in \mathbb{R}^3$, i.e., \mathbf{s} is the closest point on Φ to \mathbf{p} . Expressed in the principal frame at \mathbf{s} the second order Taylor approximant F_d of the function d^2 at a point $\mathbf{x} \in \mathbb{R}^3$ in a neighborhood of \mathbf{p} is

$$F_d(\mathbf{x}) = \frac{d}{d - \rho_1} [\mathbf{n}_1 \cdot (\mathbf{x} - \mathbf{s})]^2 + \frac{d}{d - \rho_2} [\mathbf{n}_2 \cdot (\mathbf{x} - \mathbf{s})]^2 + [\mathbf{n}_3 \cdot (\mathbf{x} - \mathbf{s})]^2. \quad (3.1)$$

Here, $[\mathbf{n}_j \cdot (\mathbf{x} - \mathbf{s})]^2$, $j = 1, 2, 3$, are the squared distances of \mathbf{x} to the principal planes and tangent plane at \mathbf{s} , respectively.

In the important special case of $d = 0$ (i.e., $\mathbf{p} = \mathbf{s}$), the approximant F_d equals the squared distance function to the tangent plane of Φ at \mathbf{s} . Thus, if \mathbf{p} is close to Φ , the squared distance function to the tangent plane at \mathbf{p} 's closest point on Φ is a good approximant of d^2 .

In a Newton-like iteration it is important to employ nonnegative quadratic approximants; we obtain them by removing from the expression of $F_d(\mathbf{x})$ in (3.1) those terms with a negative coefficient $d/(d - \rho_j)$.

3.2.2 Registration using SDM

A set of points $X^0 = (\mathbf{x}_1^0, \mathbf{x}_2^0, \dots) \subset \mathbb{R}^3$ is given in some coordinate system Σ^0 . It will be rigidly moved (i.e., registered) to be in best alignment with a given surface Φ , represented in another system Σ . We view Σ^0 and Σ as a moving system and a fixed system, respectively. A position of X^0 in Σ is denoted by $X = (\mathbf{x}_1, \mathbf{x}_2, \dots)$. It is the image of X^0 under some rigid body motion Υ . Since we identify positions with motions and the motions have to act on the same initial position, we write $X = \Upsilon(X^0)$, or $\mathbf{x}_i = \Upsilon(\mathbf{x}_i^0)$.

The *registration problem* is formulated in a least squares sense [13, 30]: Compute a rigid body transformation Υ^* , which minimizes the sum of squared distances of $\Upsilon(\mathbf{x}_i^0)$ to Φ ,

$$F(\Upsilon) = \sum_i d^2(\Upsilon(\mathbf{x}_i^0), \Phi). \quad (3.2)$$

Starting from an appropriate initial position Υ^0 , SDM performs a Newton-like iteration to minimize F [117]. We describe here a single iteration of the algorithm: Since F is the sum of squared distances of the data points \mathbf{x}_i to the model shape Φ , a quadratic approximant is

$$G = \sum_i F_{d,i}, \quad (3.3)$$

where the $F_{d,i}$ are the second order approximants of the squared distance functions of \mathbf{x}_i to the model shape. These approximants have been described in Section 3.2.1. Then, by equation (3.1), a second order Taylor approximant of the squared distance function at \mathbf{x}_i is written in the form

$$F_{d,i}(\mathbf{x}) = \sum_{j=1}^3 \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\mathbf{x} - \mathbf{s}_i)]^2, \quad (3.4)$$

where $\mathbf{n}_{i,j} \cdot (\mathbf{x} - \mathbf{s}_i) = 0$, $j = 1, 2, 3$, denote the coordinate planes of the principal frame at the foot point $\mathbf{s}_i \in \Phi$ of the point \mathbf{x}_i^0 , and the $\alpha_{i,j}$ can readily be read from equation (3.1). The same form holds for a nonnegative modification, i.e., terms with negative coefficients will be discarded. One now approximates the displacement of the data point \mathbf{x}_i up to the first order by,

$$\mathbf{x}'_i = \mathbf{x}_i^0 + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i^0, \quad (3.5)$$

where $\bar{\mathbf{c}} = (\bar{c}_1, \bar{c}_2, \bar{c}_3)$ and $\mathbf{c} = (c_1, c_2, c_3)$ represent the translational and rotational components of a velocity field.

Plugging \mathbf{x}'_i into G in equation (3.3) gives a local quadratic model of the objective function,

$$F_2(\mathbf{c}, \bar{\mathbf{c}}) = \sum_i \sum_{j=1}^3 \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\mathbf{x}_i^0 + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i^0 - \mathbf{s}_i)]^2.$$

Since $\mathbf{n}_{i,j} \cdot (\mathbf{x}_i^0 - \mathbf{s}_i)$ is the distance of \mathbf{x}_i^0 to the j -th coordinate plane of the principal frame, it vanishes for $j = 1, 2$; and it equals the oriented distance d_i of \mathbf{x}_i^0 to the model surface Φ for $j = 3$. Therefore we may rewrite F_2 as

$$F_2(\mathbf{c}, \bar{\mathbf{c}}) = \sum_i \sum_{j=1}^2 \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i^0)]^2 + \tilde{F}_2(\mathbf{c}, \bar{\mathbf{c}}). \quad (3.6)$$

Here, \tilde{F}_2 denotes the part arising from the squared distances to the tangent planes at the foot points, given by

$$\tilde{F}_2(\mathbf{c}, \bar{\mathbf{c}}) = \sum_i [\mathbf{n}_i \cdot (\bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i^0) + d_i]^2, \quad (3.7)$$

where $\mathbf{n}_i = \mathbf{n}_{i,3}$. Since F_2 is a quadratic function in $(\mathbf{c}, \bar{\mathbf{c}})$, the unique minimizer $(\mathbf{c}', \bar{\mathbf{c}}')$ can be given explicitly by solving a system of linear equations.

Remark 3.1. In the above application of SDM we measure the squared distance errors from the moving points \mathbf{x}_i orthogonal to the fixed model surface Φ . The moving points \mathbf{x}_i are functions of the motion parameters $(\mathbf{c}, \bar{\mathbf{c}})$ to be optimized.

So far we have estimated the displacement of the data point cloud with help of the velocity field $(\mathbf{c}, \bar{\mathbf{c}})$. We now apply an appropriate helical motion which is determined by this velocity field: The Plücker coordinates $(\mathbf{g}, \bar{\mathbf{g}})$ of the axis, the rotational angle ϕ and the pitch p of the helical motion (including special cases) are given by

$$p = (\mathbf{c} \cdot \bar{\mathbf{c}})/(\mathbf{c})^2, \quad \phi = \|\mathbf{c}\|, \quad (\mathbf{g}, \bar{\mathbf{g}}) = (\mathbf{c}, \bar{\mathbf{c}} - p\mathbf{c}). \quad (3.8)$$

Recall that the *Plücker coordinates* of a line G consist of a direction vector \mathbf{g} and the moment vector $\bar{\mathbf{g}} = \mathbf{p} \times \mathbf{g}$, where \mathbf{p} represents an arbitrary point on G . Altogether, the

desired motion is the superposition of a rotation about some axis A through an angle of $\phi = \|\mathbf{c}\|$ and a translation parallel to A by the distance of $p \cdot \phi$. For the explicit formulae we refer to the literature [115].

3.2.3 SDM for B-spline surface fitting

In this subsection, we describe the basic idea of B-spline curve fitting according to Chapter 2. Different from the SDM introduced in Section 3.2.1 and 3.2.2 (ref. Remark 3.1), this method measures the fitting error orthogonal to a moving fitting surface. But, since the fitting error is also given by a quadratic approximation of the squared distance to the fitting surface, we shall also refer to the method as *squared distance minimization*, or *SDM*. Although the method has been presented in Chapter 2 for B-spline curves only, its generalization to fitting a B-spline surface to a point cloud $(\mathbf{x}_1, \dots, \mathbf{x}_N)$ is straightforward and outlined below.

The main steps are as follows.

- (1) Specify a proper initial shape of a B-spline fitting surface.
- (2) Compute squared-distance error terms for all data points to obtain a local quadratic model of the objective function.
- (3) Solve a linear system of equations to optimize the local quadratic model to obtain an updated spline surface.
- (4) Repeat steps 2 and 3 until convergence, e.g., until a pre-specified error threshold is satisfied or the incremental change of the control points falls below a preset threshold.

We explain below briefly steps 2 and 3.

Step 2. An error term is associated with each data point \mathbf{x}_i . The error term to be used in the next iteration is found as follows. Compute a (nonnegative) second order approximant of the squared distance function from \mathbf{x}_i to the current instance of the fitting surface Φ ; see equation (3.4). Let $\mathbf{s}_i = \mathbf{s}_c(u_i, v_i)$ be the closest point on the fitting surface to \mathbf{x}_i . Then the error term for \mathbf{x}_i is

$$E_{i,c} = \sum_{j=1}^3 \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\mathbf{x}_i - \mathbf{s}_c(u_i, v_i))]^2. \quad (3.9)$$

When we update the surface Φ to $\mathbf{s}_+(u, v)$ with the new control points, the surface point $\mathbf{s}_+(u_i, v_i)$ given by the same parameters (u_i, v_i) is, in general, no longer the foot point

of \mathbf{x}_i ; moreover, the normal vectors $\mathbf{n}_{i,j}$ and curvature radii $\rho_{i,j}$ will have changed there. However, when the model surface is updated by a small change of the control points, we still use equation (3.9) to estimate the new fitting error at \mathbf{x}_i , by

$$E_{i,+}(\mathbf{D}) = \sum_{j=1}^3 \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\mathbf{x}_i - \mathbf{s}_+(u_i, v_i))]^2, \quad (3.10)$$

where the variables are the control points $\mathbf{D} := (\mathbf{d}_1, \dots, \mathbf{d}_m)$ of the B-spline fitting surface, in the expressions of the updated surface points $\mathbf{s}_+(u_i, v_i)$; we use $E_{i,+}(\mathbf{D})$ to emphasize the dependence of the error term on the control points \mathbf{D} . It has been shown in Chapter 2 that this simplification yields a quasi-Newton method for optimization, which is not of a standard type (such as BFGS [74]), but provides a very good trade-off between computational simplicity and fast convergence.

Step 3. We use a B-spline surface, whose representation of the form $\mathbf{s}(u, v) = \sum_k B_k(u, v) \mathbf{d}_k$ is linear in the control points. Substituting this form for $\mathbf{s}_+(u_i, v_i)$ in the objective function yields

$$\begin{aligned} F(\mathbf{D}) &= \sum_{i=1}^N E_{i,+}(\mathbf{D}) + F_s(\mathbf{D}) \\ &= F_s(\mathbf{D}) + \sum_{i=1}^N \sum_{j=1}^3 \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\mathbf{x}_i - \sum_k B_k(u_i, v_i) \mathbf{d}_k)]^2. \end{aligned} \quad (3.11)$$

Here, F_s is a smoothing term, assumed to be quadratic in \mathbf{D} . The part coming from the sum of squared distances is also quadratic in the unknown control points \mathbf{D} . Therefore the minimization of F requires only the solution of a linear system.

3.2.4 TDM and PDM

When setting $\alpha_{i,1} = \alpha_{i,2} = 0$ and $\alpha_{i,3} = 1$ in Equation. (3.4) (or Equation. (3.10)), we obtain the *tangent distance minimization* or *TDM*, since $F_{d,i}(\mathbf{x}) = [\mathbf{n}_i(\mathbf{x} - \mathbf{s}_i)]^2$ which measures the squared distance from \mathbf{x} to the tangent plane at \mathbf{s}_i .

When $\alpha_{i,1} = \alpha_{i,2} = \alpha_{i,3} = 1$ in Equation. (3.4) (or Equation. (3.10)), we obtain $F_{d,i}(\mathbf{x}) = \|\mathbf{x} - \mathbf{s}_i\|^2$, which measures the distance between the two points \mathbf{x} and \mathbf{s}_i ; hence, the resulting minimization scheme is called the *point distance minimization* or *PDM*. A detailed discussion of these two minimization methods and their comparison with SDM can be found in Chapter 2.

SDM is simplified to the *TDM* method if we approximate the function d^2 at a point \mathbf{x}_i by the squared distance to the tangent plane at the foot point \mathbf{y}_i . For registration, a

method similar to TDM has been first proposed by Chen and Medioni [30] and is known to be superior to the standard ICP [13]. For B-spline curve fitting, the TDM method has been described by Blake and Isard [15]. It is known [117, 151] that TDM corresponds to a Gauss-Newton iteration. Thus, it exhibits quadratic convergence for a zero residual problem and a good initial position. However, this is not a practical assumption and thus one should enhance its stability by applying step-size control, e.g., using the Armijo rule or the Levenberg-Marquardt (L-M) regularization [74].

The standard ICP algorithm [13] approximates d^2 at \mathbf{x}_i by the squared distance to the foot point \mathbf{s}_i , i.e., the error term for \mathbf{x}_i is defined in PDM as $\|\mathbf{x}_i - \mathbf{s}_+(u_i, v_i)\|^2$ (ref. Equation. (3.10)). This *PDM* method is frequently used for freeform surface fitting [146], exhibits only linear convergence and is prone to be trapped in a poor local minimizer; see, e.g., [151].

3.3 Combination of Surface Fitting and Registration

Before entering the general discussion, let us explain the main idea with the following example: We want to fit a surface of revolution to a set of data points.

The standard solution to this problem uses estimated surface normals at the data points and line geometry to compute the rotational axis [115]. The axis is then kept fixed and an appropriate generatrix is computed to obtain the final surface. This method works very well for finding a good initial guess of the axis, but has the disadvantage that the error in the axis estimate, which arises from the normal estimates, cannot be further reduced in the subsequent computation of the generatrix. We present here the following idea.

After an initial guess of the axis A has been found, use a coordinate system in which A is a coordinate axis, say the x_3 -axis. A surface of revolution with this axis takes a very simple form. Then, we use SDM optimization to simultaneously update the fitting surface (i.e., control points of its generatrix) and move the set of data points (as a single rigid body system) until the fitting error is minimized. This is carried out by combining the techniques in Section 3.2.2. and Section 3.2.3. Moving the data point cloud is a registration process and equivalent to changing the axis. However, we register the data point cloud to a changing surface rather than a fixed one.

More specifically, let us explain the registration of a point cloud to a changing surface \mathbf{s} . In each iteration, for each data point \mathbf{x}_i^0 , we compute its closest point $\mathbf{s}_{i,c} \equiv \mathbf{s}_c(u_i, v_i)$

on the current instance of the fitting surface $\Phi : \mathbf{s}(u, v)$, and set up the SD error term,

$$E_{i,c} = \sum_{j=1}^3 \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\mathbf{x}_i - \mathbf{s}_{i,c})]^2.$$

The error after a small displacement of the data point set and a change of the surface is estimated as follows. We use a linearization for the displacement of the data point set, i.e., \mathbf{x}_i will be approximated by $\mathbf{x}' = \mathbf{x}_i^0 + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i^0$; the change of the control points updates the model surface to $\mathbf{s}_+(u, v)$; and therefore the surface points $\mathbf{s}_{i,c}$ will be replaced by $\mathbf{s}_{i,+} \equiv \mathbf{s}_+(u_i, v_i)$. Then, the new error term, as an approximation to the squared distance from \mathbf{x}_i and Φ , is

$$E_{i,+} = \sum_{j=1}^3 \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\mathbf{x}_i^0 + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i^0 - \mathbf{s}_{i,+})]^2, \quad (3.12)$$

where the variables are the motion parameters $(\mathbf{c}, \bar{\mathbf{c}})$ and the control points \mathbf{D} . A sum of these error terms, together with a quadratic fairness term F_s ,

$$F = \sum_{i=1}^N \sum_{j=1}^3 \alpha_{i,j} [\mathbf{n}_{i,j} \cdot (\mathbf{x}_i^0 + \bar{\mathbf{c}} + \mathbf{c} \times \mathbf{x}_i^0 - \mathbf{s}_{i,+})]^2 + F_s(\mathbf{D}), \quad (3.13)$$

gives rise to the minimization of a quadratic function in each iteration. Since the surface points $\mathbf{s}_{i,+}$ depend linearly on the unknown shape parameters, i.e., control points, F is quadratic in those unknowns as well as $(\mathbf{c}, \bar{\mathbf{c}})$, assuming that F_s is also a quadratic function of \mathbf{D} . Of course, the data point cloud is updated with an appropriate helical motion as for pure registration. We note that this method is applicable even when shape parameters are not linear variables, such as the weights in a rational B-spline surface – one just needs to apply a linearization (see Section 3.4.2), like using $(\mathbf{c}, \bar{\mathbf{c}})$ to approximate a rigid motion.

Due to misalignment, multiple view registration for a 3D object usually introduces errors not present in the measurement data. These errors would affect the subsequent surface fitting errors, thus the precision of the final reconstructed CAD model of the 3D object.

Within the present setting, we may use an initially registered point cloud for the initial steps in the constrained fitting procedure. In later steps, however, we can allow *a different motion for each of the K scans* that have been used. In this way we hope to remove inaccuracies resulting from the initial registration (see [71, 143]). Given K point clouds $X_k = \{\mathbf{x}_{k,i}^0, i = 1, \dots, N_k\}, k = 1, \dots, K$, which represent the individual scans

(views), our SDM objective function is reformulated as

$$F = \sum_{k=1}^K w_k \sum_{i=1}^{N_k} \sum_{j=1}^3 \alpha_{k,i,j} [\mathbf{n}_{k,i,j} \cdot (\mathbf{x}_{k,i}^0 + \bar{\mathbf{c}}_k + \mathbf{c}_k \times \mathbf{x}_{k,i}^0 - \mathbf{s}_{k,i,+})]^2 + F_s(\mathbf{D}), \quad (3.14)$$

where w_k is a weight for the points in each scan (in practice, we can choose $w_k = 1/N_k$); $\alpha_{k,i,j}$ and $\mathbf{n}_{k,i,j}$ have the same meaning as in (3.13); $(\mathbf{c}_k, \bar{\mathbf{c}}_k)$ is the velocity field of X_k . We will show some examples in Section 3.4.3.2.

The new version of the SDM method — a combination of fitting and registration — is as simple as the previously discussed cases of pure surface fitting or registration. Its performance from the viewpoint of optimization is comparable to that of pure fitting. As in Chapter 2, TDM with step size control also has good performance.

3.4 Applications

In this section we present three applications of our combined framework of registration and fitting to: 1) surface reconstruction from archeological pottery; 2) shell shape model verification; and 3) constrained CAD model reconstruction in reverse engineering. In all these applications we need to fit a surface of a special type to a given set of 3D scanned data points. For the convergence analysis we use the average error defined by the following *root mean squared error* :

$$Ave_Error = \sqrt{\frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i - \mathbf{s}_{i,c}\|^2} \quad (3.15)$$

The data set is first normalized by uniform scaling to fit in the unit box $[0, 1]^3$, to make the optimization parameters independent of model dimensions.

3.4.1 Surfaces of revolution

In order to fit a surface of revolution to a set of data points, we use the well-known line geometric method to compute an initial guess [115]. The axis is then used as x_3 -axis of the coordinate system. For the generatrix we take a B-spline curve, $(r(u), z(u)) = \sum_k (r_k, z_k) B_k(u)$ with control points $\mathbf{p}_k = (r_k, z_k)$. Then the surface is

$$\mathbf{x}(u, v) = \sum_k (r_k \cos v, r_k \sin v, z_k) B_k(u) + (0, 0, pv).$$

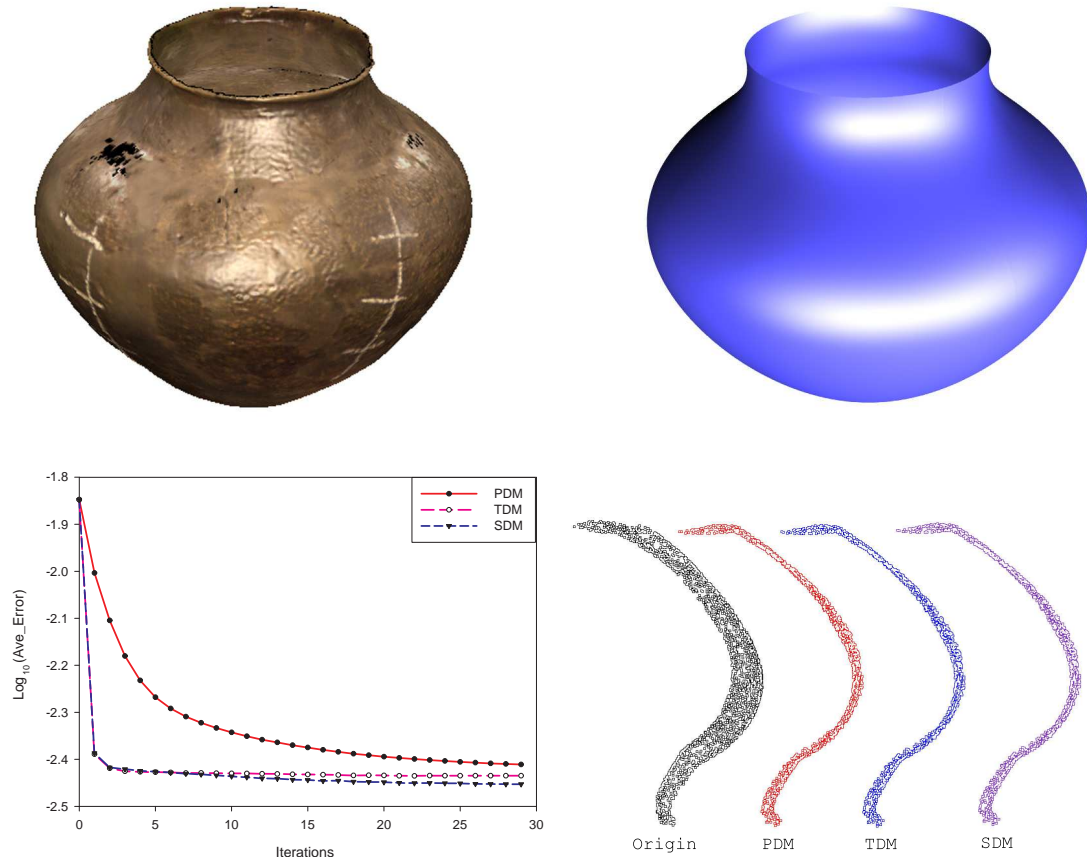


FIGURE 3.1: Approximation of an archeological finding by a rotational surface: (top left) original model; (top right) surface reconstructed with SDM method; (bottom left) average errors versus the number of iterations. Note that SDM and TDM are nearly the same, and both better than PDM; (bottom right) data points rotated into a profile plane. From left to right: initial guess and results after optimization with PDM, TDM and SDM. (# of points: 2,260)

Here p is the pitch of the helical surface; we have $p = 0$ for a surface of revolution. The essential parameters $\mathbf{p}_k = (r_k, z_k)$ and p appear linearly, and therefore the SDM method from Section 3.3 can be applied. For pure surface fitting, according to Section 3.2.3, the parameters (u_i, v_i) of the closest point $\mathbf{s}_{i,c} = \mathbf{s}_c(u_i, v_i)$ to \mathbf{x}_i are kept unchanged when we move to $\mathbf{s}_{i,+} = \mathbf{s}_+(u_i, v_i)$.

The registration part does not require the full motion group. We exclude translations parallel to the x_3 -axis by setting $\bar{\mathbf{c}} = (\bar{c}_1, \bar{c}_2, 0)$. Moreover, rotations about the x_3 -axis are excluded by setting $\mathbf{c} = (c_1, c_2, 0)$. This is done for both surfaces of revolution and helical surfaces. In the latter case, eventually, necessary translations in axis direction or rotations about the axis can be handled via a translation of the profile curve parallel to the axis.

Figure 3.1 shows an example of 2,260 data points of a scanned pot. The profile curve is a cubic B-spline curve with 7 control points and uniform knots. The significant

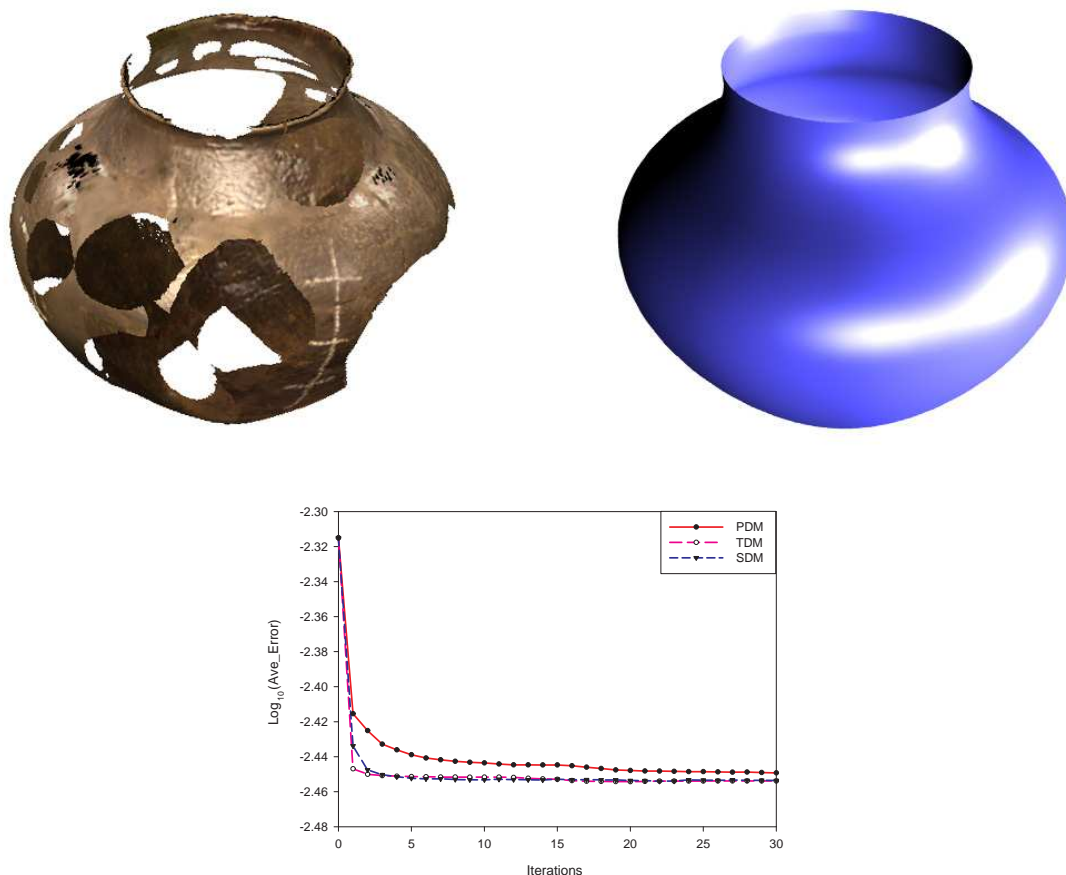


FIGURE 3.2: Approximation of a broken archeological finding by a rotational surface: (top left) original model; (top right) surface reconstructed with SDM method; (bottom) average errors versus the number of iterations. SDM and TDM are nearly the same, and both better than PDM. (# of points: 2,863)

improvement of the rotation axis is illustrated by the data points rotated into the profile plane: after optimization this cloud is much thinner than the one obtained in the initial fit via line geometry. The results of using SDM, TDM, and PDM are shown. Here and for the examples below, the stability of SDM is enhanced by applying a regularization similar to the L-M method, a trust-region based regularization conventionally applied to the Gauss-Newton method. It is observed that for this example SDM and TDM have nearly the same convergence behavior, while PDM is much slower (Figure. 3.2).

3.4.2 Spiral surfaces

A spiral surface is generated by a curve undergoing a spiral motion (one-parameter subgroup of similarities in \mathbb{R}^3), which is the composition of a rotation about a spiral axis A and an exponential scaling from the so-called spiral center $C \in A$. Placing C at the origin and setting A to be the x_3 -axis, a spiral surface with a B-spline profile can be

represented as

$$\mathbf{x}(u, v) = e^{pv} \sum_k [r_k \cos v, r_k \sin v, z_k] B_k(u). \quad (3.16)$$

Spiral surfaces are frequently taken as models for shapes of shells [94]. We would like to test the precision of this mathematical model using our new optimization method. Very recently, we devised a method which is in some sense analogous to the line geometric approach and can estimate the spiral axis and center from a set of data points and estimated normals, under the assumption that the data points are close to some spiral surfaces [63]. Using this method to provide an initial fit, we now present an improved spiral fitting algorithm based on SDM.

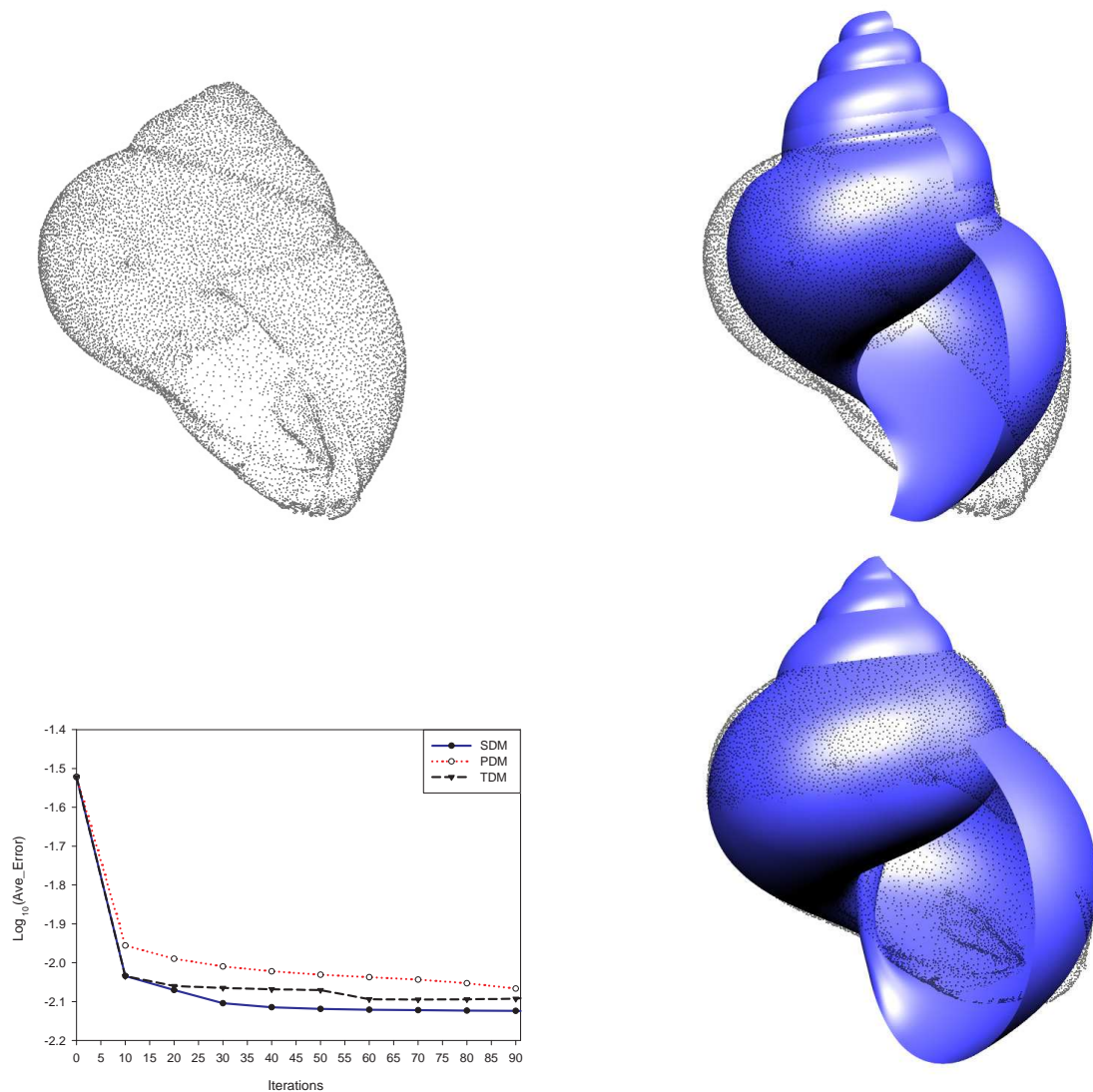


FIGURE 3.3: Shell *helix pomata*: (top left) data points of a shell; (top right) initial fit; (bottom left) average error v.s. the number of iterations. Note that SDM performs better than PDM and TDM; (bottom right) fit computed by SDM. (# of points: 13666)

We note that $\mathbf{x}(u, v)$ (3.16) is no longer linear in the unknowns p (spiral parameter) and $\mathbf{p}_k = (r_k, z_k)$. Hence, we use the following first order Taylor approximant at the current values p_c and $\mathbf{p}_{k,c}$ of the unknowns,

$$\begin{aligned} \mathbf{x}(u, v) &= e^{p_c v} \sum_k [r_k \cos v, r_k \sin v, z_k] B_k(u) + \\ &+ v(p - p_c) e^{p_c v} \sum_k [r_{k,c} \cos v, r_{k,c} \sin v, z_{k,c}] B_k(u). \end{aligned}$$

Keeping the parameter values (u_i, v_i) of the foot points $\mathbf{s}_{i,c}$, we arrive at new points $\mathbf{s}_{i,+}$, which depend linearly on the unknown parameters. Therefore, SDM from Section 3.3 works again.

Figure 3.3 shows that an initial fit can be improved significantly by our optimization algorithm, although for this example the residual fitting error is still rather large. For this example, we see that PDM and TDM do not converge as fast as SDM. Since the optimized fit is not ideal, we are inclined to conclude that the spiral surface is not an accurate model for this type of shell.

3.4.3 Constrained 3D shape reconstruction

3.4.3.1 Constrained fitting to a single set of data points

In the applications described above, it is an advantage to represent a fitting surface in an *adapted coordinate system* Σ ; for example, we put the rotational axis into a coordinate axis or the spiral center into the origin. Choosing an adapted coordinate system may also be possible for a typical engineering object: important elements such as rotational axes, planar faces, etc. can be brought into a special position with respect to Σ . Using an adapted coordinate system, we can set up a parametric model (see Figures 3.4, 3.8, 3.10). Varying the parameters gives a family of models all of which satisfy the constraints. Thus, our viewpoint leads to an unconstrained optimization problem for the parameters of the model.

Identifying such a coordinate system Σ for building the parametric model is made feasible either with some prior knowledge about the model or by user interaction. Within the general SDM procedure described in Section 3.3, we now adapt the model shape parameters (if necessary, using linearization as in Section 3.4.2) and update the position of the data set using a rigid motion with respect to the model shape.

We use an example to illustrate these steps. Figure 3.4 shows 33,981 measured data points of a machine part, and the parametric CAD model of the part and three side

views, with constraints and model parameters indicated. Figure 3.5 shows the initial fit, final fit and error curves. The constraints of the model are preserved strictly and only the values of the model parameters are updated in each iteration. Here and later for the examples in Section 3.4.3, only the TDM method is used, since many faces of the CAD model are planar, i.e., having zero curvature, thus making SDM reduce to TDM at these planar locations. The variations of all the model parameters are shown in Figure 3.6.

For evaluation, using the same data and CAD model in Figure 3.4, we compare our combined approach with an approach that optimizes position and surface shape alternatively [71]; (the latter is therefore called the *alternating method*). TDM is used in both position and shape updates in the alternating method. One position update and its successive surface shape update are counted as one iteration of the alternating method. The two error curves are shown in Figure 3.5 (bottom right), from which it is clear that the combined approach is more efficient. A theoretical explanation is that in the alternating method the iteration follows a zig-zag path in the subspace of motion parameters and the subspace of shape parameters, thus the resulting only in linear convergence in the alternating method. On the other hand, the combined method using TDM behaves similarly to the Gauss-Newton method, and therefore can have near quadratic convergence for small-residual problems.

3.4.3.2 Constrained fitting to multiple views

In this section we provide an experimental validation of our conjecture that relaxing the initial registration by allowing different motions for the individual scans can reduce the overall error. As seen from Figures 3.7,3.8 and 3.9,3.10, the combination of fitting with multiple-view registration leads to higher accuracy than first performing the registration on the point cloud data and then fitting a model to the registered data. We consider it an important feature of our algorithm that multiple-view registration can so easily be combined with fitting.

3.4.4 Remarks on the implementation

In this section we provide a few details on the actual implementation.

Closest points. Our algorithm requires to find, for every data point \mathbf{x} , its closest point (foot-point) \mathbf{s} on the initial surface. If there are too many points, this step will be very time consuming. To speed up this step, we sample a number of points on the surface firstly, then construct a *kd*-tree structure for finding the nearest point in the set of sample points. Viewing the nearest sample point as an initial point, we then apply a

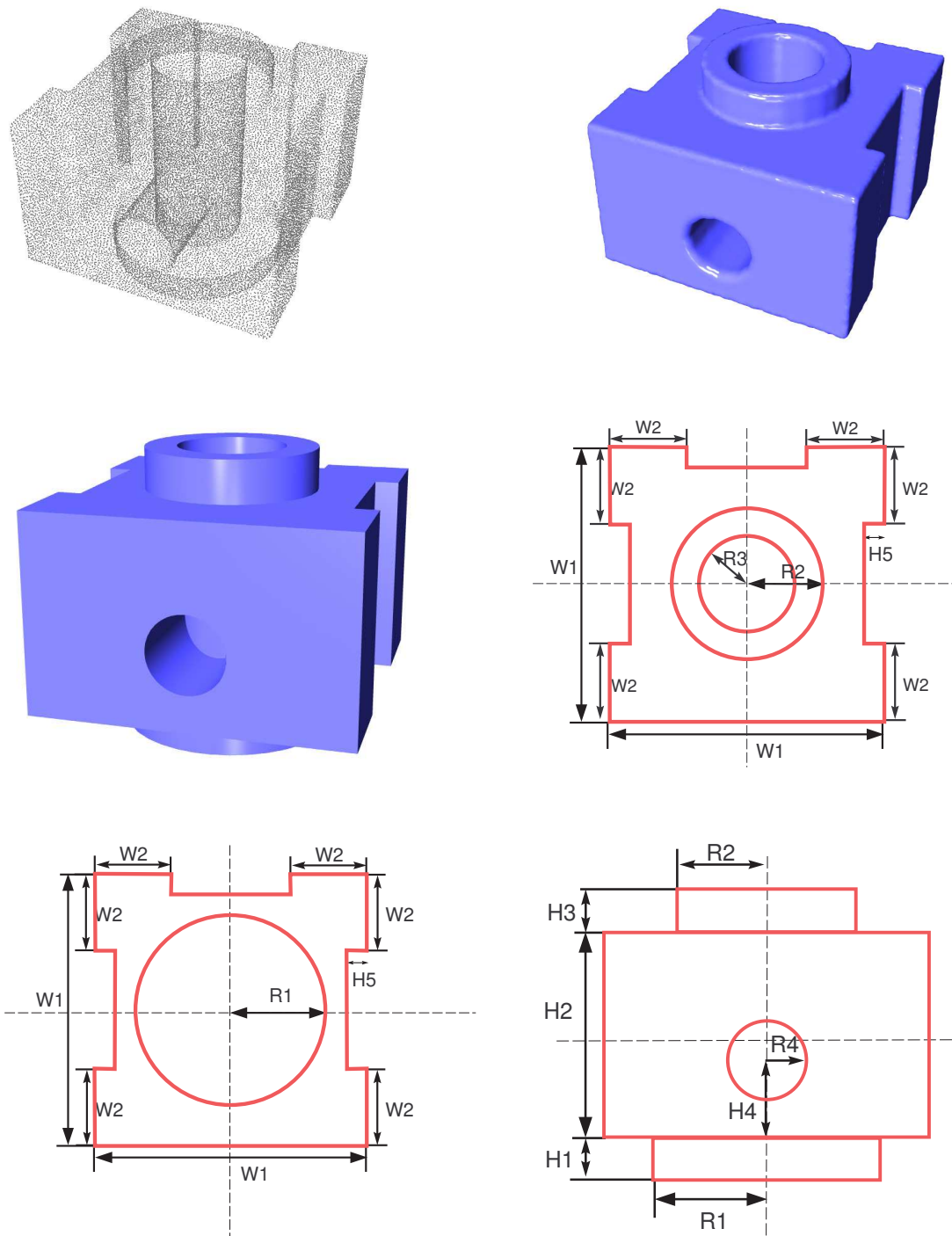


FIGURE 3.4: (top left): A data set of 33,981 points from a machine part; (top right): its triangulated surface; (middle left): a parametric model; (middle right): top view; (bottom left): bottom view; (bottom right): front view.

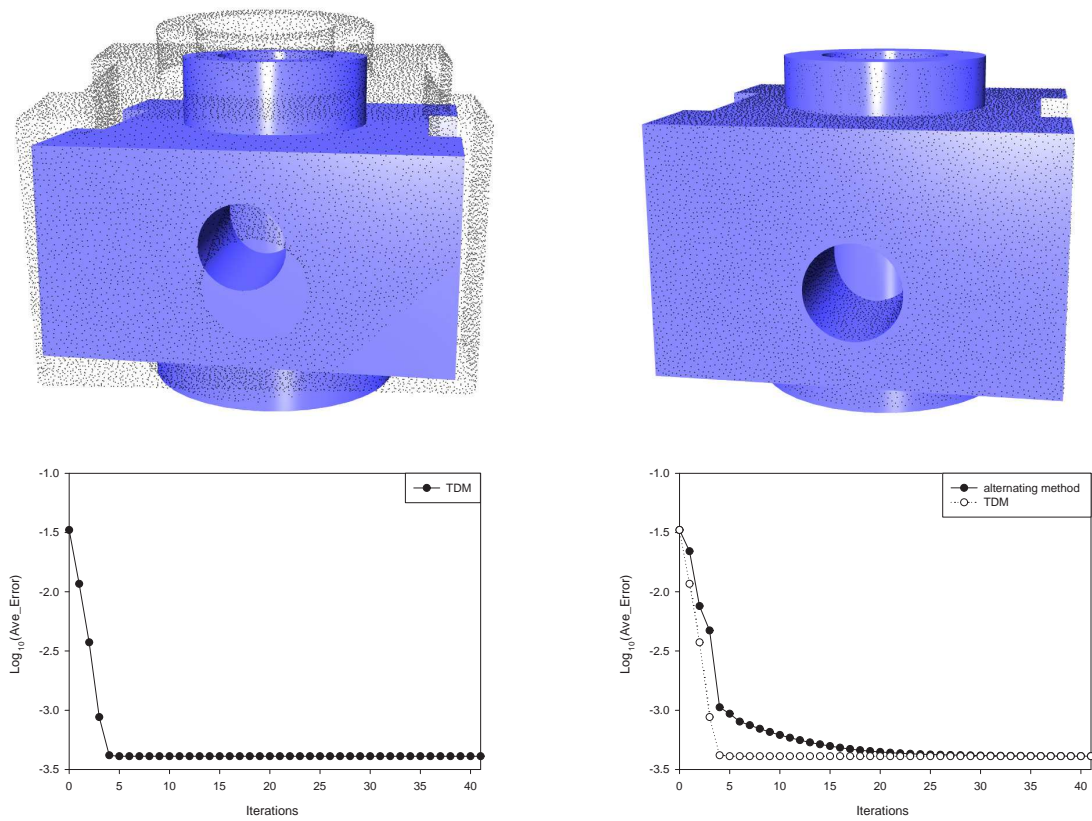


FIGURE 3.5: (top left): Initial fit; (top right): optimized fit after 6 iterations of TDM; (bottom left): average error of our method v.s. the number of iterations; (bottom right): comparison with the alternating method.

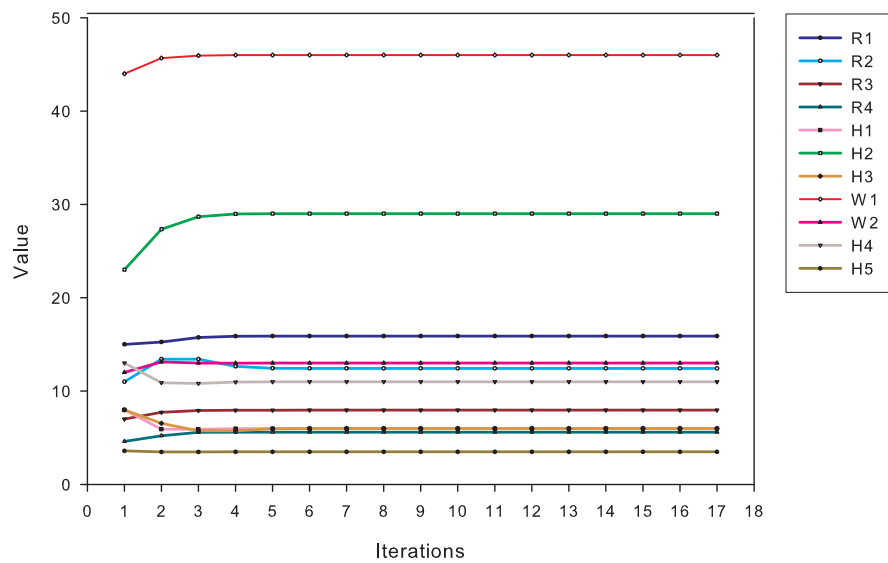


FIGURE 3.6: Variations of the parameters for the machine part model in Figure 3.4.

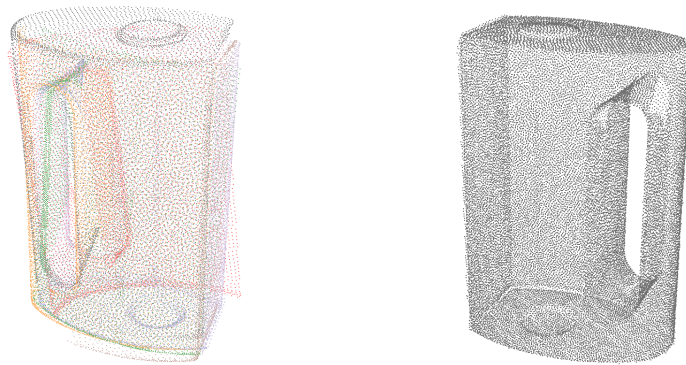


FIGURE 3.7: left: A data set of 331,150 points from a 3D model using 7 scans; right: the registered point set

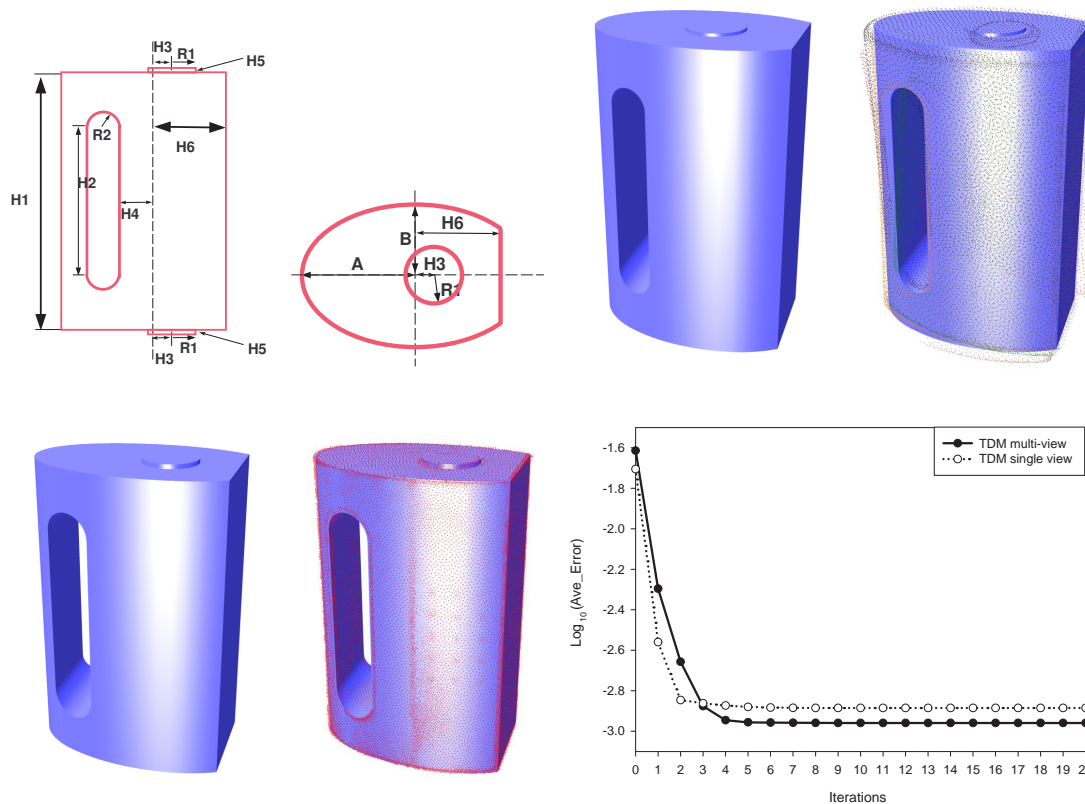


FIGURE 3.8: (top left): front and bottom views of a parametric model; (top right): initial model and initial fit; (bottom left): optimized fit after 8 iterations of TDM; (bottom right): average error v.s. the number of iterations, and comparison with the single view case. We see that combining fitting and multiple-view registration lead to higher accuracy.

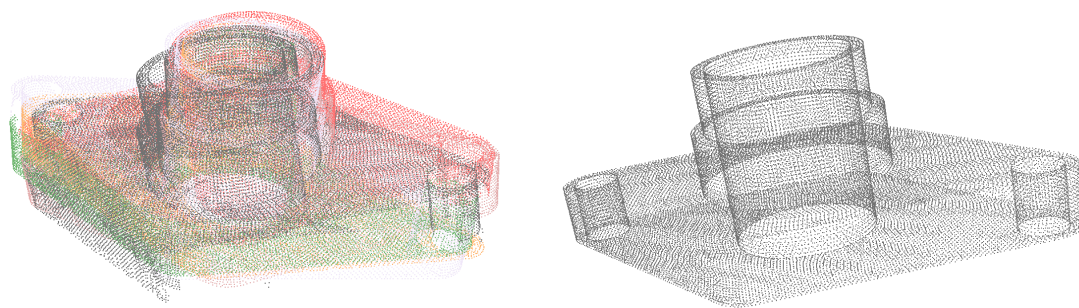


FIGURE 3.9: left: A data set of 107670 points from a CAD model based on 7 scans; right: the registered point set

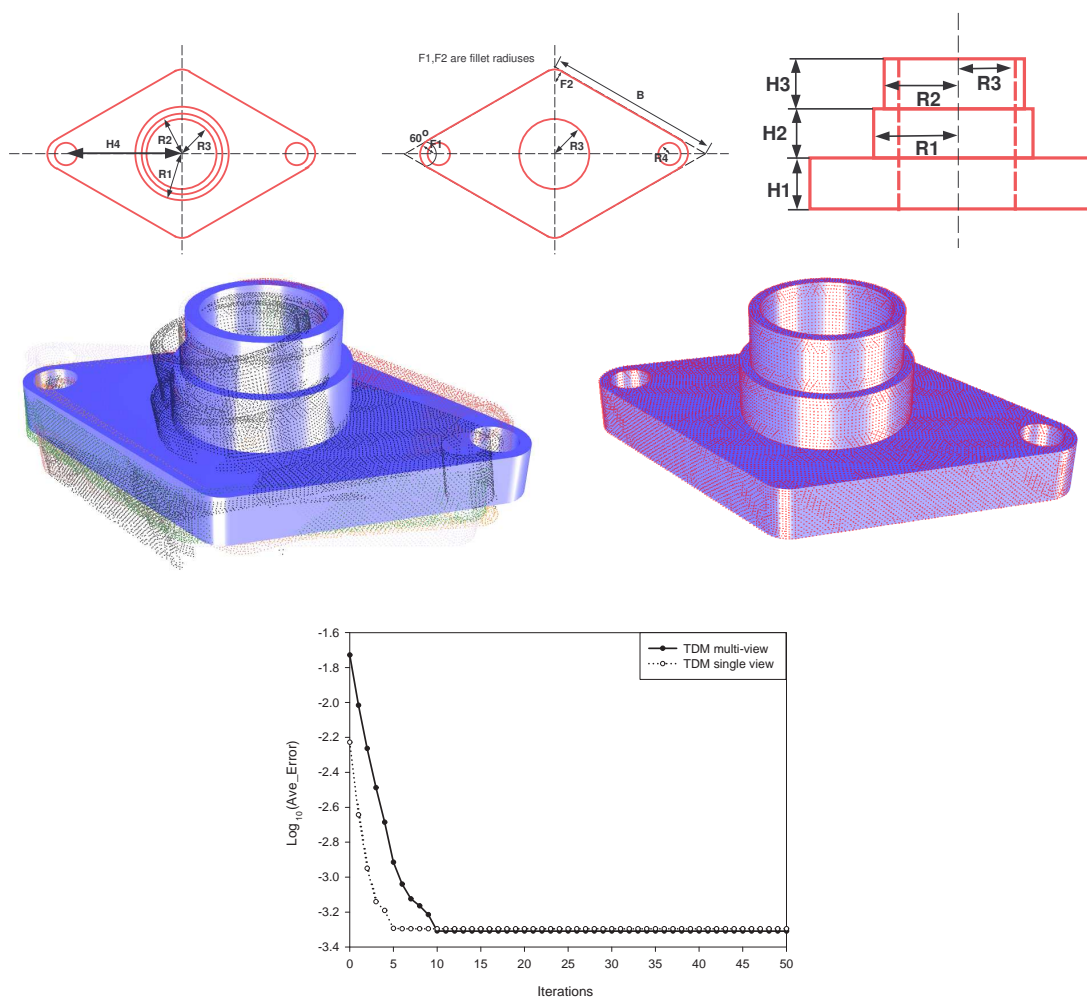


FIGURE 3.10: Combining fitting and multiple-view registration. (top left): front view of the parametric model; (top right): side view; (middle left): initial fit; (middle right): optimized fit after 12 iterations of multiple-views TDM; (bottom) average error v.s. the number of iterations, and comparison with the single view case.

Newton iteration to compute a more precise foot-point on the surface. Of course, if the surface is very simple, like a plane or cylinder, we can find the closest point directly.

Step size control. In each step we minimize the objective function by solving a linear system. We use Levenberg-Marquart regularization in order to avoid instabilities and too large steps [74].

3.5 Conclusions

We have shown that 3D shape fitting in the presence of constraints may be simplified and made more efficient by combining registration and surface approximation. To achieve a good convergence behavior, we have implemented this idea within the framework of SDM. We have also compared SDM with TDM and PDM, and found that SDM and TDM are more efficient than PDM, and with proper step size control TDM is as good as SDM in many cases. Moreover, we have shown that relaxing the initial registration in the final phase of our algorithm is easily formulated within our framework and improves the fitting accuracy.

Claim: the presented material in this chapter has been published in [84].

Chapter 4

Least Squares Orthogonal Distance Fitting of Parametric Curves and Surfaces

4.1 Introduction

Effective and accurate curve/surface fitting plays an important role and serves as a basic module in CAGD, computer graphics, computer vision and other scientific and engineering fields. We extend the fitting and registration problem discussed in Chapter 2,3 in a more general setting, : fit a parametric curve/surface $\mathbf{C}(\mathbf{P}; \mathbf{t})$ (whose parametric form is known but the parameter values are to be determined) to a set of given data points $\{\mathbf{X}_j\}_{j=1}^n \subset \mathbb{R}^s$. Here \mathbf{P} are the shape parameters and $\mathbf{t} = (\tau_1, \dots, \tau_m)$ are location parameters (For instance, \mathbf{t} of a 3D parametric surface $\mathbf{C}(u, v)$ is (u, v)). This problem is usually stated as a standard nonlinear least squares problem:

$$\min_{\mathbf{P}, \mathbf{t}_1, \dots, \mathbf{t}_n} \sum_{j=1}^n \|\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j\|^2 \quad (4.1)$$

Where \mathbf{t}_j is associated with the data point \mathbf{X}_j .

There exists vast literature about this problem in mathematics, statistics and computer science. Despite the differences of existing methods in variant contexts, the basics of most methods are classical optimization theory and optimization techniques such as decent methods and Gauss-Newton methods [99] appearing in different forms.

First we introduce the traditional way of fitting a parametric curve/surface to a given data set in CAGD [67, 68, 154]. The first step is the parameterization which associates

the location parameter \mathbf{t}_j to each data point \mathbf{X}_j . After substituting \mathbf{t}_j into (4.1), the second step is solving a linear least squares problem if the shape parameters occur in linear form; for instance, \mathbf{P} are the control points of the B-spline curve/surface. By executing these two steps iteratively, improved location parameters and shape parameters are obtained. This approach has been widely used because of its simplicity. However its convergence rate is slow and is proven to be linear [14]. On the other hand, without separating \mathbf{P} and $\mathbf{t}_1, \dots, \mathbf{t}_n$, the general optimization techniques of course can be applied. One can optimize $\mathbf{P}, \mathbf{t}_1, \dots, \mathbf{t}_n$ simultaneously [61, 138]. Moreover if \mathbf{P} are in the linear form, the separable nonlinear least squares method (variable projection) can be employed and is better than the simultaneous method [14, 24, 57]. But the size of corresponding nonlinear least squares problem becomes larger when n increases. Therefore these methods are not suitable for fitting a large number of data points. In the metrology and pattern recognition communities people prefer the least squares orthogonal distance technique which is an iterative method and considers the relationship between shape parameters and location parameters. In Chapter 2 a curvature-based squared distance minimization (SDM) is proposed for orthogonal distance fitting for B-spline curve fitting. In this chapter we consider general parametric curve/surface fitting problems, which are not limited in 2D, 3D curves and surfaces.

Contributions: Inspired by the approaches in [8] and our work in Chapter 2, 3, we analyze the existing orthogonal distance techniques by rephrasing them into a general optimization framework. We propose two modified methods CDM and GTDM based on geometric and optimizational analysis. We reveal that the existing and our proposed methods have clear geometric meanings. This better understanding will benefit the general parametric models fitting and registration.

4.2 Preliminary

4.2.1 Notation

Let $\mathbf{C}(\mathbf{P}; \mathbf{t}) \subset \mathbb{R}^s$ represent a family of parametric curves or surfaces. A set of points $\{\mathbf{X}_j\}_{j=1}^n \subset \mathbb{R}^s$ are to be approximated by $\mathbf{C}(\mathbf{P}; \mathbf{t})$. Here $\mathbf{t} = (\tau_1, \dots, \tau_m) \in \mathbb{R}^m$ is the location parameter and $\mathbf{P} = (p_1, \dots, p_r)$ is the shape parameter. For instance, if $m = 1$, $\mathbf{C}(\mathbf{P}; \mathbf{t})$ represents a parametric curve. We assume that $\mathbf{C}(\mathbf{P}; \mathbf{t})$ has C^2 continuity. In this chapter vectors and matrices are denoted by bold face and vectors are in the column format. The first-order partial derivatives of $\mathbf{C}(\mathbf{P}; \mathbf{t})$ are denoted as follows:

$$\frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t})}{\partial \mathbf{P}} = \left[\frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t})}{\partial p_1}, \dots, \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t})}{\partial p_r} \right], \quad \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t})}{\partial \mathbf{t}} = \left[\frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t})}{\partial \tau_1}, \dots, \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t})}{\partial \tau_m} \right]$$

$$\nabla_{\mathbf{P}} \mathbf{t} = \begin{bmatrix} \frac{\partial \tau_1}{\partial p_1} & \cdots & \frac{\partial \tau_m}{\partial p_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \tau_1}{\partial p_r} & \cdots & \frac{\partial \tau_m}{\partial p_r} \end{bmatrix}.$$

In many curves and surfaces fitting applications, the initial positions of data points and the model are not aligned well. The data points or the model is allowed to be transformed in the fitting process. By introducing proper transformation, the fitting process can be accelerated and overcome some local minimum cases. The most common transformation is rigid transformation[1, 8]. Combined with rigid transformation, we have shown in Chapter 3 that the convergence speed of the fitting algorithm can be faster and high accuracy also can be achieved. Although the transformation can be applied to the data points or the model, for unifying our analysis we assume the transformation is applied on the parametric model, i.e. the shape parameter \mathbf{P} can contain the transformation parameters if needed.

4.2.2 Nonlinear least squares

We consider a standard nonlinear least squares problem which minimizes the objective function $\mathbf{f}(\mathbf{X})$:

$$\min_{\mathbf{X}} \frac{1}{2} \sum_{i=1}^n f_i^2(\mathbf{X}) \triangleq \mathbf{f}(\mathbf{X}) \quad (4.2)$$

The residual vector is defined as $\mathbf{r}(\mathbf{X}) = (f_1(\mathbf{X}), f_2(\mathbf{X}), \dots, f_n(\mathbf{X}))^T$. The first derivative of $\mathbf{f}(\mathbf{X})$ can be expressed in terms of the Jacobian of \mathbf{r} : $\mathbf{J}(\mathbf{X}) = \begin{pmatrix} \nabla f_1(\mathbf{X}) \\ \vdots \\ \nabla f_n(\mathbf{X}) \end{pmatrix}$, where $\nabla f_i(\mathbf{X})$ is the gradient of f_i with respect to \mathbf{X} . The gradient and Hessian of $\mathbf{f}(\mathbf{X})$ have the following forms

$$\nabla \mathbf{f}(\mathbf{X}) = \mathbf{J}(\mathbf{X})^T \mathbf{r}(\mathbf{X}); \quad \mathbf{H} = \nabla^2 \mathbf{f}(\mathbf{X}) = \mathbf{J}(\mathbf{X})^T \mathbf{J}(\mathbf{X}) + \sum_{i=1}^n f_i(\mathbf{X}) \nabla^2 f_i(\mathbf{X})$$

The Gauss-Newton method approximates the Hessian by $\mathbf{J}(\mathbf{X})^T \mathbf{J}(\mathbf{X})$. In practice the line search strategy or the Levenberg-Marquardt method

$$(\mathbf{J}(\mathbf{X})^T \mathbf{J}(\mathbf{X}) + \lambda \mathbf{I}) \delta \mathbf{X} = -\mathbf{J}(\mathbf{X})^T \mathbf{r}(\mathbf{X})$$

is incorporated with the Gauss-Newton method. The Quasi-Newton type method approximates the Hessian or the inverse of the Hessian by a positive-definite matrix which

is updated at each iteration with some specified schemes such as **BFGS** [99]. But in this chapter we mainly focus on Gauss-Newton type methods.

4.2.3 Principal directions and curvatures of parametric curves and surfaces

For a smooth parametric curve/surface $\mathbf{C}(\mathbf{t})$, its first-order derivatives $\partial_{\tau_1}\mathbf{C}(\mathbf{t}_p), \dots, \partial_{\tau_m}\mathbf{C}(\mathbf{t}_p)$ at point $\mathbf{C}(\mathbf{t}_p)$ span a tangential space $\top_p\mathbf{C}$. Its orthogonal complement defines the normal space $\perp_p\mathbf{C}$. For a given unit normal vector $\mathbf{n}_p \in \perp_p\mathbf{C}$, we can define the principal vectors and curvatures with respect to \mathbf{n}_p . The details can be found in Section 2.2 of [148]. Let $\mathbf{T}_1, \dots, \mathbf{T}_m$ be the principle vectors which span $\top_p\mathbf{C}$ and $\kappa_1, \dots, \kappa_m$ be the corresponding principle curvatures with respect to \mathbf{n}_p . The orthonormal basis of $\perp_p\mathbf{C}$ are $\mathbf{N}_{m+1}, \dots, \mathbf{N}_s$. One identity about the orthonormal basis will be useful in this chapter:

$$\mathbf{I}_s = \mathbf{T}_1\mathbf{T}_1^T + \dots + \mathbf{T}_m\mathbf{T}_m^T + \mathbf{N}_{m+1}\mathbf{N}_{m+1}^T + \dots + \mathbf{N}_s\mathbf{N}_s^T. \quad (4.3)$$

Where \mathbf{I}_s is a $s \times s$ identity matrix.

Remark 4.1. For a 3D parametric curve, the curvature K and curvature direction \mathbf{N}^0 are well defined from differential geometry. Since in our discussion \mathbf{N} is not necessarily coincident with \mathbf{N}^0 , we have $\kappa = K \cdot \langle \mathbf{N}, \mathbf{N}^0 \rangle$. $\langle \star, \star \rangle$ is the inner product of two vectors.

4.3 Orthogonal Distance Fitting

The optimization process of orthogonal distance fitting contains two steps which are executed repeatedly:

1. Reparameterization: compute the foot-point of \mathbf{X}_j on $\mathbf{C}(\mathbf{P}; \mathbf{t})$, i.e, minimize the distance from \mathbf{X}_j to \mathbf{C} :

$$\min_{\mathbf{t}_j} \|\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j\|, j = 1, \dots, n \quad (4.4)$$

2. minimize one of the following objective functions by applying one step of optimization techniques such as Gauss-Newton methods:

$$\min_{\mathbf{P}} \left\| \left(\|\mathbf{C}(\mathbf{P}; \mathbf{t}_1(\mathbf{P})) - \mathbf{X}_1\|, \dots, \|\mathbf{C}(\mathbf{P}; \mathbf{t}_n(\mathbf{P})) - \mathbf{X}_n\| \right)^T \right\| \quad (4.5)$$

or

$$\min_{\mathbf{P}} \left\| \left(\mathbf{C}(\mathbf{P}; \mathbf{t}_1(\mathbf{P}))^T - \mathbf{X}_1^T, \dots, \mathbf{C}(\mathbf{P}; \mathbf{t}_n(\mathbf{P}))^T - \mathbf{X}_n^T \right)^T \right\| \quad (4.6)$$

Since (4.5) minimizes the l_2 norm of the residual vector \mathbf{r}_d :

$$\mathbf{r}_d = (\|\mathbf{C}(\mathbf{P}; \mathbf{t}_1(\mathbf{P})) - \mathbf{X}_1\|, \dots, \|\mathbf{C}(\mathbf{P}; \mathbf{t}_n(\mathbf{P})) - \mathbf{X}_n\|)^T, \quad (4.7)$$

the corresponding method is called *Distance-based* method; also since (4.6) minimizes the l_2 norm of the residual vector \mathbf{r}_c :

$$\mathbf{r}_c = \left(\mathbf{C}(\mathbf{P}; \mathbf{t}_1(\mathbf{P}))^T - \mathbf{X}_1^T, \dots, \mathbf{C}(\mathbf{P}; \mathbf{t}_n(\mathbf{P}))^T - \mathbf{X}_n^T \right)^T, \quad (4.8)$$

the corresponding method is called *Coordinate-based* method. By applying nonlinear least squares optimization technique these two methods produce different results. Atieg and Watson present their analysis on *Distance-based* and *Coordinate-based* Gauss-Newton approaches in [8]. We will show the geometry behind these two methods and their variations.

Orthogonality: Because \mathbf{t}_j is the minimizer of (4.4), the orthogonality condition (4.9) below always holds in each step, except when the foot-point is at the boundary of \mathbf{C} .

$$\left\langle \mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j, \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_k} \right\rangle = 0, \quad j = 1, \dots, n; k = 1, \dots, m \quad (4.9)$$

The orthogonality condition (4.9) plays an important role in parameterization correction and optimization. Many effective foot-point computation methods are available in literature [68, 106, 127]. If the explicit foot-point formula is not available, one can apply Newton-like optimization methods on (4.9) to obtain the foot-point and corresponding location parameter. But the initial guess \mathbf{t}^0 is a key issue in foot-point computation. For complex parametric curves/surfaces, one good strategy is to build a k-D tree from the sample points $\{\mathbf{C}(\mathbf{P}; \mathbf{t}_k), k = 1, \dots, L\}$ then find the nearest point for \mathbf{X}_j which serves as the initial foot-point.

4.3.1 Distance-based Gauss-Newton method

Distance-based methods are widely used in metrology. Here the l_2 norm of residual vector \mathbf{r}_d is to be minimized. Depending on whether considering the association between the shape parameter \mathbf{P} and the local parameter \mathbf{t} , Gauss-Newton distance-based methods can be categorized to two types: the separated method and the standard method.

(1) Separated distance-based Gauss-Newton method

The residual vector \mathbf{r}_d in the separated distance-based Gauss-Newton method is defined as

$$\mathbf{r}_d = (\|\mathbf{C}(\mathbf{P}; \mathbf{t}_1) - \mathbf{X}_1\|, \dots, \|\mathbf{C}(\mathbf{P}; \mathbf{t}_n) - \mathbf{X}_n\|)^T,$$

where each \mathbf{t}_j is fixed. The first-order total derivative of $\|\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j\|$ with respect to \mathbf{P} is

$$\nabla_{\mathbf{P}} \|\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j\| = \frac{\mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T - \mathbf{X}_j^T}{\|\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j\|} \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}}, \quad (4.10)$$

where it must be assumed that $\mathbf{C}(\mathbf{P}; \mathbf{t}_j) \neq \mathbf{X}_j$ such that the derivative exists. Numerical computation can be unstable when $\mathbf{C}(\mathbf{P}; \mathbf{t}_j)$ approaches \mathbf{X}_j . Notice that if \mathbf{C} is a 2D parametric curve or a 3D parametric surface, the vector $\frac{\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j}{\|\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j\|} := \mathbf{N}_j$ actually is the unit normal at $\mathbf{C}(\mathbf{t}_j)$ whose sign may be positive or negative. Thus the instability can be eliminated if we replace it with the unit normal. The Jacobian of \mathbf{r}_d at $\mathbf{C}(\mathbf{P}; \mathbf{t}_j)$ can be written as

$$\mathbf{J}_1 = \begin{pmatrix} \mathbf{N}_1^T \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_1)}{\partial \mathbf{P}} \\ \vdots \\ \mathbf{N}_n^T \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_n)}{\partial \mathbf{P}} \end{pmatrix}.$$

From the normal equation $\mathbf{J}_1^T \mathbf{J}_1 \cdot \delta \mathbf{P} = -\mathbf{J}_1^T \mathbf{r}_d$, we can derive that

$$\sum_{j=1}^n \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T}{\partial \mathbf{P}} \mathbf{N}_j \mathbf{N}_j^T \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} \cdot \delta \mathbf{P} = - \sum_{j=1}^n \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T}{\partial \mathbf{P}} (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j), \quad (4.11)$$

where $\delta \mathbf{P}$ is the increment of the shape parameter \mathbf{P} .

Now we show the geometric meaning behind (4.11). the right hand side of (4.11) can be rewritten as:

$$\frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T}{\partial \mathbf{P}} (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j) = \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T}{\partial \mathbf{P}} \mathbf{N}_j \cdot \mathbf{N}_j^T (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j) \quad (4.12)$$

Now Equation.(4.11) actually minimizes the squared distance from data points to their tangent planes at the foot-points:

$$\min_{\mathbf{P}} \sum_{j=1}^n [\mathbf{N}_j^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2 \quad (4.13)$$

It is easy to verify the normal equation of Equation.(4.13) is Equation.(4.11) just by applying the Gauss-Newton method on Equation.(4.13). We call this kind of geometric minimization TDM (**T**angent **D**istance **M**inimization)(cf. Chapter 2).

As we have pointed out, there is no numerical problem for 2D parametric curves and 3D parametric surfaces if we replace \mathbf{N}_j with curves/surfaces' normals. For high dimension parametric curves/surfaces ($m < s - 1$), TDM is not suitable when the data points are almost contained in a low dimension space $\mathbb{R}^l, l < s$. For instance, fitting a 3D parametric curve to a set of points in a plane causes the ill-conditioning of Jacobian matrix [8]. We use a simple example to illustrate this problem. Assume that a 3D curve has the following parametric form (at^2, bt^3, c) , where a, b, c are shape parameters and the data points lie in the x - y plane. The third component of \mathbf{N}_j will be always zero. It means that c does not appear in $\mathbf{N}_j^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)$. Therefore the normal equations will be singular.

(2) Standard distance-based Gauss-Newton method

With the consideration of the association between \mathbf{t} and \mathbf{P} , in the standard distance-based Gauss-Newton method the residual vector \mathbf{r}_d is defined as in (4.7). The first-order total derivative of each element of \mathbf{r}_d with respect to \mathbf{P} is

$$\begin{aligned} \nabla_{\mathbf{P}} \|\mathbf{C}(\mathbf{P}; \mathbf{t}_j(\mathbf{P})) - \mathbf{X}_j\| &= \frac{\mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T - \mathbf{X}_j^T}{\|\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j\|} \nabla_{\mathbf{P}} \mathbf{C}(\mathbf{P}; \mathbf{t}_j(\mathbf{P})) \\ &= \frac{\mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T - \mathbf{X}_j^T}{\|\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j\|} \left[\frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} + \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{t}} \nabla_{\mathbf{P}} \mathbf{t}_j \right] \\ &= \mathbf{N}_j^T \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}}, \end{aligned} \quad (4.14)$$

where the term $(\mathbf{C}(\mathbf{P}; \mathbf{t}_j(\mathbf{P}))^T - \mathbf{X}_j^T) \cdot \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{t}} \nabla_{\mathbf{P}} \mathbf{t}_j$ is eliminated due to the orthogonality condition. The result (4.14) is the same as (4.10), which means that both separated and standard distance-based approaches produce the same geometric minimization scheme – TDM.

4.3.2 Coordinate-based Gauss-Newton method

Now we consider the coordinate-based Gauss-Newton method based on the objective function (4.6), which is widely used in pattern recognition community.

(1) Separated coordinate-based Gauss-Newton method

In the separated coordinate-based Gauss-Newton method the residual vector \mathbf{r}_c is defined as

$$\mathbf{r}_c = \left(\mathbf{C}(\mathbf{P}; \mathbf{t}_1)^T - \mathbf{X}_1^T, \dots, \mathbf{C}(\mathbf{P}; \mathbf{t}_n)^T - \mathbf{X}_n^T \right)^T.$$

The first-order total derivative of $\mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T - \mathbf{X}_j^T$ with respect to \mathbf{P} is $\frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T}{\partial \mathbf{P}}$. So the Jacobian \mathbf{J}_2 of \mathbf{r}_c is

$$\begin{pmatrix} \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_1)}{\partial \mathbf{P}} \\ \vdots \\ \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_n)}{\partial \mathbf{P}} \end{pmatrix}.$$

Still from the normal equation $\mathbf{J}_2^T \mathbf{J}_2 \cdot \delta \mathbf{P} = -\mathbf{J}_2^T \mathbf{r}_c$, we obtain

$$\sum_{j=1}^n \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T}{\partial \mathbf{P}} \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} \cdot \delta \mathbf{P} = -\sum_{j=1}^n \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T}{\partial \mathbf{P}} (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j) \quad (4.15)$$

The normal equation actually represents a geometric minimization

$$\min_{\mathbf{P}} \sum_{j=1}^n [\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j]^2 \quad (4.16)$$

which penalizes the squared distance from data points to foot points, we call this method PDM (**P**oint **D**istance **M**inimization). It is widely used in CAGD community because of its simplicity. Especially when \mathbf{P} is in the linear form in \mathbf{C} , one just needs to solve a linear equation and the $\|\mathbf{r}_c\|$ always decreases. However PDM only exhibits linear convergence.

(2) Standard coordinate-based Gauss-Newton method

In the standard distance-based Gauss-Newton method the residual vector \mathbf{r}_c is (4.8), where \mathbf{t}_j is associated with \mathbf{P} through (4.9). The first-order total derivative of each element with respect to \mathbf{P} is

$$\nabla_{\mathbf{P}} (\mathbf{C}(\mathbf{P}; \mathbf{t}_j(\mathbf{P})) - \mathbf{X}_j) = \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} + \sum_{k=1}^m \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_{j,k}} \nabla_{\mathbf{P}} \tau_{j,k}(\mathbf{P}) \quad (4.17)$$

In general the explicit expression of $\tau_{j,k}(\mathbf{P})$ with respect to \mathbf{P} is not always available. So we use the implicit procedure presented in [8]. Since the orthogonality condition (4.9) holds and it is an identity in \mathbf{P} , its total derivative with respect to \mathbf{P} is still $\mathbf{0}$. Therefore

we have

$$\begin{aligned} \mathbf{0} &= \nabla_{\mathbf{P}} \left\langle \mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j, \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_{j,k}} \right\rangle \\ &= \left\langle \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} + \sum_{l=1}^m \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_{j,l}} \nabla_{\mathbf{P}} \tau_{j,l}(\mathbf{P}), \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_{j,k}} \right\rangle + \\ &\quad \left\langle \mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j, \frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_{j,k} \partial \mathbf{P}} + \sum_{l=1}^m \frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_{j,k} \partial \tau_{j,l}} \nabla_{\mathbf{P}} \tau_{j,l}(\mathbf{P}) \right\rangle. \end{aligned}$$

Without loss of generality, suppose $\mathbf{C}(\mathbf{P}; \mathbf{t}_j)$ is a local regular parameterization such that $\tau_{j,1}, \dots, \tau_{j,m}$ -direction vectors are unit principle direction vectors $\mathbf{T}_{j,1}, \dots, \mathbf{T}_{j,m}$ with respect to \mathbf{N}_j (see Section 4.2.3). The above equation can be simplified as

$$\begin{aligned} \mathbf{0} &= \left\langle \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} + \sum_{l=1}^m \mathbf{T}_{j,l} \nabla_{\mathbf{P}} \tau_{j,l}(\mathbf{P}), \mathbf{T}_{j,k} \right\rangle + \\ &\quad \left\langle \mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j, \frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_{j,k} \partial \mathbf{P}} + \frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_{j,k}^2} \nabla_{\mathbf{P}} \tau_{j,k}(\mathbf{P}) \right\rangle \\ &= \mathbf{T}_{j,k}^T \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} + \sum_{l=1}^m \mathbf{T}_{j,k}^T \mathbf{T}_{j,l} \nabla_{\mathbf{P}} \tau_{j,l}(\mathbf{P}) + (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)^T \frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_{j,k} \partial \mathbf{P}} + \\ &\quad d_j \mathbf{N}_j^T \kappa_{j,k} \mathbf{N}_j \nabla_{\mathbf{P}} \tau_{j,k}(\mathbf{P}) \\ &= \mathbf{T}_{j,k}^T \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} + (1 + d_j \kappa_{j,k}) \nabla_{\mathbf{P}} \tau_{j,k}(\mathbf{P}) + (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)^T \frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_{j,k} \partial \mathbf{P}}, \end{aligned}$$

where $d_j = \|\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j\|$, $\kappa_{j,k}$ is the principle curvature along $\mathbf{T}_{j,k}$ with respect to \mathbf{N}_j . Then we obtain

$$\nabla_{\mathbf{P}} \tau_{j,k} = - \frac{\mathbf{T}_{j,k}^T \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} + (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)^T \frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \tau_{j,k} \partial \mathbf{P}}}{1 + d_j \kappa_{j,k}} \quad (4.18)$$

We can rewrite (4.17) as

$$\nabla_{\mathbf{P}} (\mathbf{C}(\mathbf{P}; \mathbf{t}_j(\mathbf{P})) - \mathbf{X}_j) = \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} - \sum_{k=1}^m \frac{\mathbf{T}_{j,k} \mathbf{T}_{j,k}^T \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} + d_j \mathbf{T}_{j,k} \mathbf{N}_j^T \frac{\partial \mathbf{T}_{j,k}}{\partial \mathbf{P}}}{1 + d_j \kappa_{j,k}} \quad (4.19)$$

In the degenerate case when $1 + d_j \kappa_{j,k} \approx 0$, one can modify the denominator to $1 + d_j |\kappa_{j,k}|$ to improve the condition number of the normal equation. We note that this degenerate case is not addressed in the literature of orthogonal distance fitting, such as [1, 8]. But it can happen in practice. For example, let a parametric circle be $(r \cos t, r \sin t)$ and one data point \mathbf{X}_j be near to the origin. We have $1 + d_j \kappa_{j,k} \approx 1 + r \cdot \frac{-1}{r} = 0$.

(3) Modified standard coordinate-based Gauss-Newton methods

The computational cost of the second-order derivatives $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{t}^2}$ and $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P} \partial \mathbf{t}}$ may be high in some applications. So we shall derive two kinds of modified standard Gauss-Newton methods with less computation cost and clear geometric meanings.

First we recall the notation in Section 4.2.3. At the point $\mathbf{C}(\mathbf{t}_j)$, $\mathbf{T}_{j,1}, \dots, \mathbf{T}_{j,m}$ span a tangent vector space $\top_{j,p} \mathbf{C}$ and $\mathbf{N}_{j,m+1}, \dots, \mathbf{N}_{j,s}$ denote the orthonormal basis of $\top_{j,p} \mathbf{C}$'s orthogonal complement space $\perp_{j,p} \mathbf{C}$. The following identity always holds

$$\mathbf{I} = \mathbf{T}_{j,1} \mathbf{T}_{j,1}^T + \dots + \mathbf{T}_{j,m} \mathbf{T}_{j,m}^T + \mathbf{N}_{j,m+1} \mathbf{N}_{j,m+1}^T + \dots + \mathbf{N}_{j,s} \mathbf{N}_{j,s}^T. \quad (4.20)$$

We will use this identity in our following derivation. By dropping the second-order derivatives from Equation. 4.19, we will derive two methods.

1. Drop $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P} \partial \mathbf{t}}$. This leads to

$$\begin{aligned} \nabla_{\mathbf{P}} (\mathbf{C}(\mathbf{P}; \mathbf{t}_j(\mathbf{P})) - \mathbf{X}_j) &\approx \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} - \sum_{k=1}^m \frac{\mathbf{T}_{j,k} \mathbf{T}_{j,k}^T \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}}}{1 + d_j \kappa_{j,k}} \\ &= \mathbf{I} \cdot \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} - \sum_{k=1}^m \frac{\mathbf{T}_{j,k} \mathbf{T}_{j,k}^T \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}}}{1 + d_j \kappa_{j,k}} \\ &= \left(\sum_{k=1}^m \frac{d_j \kappa_{j,k} \mathbf{T}_{j,k} \mathbf{T}_{j,k}^T}{1 + d_j \kappa_{j,k}} + \sum_{k=m+1}^s \mathbf{N}_{j,k} \mathbf{N}_{j,k}^T \right) \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} \end{aligned}$$

Substituting the above result into the normal equation $\mathbf{J}^T \mathbf{J} \cdot \delta \mathbf{P} = -\mathbf{J}^T \mathbf{r}_c$, we obtain

$$\begin{aligned} \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T}{\partial \mathbf{P}} \left(\sum_{k=1}^m \frac{(d_j \kappa_{j,k})^2 \mathbf{T}_{j,k} \mathbf{T}_{j,k}^T}{(1 + d_j \kappa_{j,k})^2} + \sum_{k=m+1}^s \mathbf{N}_{j,k} \mathbf{N}_{j,k}^T \right) \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}} \delta \mathbf{P} = \\ \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T}{\partial \mathbf{P}} \left(\sum_{k=1}^m \frac{d_j \kappa_{j,k} \mathbf{T}_{j,k} \mathbf{T}_{j,k}^T}{1 + d_j \kappa_{j,k}} + \sum_{k=m+1}^s \mathbf{N}_{j,k} \mathbf{N}_{j,k}^T \right) (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j). \end{aligned}$$

The normal equation represents the following geometric minimization

$$\min_{\mathbf{P}} \sum_{j=1}^n \left\{ \sum_{k=1}^m \frac{(d_j \kappa_{j,k})^2}{(1 + d_j \kappa_{j,k})^2} [\mathbf{T}_{j,k}^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2 + \sum_{k=m+1}^s [\mathbf{N}_{j,k}^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2 \right\} \quad (4.21)$$

We will call it CDM (**C**urvature **D**istance **M**inimization).

2. Drop $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P} \partial \mathbf{t}}$ and $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{t}^2}$. It is easy to verify in this case that the normal equation corresponds to the following geometric minimization

$$\min_{\mathbf{P}} \sum_{j=1}^n \left\{ \sum_{k=m+1}^s [\mathbf{N}_{j,k}^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2 \right\} \quad (4.22)$$

Since (4.22) only penalizes the squared distance from data point \mathbf{X}_j to the tangent space $\mathbf{T}_{j,p}(\mathbf{C})$ we call it GTDM (**G**eneralized **T**angent **D**istance **M**inimization). This scheme does not suffer from the ill conditioning problem of high dimension parametric curves/surfaces ($m < s - 1$) which is mentioned before. Still using the same example at the end of the last subsection, let one normal be \mathbf{N}_j and another normal be $\mathbf{N}_z = (0, 0, 1)^T$. The variable c will appear in $\mathbf{N}_z^T (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)$, so that rank deficiency of the normal equation is avoided.

4.3.3 SDM - modified Hessian approximation

So far our discussion is based on Gauss-Newton methods. Now we look at the Hessian directly. In Chapter 2 we proposed a curvature based squared distance minimization method called SDM where the Hessian is modified to be definite-positive. We do not go into the details and just describe the basic idea here. For each $\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j$, its second-order derivatives $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}^2}$ and $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P} \partial \mathbf{t}}$ are dropped. So the modified Hessian is

$$\tilde{\mathbf{H}} = \sum_{j=1}^n \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)^T}{\partial \mathbf{P}} \left[\sum_{k=1}^m \frac{d_j \kappa_{j,k}}{1 + d_j \kappa_{j,k}} \mathbf{T}_{j,k} \mathbf{T}_{j,k}^T + \sum_{k=m+1}^s \mathbf{N}_{j,k} \mathbf{N}_{j,k}^T \right] \frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}}. \quad (4.23)$$

The corresponding geometric minimization is

$$\min_{\mathbf{P}} \sum_{j=1}^n \left\{ \sum_{k=1}^m \frac{d_j \kappa_{j,k}}{1 + d_j \kappa_{j,k}} [\mathbf{T}_{j,k}^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2 + \sum_{k=m+1}^s [\mathbf{N}_{j,k}^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2 \right\}. \quad (4.24)$$

Remark 4.2. If, besides $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P}^2}$ and $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P} \partial \mathbf{t}}$, we also drop $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{t}^2}$ from the Hessian, SDM will become GTDM. Thus GTDM also is an approximation of the Hessian.

4.3.4 Comparisons

We summarize the geometric minimization schemes introduced in previous sections in Table 4.1 and compare them in several aspects.

TABLE 4.1: Geometric minimization schemes

Method	Geometric terms
PDM	$[\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j]^2$
TDM	$[\mathbf{N}_j^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2$
GTDM	$\sum_{k=m+1}^s [\mathbf{N}_{j,k}^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2$
CDM	$\sum_{k=1}^m \frac{(d_j \kappa_{j,k})^2}{(1 + d_j \kappa_{j,k})^2} [\mathbf{T}_{j,k}^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2 + \sum_{k=m+1}^s [\mathbf{N}_{j,k}^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2$
SDM	$\sum_{k=1}^m \frac{d_j \kappa_{j,k}}{1 + d_j \kappa_{j,k}} [\mathbf{T}_{j,k}^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2 + \sum_{k=m+1}^s [\mathbf{N}_{j,k}^T \cdot (\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j)]^2$

Computational cost: The standard coordinate-based Gauss-Newton method (or GN for short) is the most expensive method because of computations of the second-order derivatives $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{t}^2}$ and $\frac{\partial^2 \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{P} \partial \mathbf{t}}$. Since CDM and SDM have similar expressions, their computational costs are the same. GTDM only involves computation of $\frac{\partial \mathbf{C}(\mathbf{P}; \mathbf{t}_j)}{\partial \mathbf{t}}$ for constructing the normal space if $m < s - 1$. TDM and PDM do not need to compute any derivative of $\mathbf{C}(\mathbf{P}; \mathbf{t})$ with respect to \mathbf{t} . thus they are more efficient than the others in constructing the approximated Hessian.

Applicability: With proper step-size control or combining Levenberg-Marquardt methods, most methods are suitable for general parametric curve/surface fitting. Only TDM may have problems in fitting high dimension parametric curves/surfaces, i.e, when $m < s - 1$.

Convergence: Because GN and TDM are standard Gauss-Newton methods, they show quadratic convergence for zero residual problems, super-linear convergence for small residual problems and linear convergence in other cases. For our modified methods CDM and GTDM, they also have the same convergence as TDM. One can see that when $\|\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j\|$ approaches zero, the second-order derivatives in Equation. (4.19) can be ignored so that the Hessian is still well approximated. Unfortunately PDM is an alternating method which is a typical optimization technique for solving a separable nonlinear least squares problem and is known to have only linear convergence [14].

Remark 4.3. For high dimension curves/surfaces fitting, i.e. $s > 3$, the principle curvature computation can be expensive. In this case GTDM is a good candidate under the consideration of performance and effectiveness. Also when $m = s - 1$, GTDM is reduced to TDM actually.

4.4 Numerical Experiments

Now we compare the methods introduced in Section 4.3: PDM, TDM, CDM, GTDM, GN, SDM. For demonstrating the effectiveness of GTDM, we choose a planar ellipse in 3D space as our parametric models and a point cloud with different scale noises (For general comparison in 2D/3D curve and surface fitting, we refer the reader to the references [1, 8, 84, 151]). In our implementation the Levenberg-Marquardt method is integrated.

Example: We consider fitting an ellipse to 200 data points sampled from an ellipse: $(\cos \frac{2\pi i}{200}, 2 \sin \frac{2\pi i}{200}, 0), i = 0, 1, \dots, 199$ in 3D. The parametric ellipse has the following form which involves rotation and translation

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{R}_x \cdot \mathbf{R}_y \cdot \mathbf{R}_z \cdot \begin{pmatrix} a \cos t \\ b \sin t \\ 0 \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix}$$

$$\text{Where } \mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix}, \mathbf{R}_y = \begin{pmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{pmatrix},$$

$$\mathbf{R}_z = \begin{pmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}. \text{ The shape parameters are } \mathbf{P} = [a, b, c_x, c_y, c_z, \alpha, \beta, \gamma]. \text{ We}$$

choose four cases to illustrate the convergence of each method, with the following initial values for \mathbf{P}

- Case 1: $\mathbf{P} = [3.1, 1.0, 1.0, 2.0, 0.2, 4.0, 1.0, 6.0]$;
- Case 2: $\mathbf{P} = [0.1, 4.0, 2.0, 0.0, 1.0, 1.0, -1.0, 2.0]$;
- Case 3: same \mathbf{P} as in Case 1 but perturb the data points with random noise distributed uniformly in $[-0.001, 0.001]$;
- Case 4: same \mathbf{P} as in Case 1 but perturb the data points with random noise distributed uniformly in $[-0.1, 0.1]$. (See Figure. 4.1)

Figure.4.2 shows that the average error versus the number of iterations of the six methods. The average error is defined as $\sqrt{\frac{\sum_{j=1}^n \|\mathbf{C}(\mathbf{P}; \mathbf{t}_j) - \mathbf{X}_j\|^2}{n}}$. From the figure we find the surprising fact that GTDM is much better than the other methods. It converges very fast and only needs several iterations. The behaviors of TDM in the four cases are different. In Case 2 and 3 TDM is easy to be trapped in the local minimum. In Case 4,

since the data points are not nearly planar, TDM shows good performance. In all the cases GN is a little better than CDM but is still slower than GTDM and SDM.

From our experience in 2D/3D curves and surfaces fitting, actually there is no strong evidence and theoretical guarantee that shows which method (TDM, CDM, GTDM, GN, SDM) is best for most fitting problems since the integrated step-control strategy like line search or the Levenberg-Marquardt method affects the behavior and unexpected local minimum may stop the optimization. Also for large residual problems all the methods exhibit linear convergence which is similar to PDM. For instance, see Case 4 of the example. But in general GTDM is as good as the others at least in most cases. By considering the computational cost and overall performance, we strongly recommend GTDM for general parametric curve and surface fitting including parametric sub-manifold fitting (i.e, when $m < s - 1$) due to its clear geometric meaning and its simplicity since it does not need to compute the principle curvatures and directions.

4.5 Conclusions

A systematic geometrical and optimizational analysis on least squares orthogonal distance fitting of parametric curves and surfaces is presented. We give the geometric characterization of existing techniques and propose two modified versions based on geometric meanings. We show how principle curvature and directions are embedded in optimization methods. The presented geometric understanding of optimization techniques will benefit efficient and effective curve/surface fitting. Also for further research, it is interesting to study the geometry behind methods for implicit curve/surface fitting.

Claim: the presented material in this chapter has been published in [85].

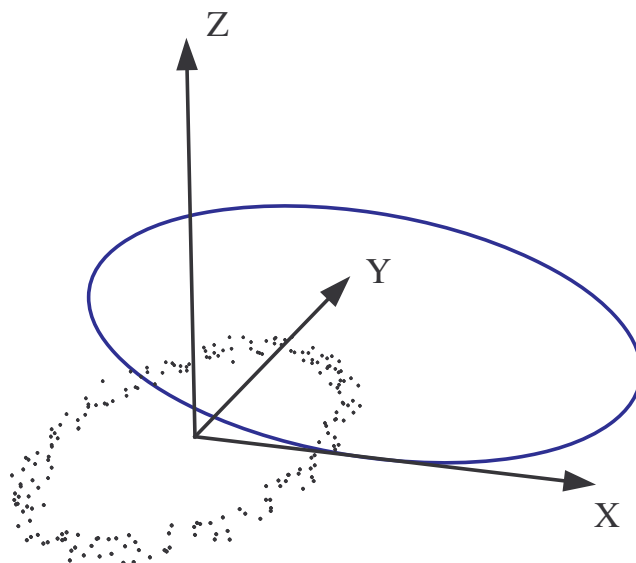


FIGURE 4.1: The initial ellipse and data points of Case 4.

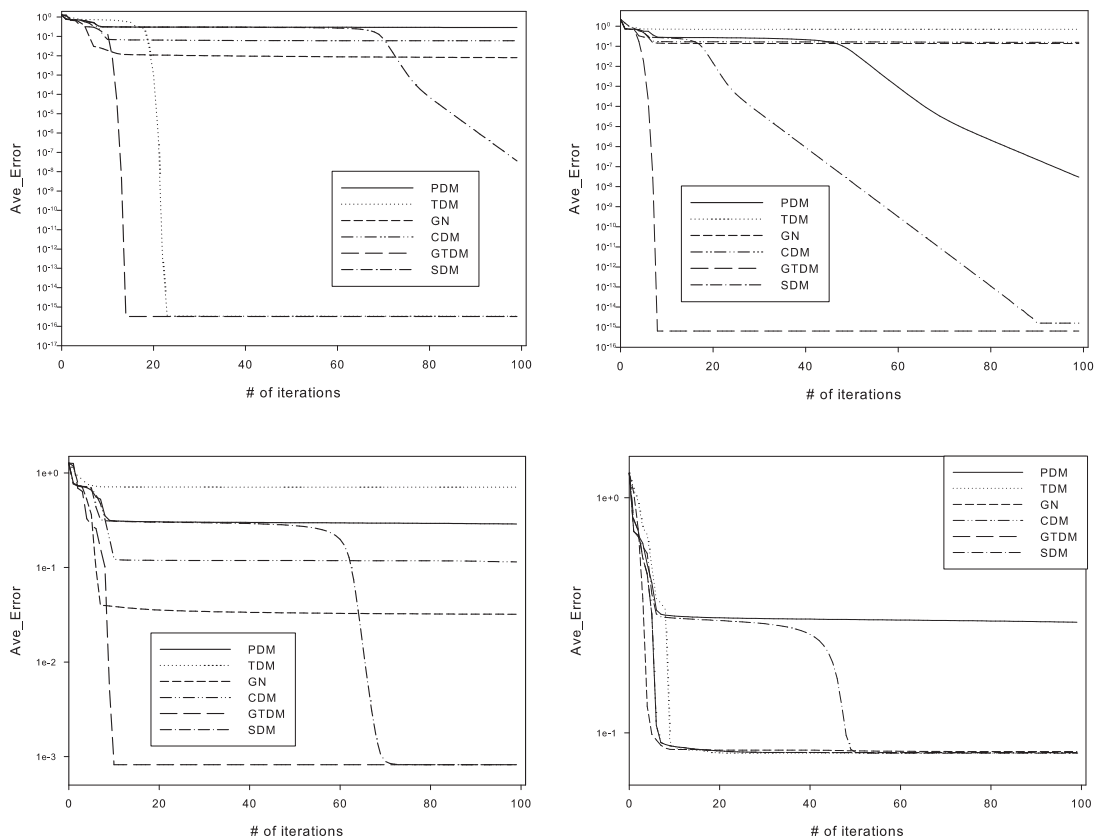


FIGURE 4.2: Comparisons of the six methods on a set of 200 data points. Upper left: Case 1; upper right: Case 2; lower left: Case 3; lower right: Case 4.

Chapter 5

Computing Centroidal Voronoi Tessellation with Superlinear Convergence

5.1 Introduction

5.1.1 Problem setting and previous work

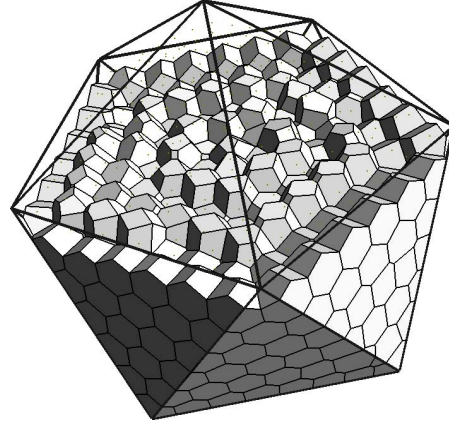
Lloyd's optimal quantizer :

Lloyd's relaxation (invented in 1957, published later [86]) is an algorithm that computes the optimal sampling of a signal (e.g. optimal colormap of an image, optimal placement of cellphone antennas etc...). It is also used to optimize the locations of vertices in mesh generation algorithms [48]. Given a domain $\Omega \subset \mathbb{R}^N$, Lloyd's algorithm computes an optimal sampling of Ω (also called an *optimal quantizer*) by minimizing a certain energy F (also called *quantization noise power*), defined over the cells of the Voronoi tessellation of the samples :

$$F(\mathbf{X}) = \sum_{i=1}^n f_i(\mathbf{X}) = \sum_{i=1}^n \int_{\Omega_i} \rho(z) \|z - \mathbf{x}_i\|^2 dz, \quad (5.1)$$

where $\mathbf{x}_i = (x_i, y_i, z_i, \dots) \in \mathbb{R}^N$ denotes a sample (also called a *seed*), $\Omega_i = \text{Vor}(i) \cap \Omega$ denotes the intersection between the Voronoi cell of sample i and the domain Ω , and $\rho(\cdot)$ is a "background density" function.

The vector $\mathbf{X} \in \mathbb{R}^{nN}$ gathers all the coordinates at all the samples (for $N = 3$, $\mathbf{X} = (x_1, y_1, z_1, \dots, x_n, y_n, z_n)$). In a certain sense, f_i expresses the compactness (or inertia momentum) of the Voronoi cell associated with the seed i . This property is suitable in many applications. For instance, in sampling theory, we can imagine that Ω represents a space that needs to be approximated (e.g., a color space) and that the \mathbf{x}_i 's correspond to samples (e.g., the elements of a colormap). The cells $\Omega_i = \{z \in \Omega | \forall j \neq i, \|z - \mathbf{x}_i\| < \|z - \mathbf{x}_j\|\}$ correspond to the subsets of the original space Ω that will be mapped to the same sample \mathbf{x}_i . Minimizing the energy F ensures that each sample \mathbf{x}_i is representative of the same amount of information in Ω . In the inset, the space Ω to be approximated is the black icosahedron. As can be seen, the Voronoi cells are very regular, and distributed in a way that preserves the structure of Ω . These properties are also suitable for geometry processing applications. Note that the optimal quantization problem is an instance of *locational optimization*, a more general problem important in geography and dynamic systems, used to compute the optimal distribution of facilities with the lowest cost [100].



Centroidal Voronoi Tessellations :

As explained in [7, 100], the gradient of F is given by :

$$\frac{\partial F}{\partial \mathbf{x}_i} = 2m_i(\mathbf{x}_i - \mathbf{c}_i) \quad (5.2)$$

where $m_i = \int_{z \in \Omega_i} \rho(z) dz$ and \mathbf{c}_i denote respectively the mass and centroid of Ω_i . With this formula of the gradient, it is easy to check that if each sample \mathbf{x}_i coincides with the centroid of its Voronoi cell Ω_i , then the \mathbf{x}_i 's correspond to a critical point (zero gradient) of the energy F . This can be simply written as :

$$\forall i, \quad \mathbf{x}_i = \mathbf{c}_i = \frac{\int_{\Omega_i} \rho(z) z dz}{\int_{\Omega_i} \rho(z) dz} \quad (5.3)$$

This is a system of nonlinear equations since the boundaries of any VD cell Ω_i are determined by all the seeds \mathbf{x}_i .

A Voronoi tessellation that satisfies Equation 5.3 is referred to as a CVT (Centroidal Voronoi Tessellation) [10, 48, 100]. This notion was recently popularized in the domain of meshing and geometry processing [5, 45, 46]. Similar to image segmentation, the concept of CVT and its algorithms have been transported to mesh segmentation and simplification. Peyré and Cohen [105] extend CVT via defining geodesic metrics on

mesh surfaces. Due to the heavy computation of geodesic paths and centroids, CVT under Euclidean metric is more efficient in practice. Isotropic and anisotropic CVT are considered by Du et al. [47, 49] and Valette et al. [144, 145]. The algorithm employed in the above applications is Lloyd’s method or the probabilistic method, with slow convergence rate.

The remark that a CVT is a critical point of F naturally leads us to explain the historical algorithm that constructs a CVT, namely *Lloyd’s relaxation* [86], which is a fixed-point iteration, that simply consists in iteratively moving all the seeds \mathbf{x}_i to the centroid of their Voronoi cells. Its convergence to a CVT is proved in some particular cases by Du *et.al* [48, 50]. They show that besides being a fixed-point iteration that solves Equation 5.3, Lloyd’s method can be understood as a gradient descent that always decreases the energy F without steplength control. The same type of analysis can be applied to the discrete k-means clustering version [101].

At this point, there can be two different point of views about the definition of CVT.

- **Geometric characterization :** CVT is the solution of a system of non-linear equations that says that each seed \mathbf{x}_i coincides with the centroid of its cell \mathbf{c}_i (Equation 5.3).
- **Variational characterization :** CVT is a critical point of the energy F in Equation 5.1. In many applications it is desirable to compute the minimizer of F , rather than just a critical point, so that in the CVT the cells are as compact as possible. This essentially excludes those nonstable critical points that are not minimizers of F (more on this in Section 5.2.1). It is this type of CVT that we aim to compute.

The distinction between these two point of views is at the heart of our approach. From the variational point of view, Lloyd’s algorithm is a gradient descent, with linear speed of convergence. We show how fully studying this variational point of view leads to new methods for computing CVT with superlinear convergence and show its superior performance over Lloyd’s method. We believe this represents a fundamental contribution to CVT computation that is likely to have important impact in many areas of application.

5.1.2 The variational point of view

Lloyd’s method is a gradient-descent method that minimizes the function F (Equation 5.1) with linear convergence. To go beyond Lloyd’s method, we consider computing

CVT still by minimizing the function F (Equation 5.1), but this time using a quasi-Newton method with *superlinear* convergence rate. It is well known that a better speed of convergence can be obtained by using higher-order methods (e.g., Newton’s method and its variants). However, due to the believed lack of smoothness of F and the apparent complexity in computing Hessian for a large scaled CVT problem, there has been little successful attempt in the literature in this direction [44, 70]. The Lloyd-Newton algorithm [44] is essentially based on the above geometric characterization of CVT, since it uses a different function whose minimizers include all the critical points of F , especially all those unstable ones. Our experiments confirm that Lloyd-Newton often gets trapped by such unstable critical points that are not local minimizers of F , unless a large number of Lloyd’s iterations are used first for initialization. We note that our method is also different from Alliez *et.al*’s variational tetrahedral meshing [5], which uses a gradient-descent method.

Minimizing the piecewise-defined CVT energy function F is a numerical optimization problem has several unconventional aspects. This raises interesting questions about the structure and continuity of F , as we shall explain. Newton’s iteration for non-linear optimization considers a 2^{nd} -order approximation of F :

$$F(\mathbf{X} + \delta_{\mathbf{X}}) \simeq F^*(\delta_{\mathbf{X}}) = F(\mathbf{X}) + \nabla F \delta_{\mathbf{X}}^T + 1/2(\delta_{\mathbf{X}}^T \mathbf{H} \delta_{\mathbf{X}})$$

where $\delta_{\mathbf{X}}$ denotes a small displacement around \mathbf{X} , ∇F denotes the gradient of F , and \mathbf{H} its Hessian. Newton’s iteration finds the step vector $\delta_{\mathbf{X}}$ that minimizes the ”model function” $F^*(\delta_{\mathbf{X}})$ as follows :

$$\begin{aligned} \text{solve } \mathbf{H} \delta_{\mathbf{X}} &= -\nabla F(\mathbf{X}) \\ \mathbf{X} &\leftarrow \mathbf{X} + \delta_{\mathbf{X}} \end{aligned}$$

As can be seen, since it uses second-order derivatives, Newton’s method has a chance to work only if F is at least C^2 [99]. Newton’s method is not suitable for large scale problems due to its costly computation of the full Hessian. Therefore, in practice, one often uses quasi-Newton methods to deal with large scale problems.

Despite the recent result [37] that shows that the function F is C^1 , a major impediment to the development of fast Newton-like methods for CVT computation is the common belief that the CVT function lacks the required C^2 smoothness [70]. We shall show that this belief is wrong, by proving that F is almost always C^2 . More specifically, F is C^2 in any convex domain in 2D and 3D. When the domain is non-convex, it is still C^2 in most cases, except in few seldom encountered configurations where it becomes C^1 . Furthermore, these results carry over to the constrained or restricted CVT

problem on mesh surfaces. This surprising result on the smoothness of F provides the necessary justification in our investigation on devising an efficient quasi-Newton method to accelerate CVT computation.

Before entering the heart of the matter, we need to explain why this result is surprising, and why studying the continuity of F is a difficult problem. If one wants to evaluate F and its derivatives for a specific value of the parameters \mathbf{X} , it is necessary to construct the Delaunay triangulation of the vertices defined by \mathbf{X} , then evaluate the integrals over each cell of its dual Voronoi diagram (see Equation 5.1), then differentiate them. The expression of these integrals is quite complicated, since the vertices of the integration domains Ω_i are the circumcenter of the Delaunay simplices. If we now imagine that we move one of the vertices, then when the combinatorics of the Delaunay triangulation changes, the formula of F changes as well! In other words, F is piecewise defined in the space of \mathbf{X} , and the pieces correspond to the subsets of \mathbf{X} where the combinatorics of the Delaunay triangulation does not change. In a Delaunay triangulation, a combinatorial change corresponds to an edge flip, that occurs in 2D when four vertices become cocyclic (or 5 vertices in 3D). These “degenerate” configurations are often considered as nuisances in mesh generation. In contrast, in our case, they define “gateways” that connect the pieces of \mathbf{X} space. The common face shared by two pieces corresponds to a configuration where 4 vertices are cocyclic (or 5 in 3D). Crossing such a gateway results in dramatic changes in the expression of F . Surprisingly, as explained in Section 5.2, the continuity of F is C^2 at such transitions.

Contributions

We shall show that the CVT energy function is C^2 in convex domains in 2D and 3D and almost always C^2 in non-convex domains. This surprising discovery has motivated us to develop Newton-like optimization methods for efficient computation of CVT in 2D, 3D, and on mesh surfaces. In summary, we make the following fundamental contributions in both theory and application:

- We prove that the piecewise CVT function is C^2 in a 2D convex region or a 3D convex domain. Using the similar proof technique, we show that the CVT function defined on a smooth surface is still C^2 ;
- we accelerate the CVT computation by applying quasi-Newton methods that are shown to be much more efficient than Lloyd’s method both in convergence rate and computational time;
- we developed an efficient quasi-Newton method for computing the constrained CVT and restricted CVT problems on mesh surfaces for remeshing and simplification. Unlike other Lloyd-like CVT methods on meshes that perform only face

clustering, we treat the mesh as a piecewise smooth domain in the optimization and enhance the accuracy of the Voronoi diagram computation by performing face-splitting to achieve high-quality results.

5.2 Continuity Analysis

In this section, after recalling the variational formulation, we will give the key theorems that characterize the smoothness of the CVT energy function.

5.2.1 Variational formulation

We consider the variational characterization of CVT as a minimizer of the energy F (Equation 5.1), that can be written as :

$$\text{minimize } F(\mathbf{X}) = \sum_{i=1}^n f_i(\mathbf{X}) = \sum_{i=1}^n \int_{\Omega_i} \rho(z) \|z - \mathbf{x}_i\|^2 dz \quad (5.4)$$

where the density function $\rho(\cdot) \geq 0$ satisfies $\sup_{z \in \Omega} \rho(z) \leq \gamma$ (γ is a constant value).

The variable $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) \in \Gamma = \Omega^{nN}$ is an $N \times n$ -dimensional point. A natural assumption is that no two seeds points are equal, i.e. $\forall i \neq j, \mathbf{x}_i \neq \mathbf{x}_j$. Hence, if we denote $B = \{\mathbf{X} | \mathbf{x}_i = \mathbf{x}_j, i \neq j\}$, we will only study the behavior of $F(\mathbf{X})$ in the domain $\Gamma_c := \Gamma \setminus B$.

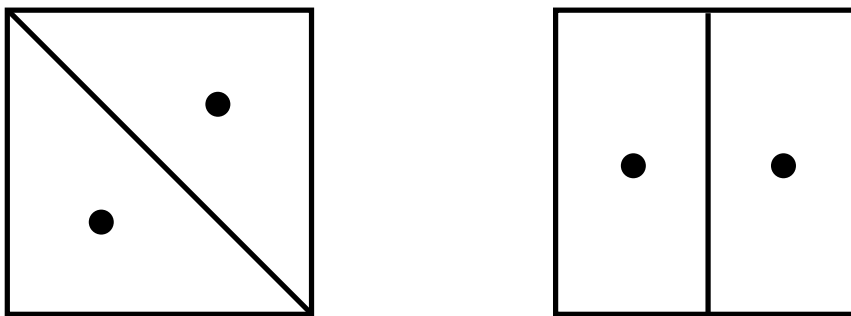


FIGURE 5.1: (Left): an unstable critical configuration for CVT of 2 samples in a square; (right) a stable configuration, which is a minimizer of the CVT function.

Remark 5.1. The geometric characterization of CVT means that the seed points are centroids of their cells. Actually this geometric characterization is only a necessary condition when considering the variational point of view. Although this corresponds to a critical point of F , the geometric condition cannot guarantee that the obtained cell structure is stable [48]. For instance, the configuration shown in Figure 5.1(left)

is unstable; it corresponds to a saddle point of F (i.e., the Hessian has some negative eigenvalues) and so is not a local minimizer of F . A stable configuration (i.e., with a positive-definite Hessian) is shown on the right.

5.2.2 Smoothness of F

We refer the reader to Section 5.5 for the proof of the following theorem.

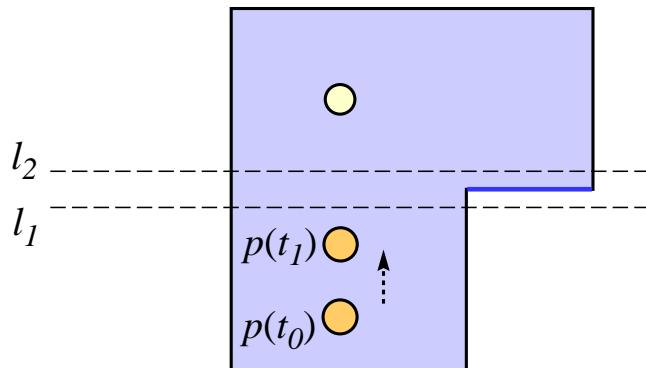
Theorem 5.2. *The 2D CVT function is C^2 in Γ_c if Ω is convex.*

With similar mathematical and geometric technique, we can prove the same conclusion about 3D CVT.

Theorem 5.3. *The 3D CVT function is C^2 in Γ_c if Ω is convex.*

Remark 5.4. In the more general locational optimization problem, the integration function is a smooth function $W(\|z - \mathbf{x}_i\|)$, i.e., $F(\mathbf{X}) = \sum_{i=1}^n \int_{z \in \Omega_i} \rho(z) W(\|z - \mathbf{x}_i\|) dz$. Theorem 5.2 and Theorem 5.3 still hold in this case.

In general, for a non-convex domain Ω , Theorems 5.2 and 5.3 no longer hold because the C^2 smoothness is spoiled when a continuous part of $\partial\Omega$ belongs to a face of CVT cells. In that case, the CVT function is only C^1 . We illustrate a 2D case in the right figure. Suppose that there are



only two points in the shown 2D non-convex domain Ω . Let the point \mathbf{p} move from $\mathbf{p}(t_0)$ to $\mathbf{p}(t_1)$ vertically. When the bisector line crosses through the blue concave edge, F is only C^1 . The same situation can occur in 3D as well. It can be shown that this is the only situation where C^2 smoothness of F is not preserved. It is easy to see that such a situation rarely occurs in practice, since in most cases the faces of the Voronoi cell are normally not parallel to the domain boundary when there are sufficiently many seeds with a reasonable distribution (see Figure 5.2).

5.2.3 Experimental evidence of continuity

We use two simple examples to illustrate the smoothness of 2D and 3D CVT function. We intentionally chose an examples with degenerate configurations in 2D to show that

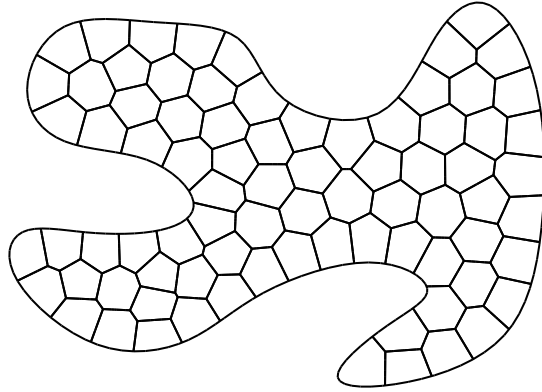
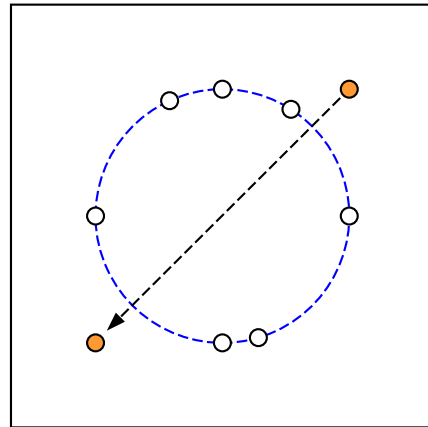


FIGURE 5.2: A 2D non-convex CVT after several Lloyd's iterations. The CVT energy is C^2 around this configuration.

sufficient continuity (C^2) is obtained even if multiple edge flips are involved simultaneously.

Example 5.1. 2D case: Here Ω is the square box $[-1, 1] \times [-1, 1]$ with eight seeds. One of the seeds \mathbf{p} moves along a trajectory $\mathbf{p}(t)$ shown in the inset on the right. The Voronoi diagram changes during the motion of $\mathbf{p}(t)$. Figure 5.3 shows the graphs of the CVT $F(t)$ and its derivatives $F'(t), F''(t)$ with respect to the motion parameter t (the explicit derivative formulae can be found in [7, 48, 70, 100]). It is clear that $F''(t)$ is a C^0 function, as the consequence of the C^2 smoothness of $F(t)$. The rapid changes in $F''(t)$ correspond to the transition of VD and boundary affects.



Example 5.2. 3D case: Here Ω is a $[-1, 1]^3$ cube and eight seed points in it. The seed \mathbf{p} (the yellow ball) moves along a trajectory $\mathbf{p}(t)$ shown in the Figure 5.4. The curves of $F(t), F'(t)$ and $F''(t)$ are shown.

5.3 Numerical Optimization

In this section we first present previous approaches on Newton's method and then propose to use a quasi-Newton method – the BFGS method – for CVT computation. Since 2D and 3D CVT functions are C^2 , we are able to apply standard Newton-like nonlinear optimization techniques [99] to find a local optimum of the CVT function $F(\mathbf{X})$ by solving $\mathbf{H}\delta\mathbf{x} = -\nabla F$ iteratively. We remind that the components of the gradient of $F(\mathbf{X})$ are given by $\partial F / \partial \mathbf{x}_i = 2m_i(\mathbf{x}_i - \mathbf{c}_i)$ (Equation 5.2). Following the explicit formulae

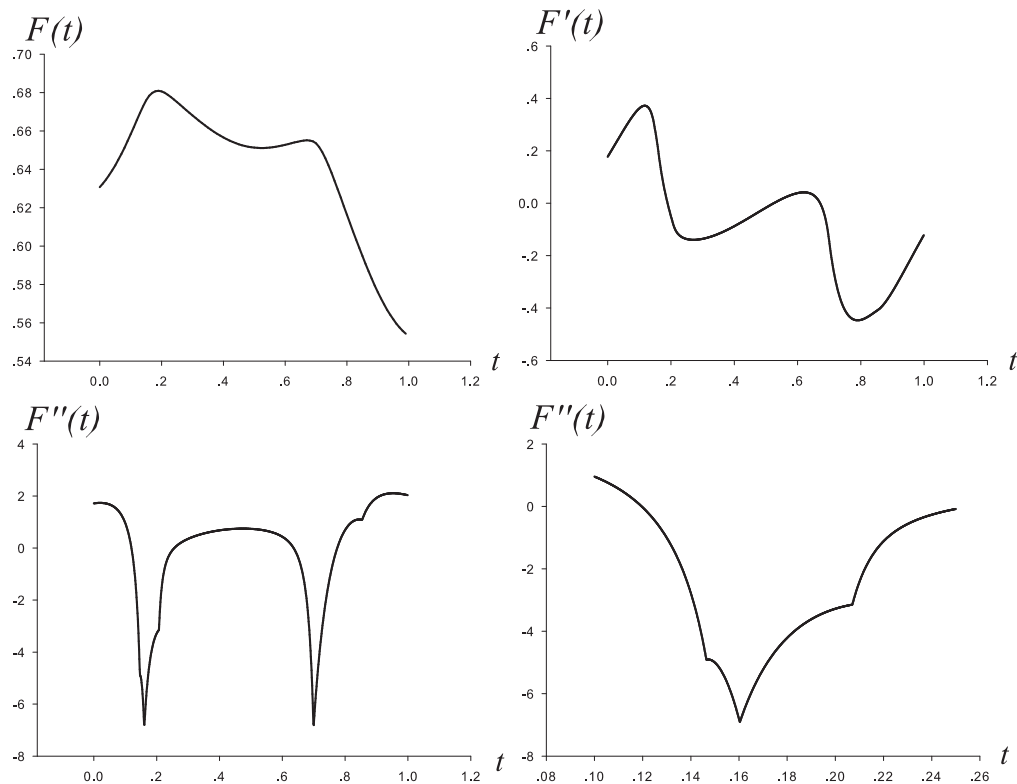


FIGURE 5.3: Illustration of 2D CVT smoothness. The bottom right figure is a zoom-in view of $F''(t)$ in $[0.08, 0.26]$ which shows $F''(t)$ is indeed C^0 .

in [7, 100], the Hessian of $F(\mathbf{X})$ can be computed. (We do not list these formulae here as they are quite complex and lengthy.) Although the Hessian matrix \mathbf{H} is sparse, it is in general not positive-definite. Iri et al. [70] modified the Hessian by removing most mixed-partial derivatives with respect to different seed points so that the Hessian is symmetric positive definite (SPD). But the modified Hessian is not a good approximation of the original one and this slows down their implementation.

quasi-Newton method

With proper modification such as modified Cholesky factorization [99, 131], Newton's method is applicable in small and median size optimizational problems. But the main disadvantage of Newton's method is that the computation of the Hessian matrix, its construction and modification are costly. Therefore it is normally not recommended when there is a large number of seed points. For large scale CVT computation, some quasi-Newton methods, such as **BFGS**, are good alternatives, since they only query the function value and the gradient, involve only sparse matrix-vector multiplications. We note that the computational cost of the gradient is the same as Lloyd's method (see Equation. 5.2). This means that there is no extra costly computation in quasi-Newton methods. We have integrated a popular large scale bounded-constrained quasi-Newton method **L-BFGSB** [27] that approximates \mathbf{H}^{-1} by a sparse SPD matrix. This method

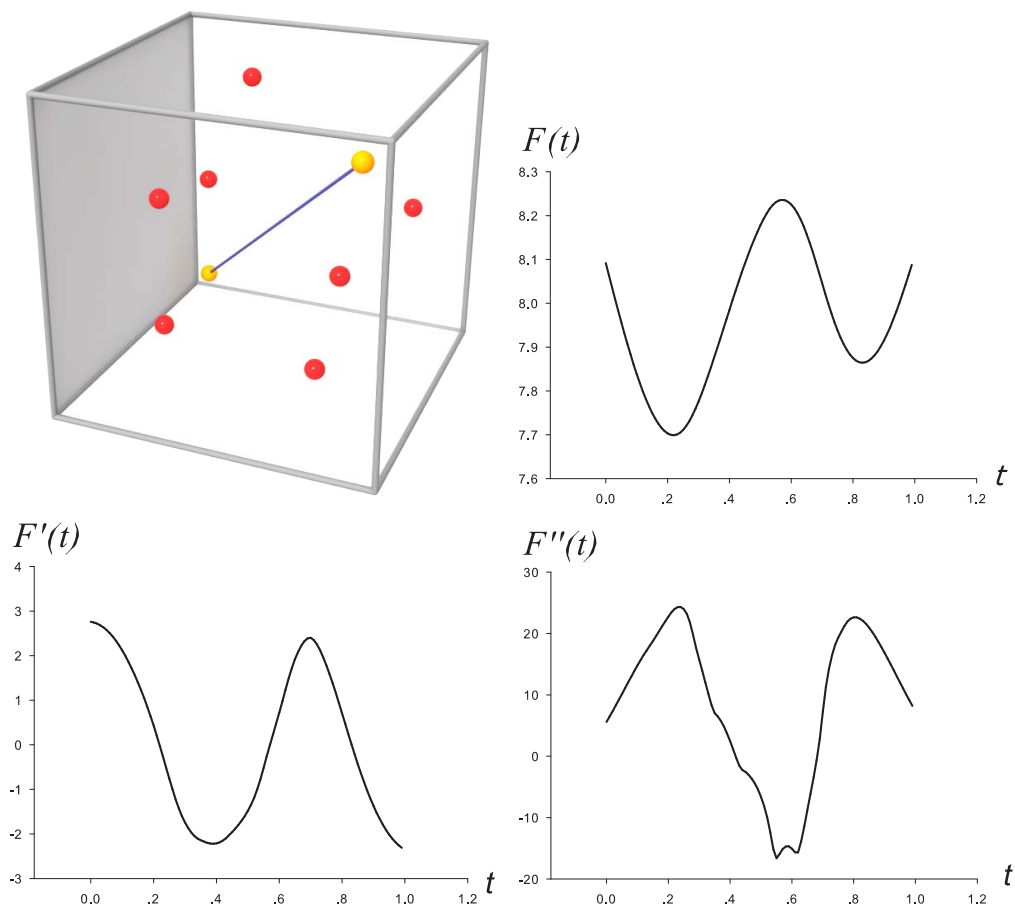


FIGURE 5.4: Illustration of 3D CVT smoothness.

reduces the computational time greatly while maintaining the super-linear convergence rate.

5.3.1 Numerical examples

Since Lloyd's method is currently the most commonly used, in this section, we will compare it with our CVT computation based on a quasi-Newton method (i.e., L-BFGSB). We will show that it is about one order of magnitude faster than Lloyd's method. To avoid converting the boundary constraints explicitly in L-BFGSB, especially when the boundary of the domain is more complex than a square/cube, we reduce the increment of the seed points if the L-BFGSB iteration moves them outside the domain (we have tested both approaches, and the latter is both simpler to implement and more efficient). Our tests were run on a PC with a 2.2GHz CPU and 2GB RAM and our implementation integrates [QHULL](#) to compute Voronoi cells. In our test, the density function is a constant function.

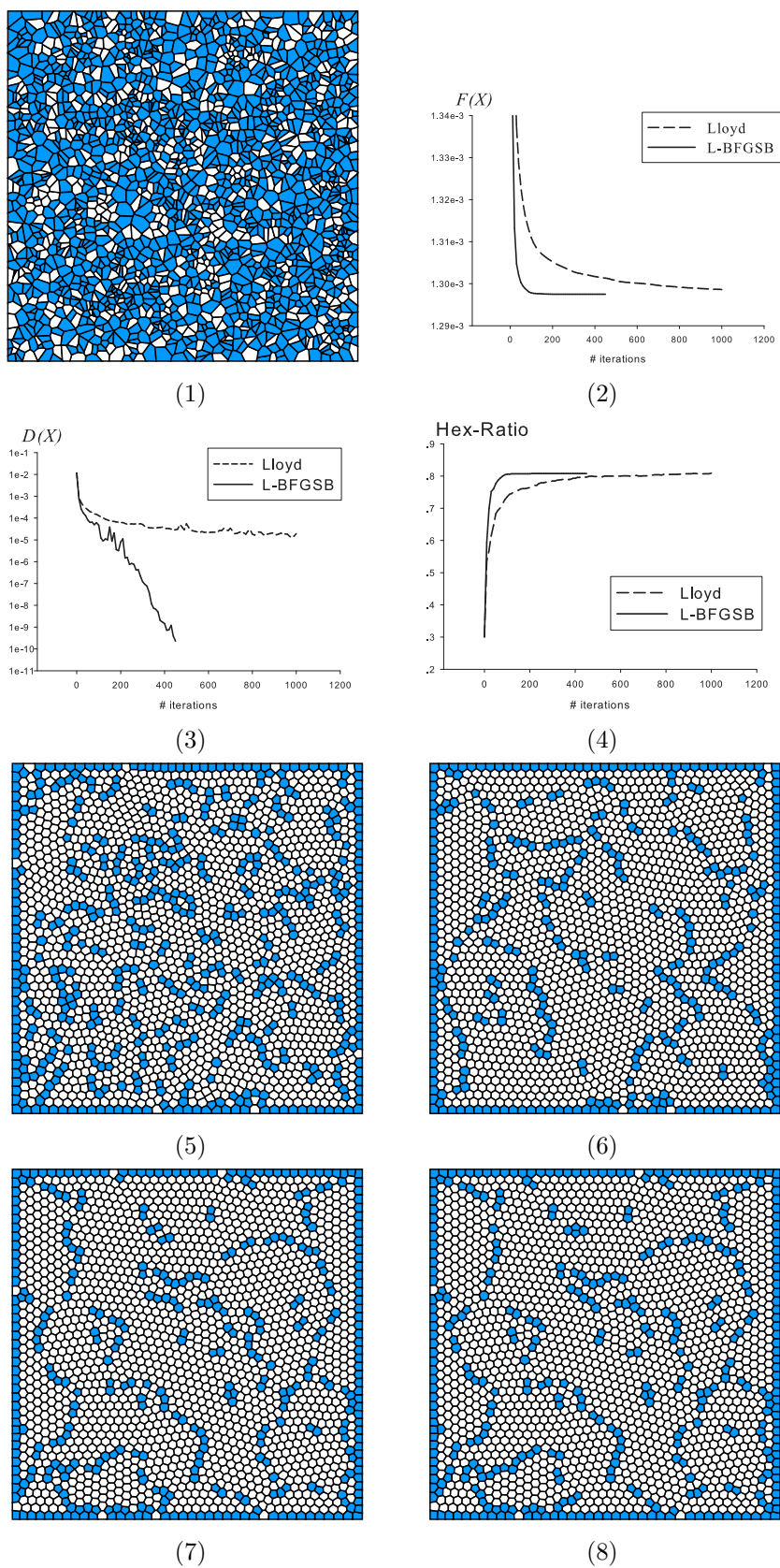


FIGURE 5.5: (1): initial Voronoi tessellation. Non-hexagonal cells are shown in blue; (2): $F(\mathbf{X})$ vs the number of iterations; (3): $D(\mathbf{X}) = \sqrt{\sum_{i=1}^{2000} (\mathbf{x}_i - \mathbf{c}_i)^2 / 2000}$ vs the number of iterations; (4): the ratio of the hexagonal cells; (5): the result after 100 iterations by Lloyd's method; (6) the result after 1000 iterations by Lloyd's method; (7) the result after 100 iterations by L-BFGSB; (8) the result after 1000 iterations by L-BFGSB.

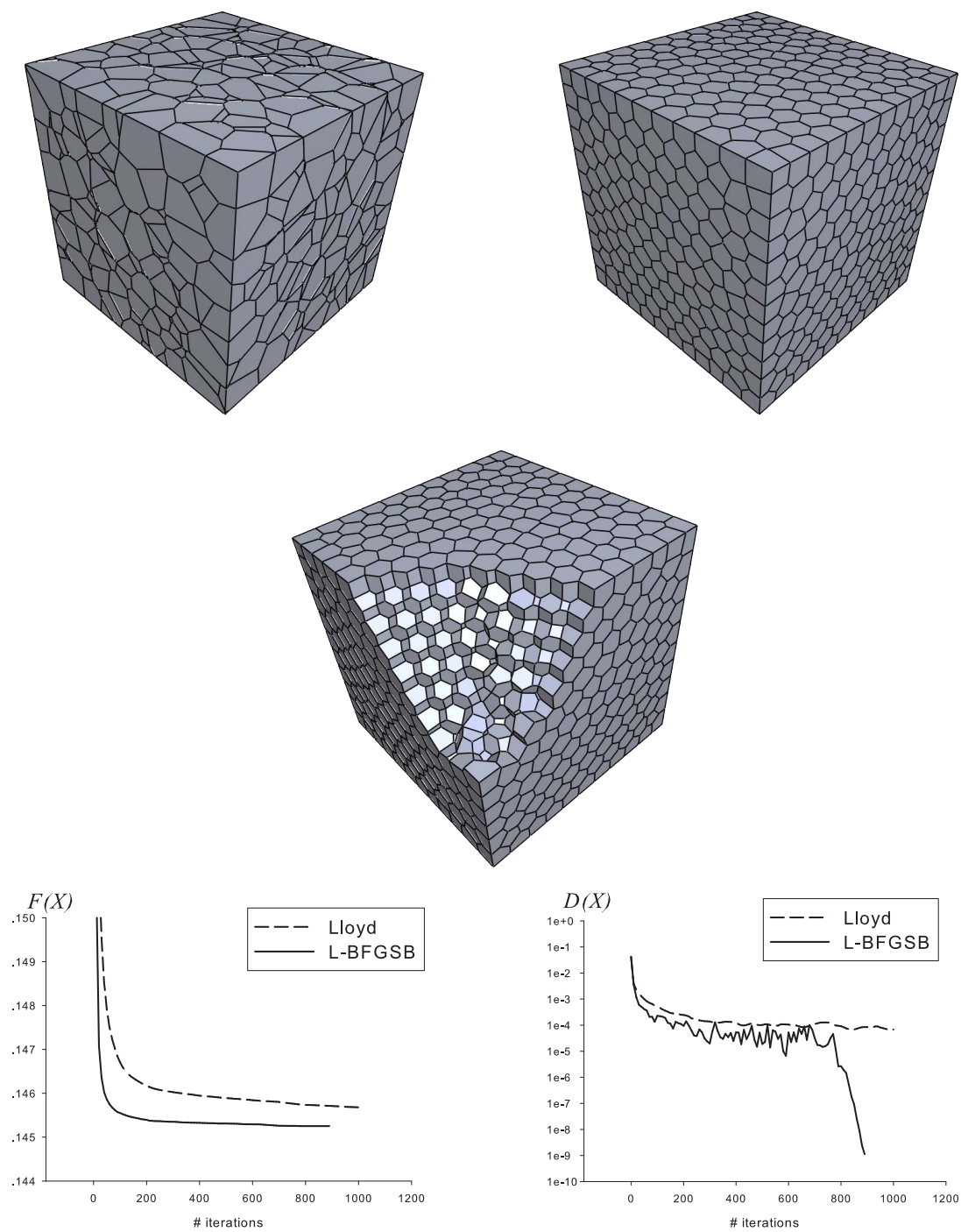


FIGURE 5.6: Initial Voronoi tessellation, result after 890 iterations of L-BFGSB, $F(\mathbf{X})$ and $D(\mathbf{X}) = \sqrt{\sum_{i=1}^{2000} (\mathbf{x}_i - \mathbf{c}_i)^2 / 2000}$ vs the number of iterations. The middle figure shows the internal structure of the final result.

Example 5.3. *2D CVT: 2000 seed points are sampled from $[-1, 1]^2$ randomly and the maximum iteration number is set 1000. Figure 5.5 shows the comparison of $F(\mathbf{X})$ with respect to the iteration number. After 625 iterations, L-BFGSB has met its stop criterion.*

453 iterations of L-BFGSB cost 98.037 secs and 1000 iterations of Lloyd’s method cost 182.764 secs. Therefore the timing ratio of each iteration of L-BFGSB and Lloyd’s is 1.19 : 1. Compared to Lloyd’s method, the function value by L-BFGSB after 80 iterations is already the same as the result of Lloyd’s method after 1000 iterations. This means that L-BFGSB is about 10 times faster than Lloyd’s method in attaining the same accuracy. According to the Hexagon Theorem [97], most of the Voronoi cells are supposed to be hexagons in CVT. So we also show in Figure 5.5 the ratio of the number of hexagonal cells to the total number of cells, with respect to the number of iterations. (The blue cells in Figure 5.5 indicate non-hexagonal cells.) L-BFGSB is again much better than Lloyd’s method in this regard.

Example 5.4. *3D CVT: 2000 seed points are sampled from $[-1, 1]^3$ randomly and the maximum iteration number is set to 1000. Figure 5.6 shows the comparison of $F(\mathbf{X})$ with respect to the iteration number. After 890 iteration, L-BFGSB has met its stop criterion.*

Similarly to Example 5.3, the function value via L-BFGSB after 80 iterations is already as small as that of Lloyd’s method after 1000 iterations. 890 iterations of L-BFGSB cost 851.2 secs and 1000 iterations of Lloyd’s method costs 900.10 secs. The timing ratio of each iteration of L-BFGSB and Lloyd’s is 1.06 : 1. Hence, L-BFGSB is again about 10 times faster than Lloyd’s method in attaining the same accuracy.

5.4 CVT on Polyhedral Surfaces

5.4.1 Constrained and restricted CVT

The concept of CVT has been extended to manifolds, with several possible applications, such as surface remeshing. Defining CVT in the geodesic metric [77, 80, 105] is a very natural idea but the computation cost of geodesic CVT is relatively too high. Du et al. proposed *Constrained Centroidal Voronoi Diagram*(CCVT) [49] that still uses Euclidean metric but constrains the seed point to lie on the surface. We follow their definition in our work. We denote a given smooth surface by $\Omega \subset \mathbb{R}^3$, the seed points as $\{\mathbf{x}_i \in \Omega\}_{i=1}^n$

and $\forall i \neq j, \mathbf{x}_i \neq \mathbf{x}_j$. The Voronoi region of \mathbf{x}_k is defined as:

$$\Omega_k = \{z \in \Omega \mid \|z - \mathbf{x}_k\| < \|z - \mathbf{x}_j\|, \forall j \neq k\}.$$

Similarly we can define the valid domain for CCVT be Γ_c . The constrained centroid of Ω_k is defined as

$$\mathbf{c}_k = \min_{y \in \Omega} \int_{z \in \Omega_k} \rho(z) \|y - z\|^2 dz$$

which exists but may be not unique [49]. The CVT function is defined by $F(\mathbf{X}) = \sum_{i=1}^N \int_{z \in \Omega_i} \rho(z) \|z - \mathbf{x}_i\|^2 dz$. Similar to 2D and 3D CVT, the minimizer of $F(\mathbf{X})$ still has nice centroidal properties: \mathbf{x}_i is the constrained centroid of Ω_i . Du et al. proved that Lloyd's method works well for CCVT [49].

If seed points are not constrained on Ω , the corresponding Voronoi diagram is known as a 3D VD restricted on Ω . Denote each Voronoi region in 3D by $Vor(i)$, the restricted region Ω_i is the intersection of $Vor(i)$ with Ω : $\Omega_i = Vor(i) \cap \Omega$. We call the centroidal Voronoi diagram under this restriction as an RCVT (Restricted CVT). It is easy to see that CCVT is a special type of RCVT.

5.4.2 C^2 smoothness

Similarly to 2D CVT, the fast computation of CCVT and RCVT can benefit from the smoothness of the CVT function. To have a well-defined problem, we assume that each Voronoi region Ω_i is singly connected and the shape of Ω is a C^2 surface which has no planar region. This condition avoids that this region coincides with any bisector plane of two seeds, which would make F no-longer C^2 (see Section 5.2.2). If Ω is a mesh surface, we suppose that the facets are small enough with respect to the number of the seeds. When the seeds $\{\mathbf{x}_i\}_{i=1}^n$ move, the adjacency relationship of all the Voronoi regions might change, which in turn causes the number of the sides of some Voronoi regions to change. As in the Euclidian case, $F(\mathbf{X})$ is defined in a piecewise manner – its expression takes different forms for combinatorially different Voronoi tessellations, and the change in this expression occurs when there is a structural “jump” of VD. The continuity analysis is similar to 2D-CVT proof detailed in Section 5.5, so we just point out the connection briefly as follows. In the singular case, we introduce a slight perturbation and compute the difference of CCVT or RCVT functions defined by different VDs. Similarly to 2D CVT, we consider the difference between two Voronoi diagrams CF_1 and CF_2 , the summation of the mass of the additional oriented curved regions U_1 and U_2 taken over by CF_1, CF_2 in the 3-D space respectively is 0, thus the mass of $(U_1 \cap \Omega) \cup (U_2 \cap \Omega)$ is still 0. This result helps us to remove constant terms in the difference of $\int_{CF_1} \rho(z) \|z - \mathbf{p}_i - \delta \mathbf{p}_i\|^2 dz + \int_{CF_2} \rho(z) \|z - \mathbf{p}_i - \delta \mathbf{p}_i\|^2 dz$. Since the integration is computed on the

surface, the difference is about $O(h)^3$. With the similar argument of Theorem 5.2, we have the following theorem:

Theorem 5.5. *CCVT and RCVT are twice-differentiable in Γ_c where Ω is a convex shape.*

Note: if Ω is nonconvex, its boundary can also induce C^1 smoothness.

5.4.3 Implementation

There are many works in remeshing and mesh segmentation via CVT [4, 49, 144, 145]. Their algorithms are based on Lloyd's method which updates the seed point with the centroid of its Voronoi region iteratively. Based on our smoothness analysis, we are able to integrate quasi-Newton method (L-BFGSB) to speed up the computation.

Voronoi diagram computation

The most costly part of CVT computation is to determine the Voronoi region for each seed. Face clustering by the flood-fill algorithm [36, 144, 145] has proven to be an extremely fast method in approximating VD. Valette et al. employed face clustering with approximated RCVT energy in their Lloyd algorithm [144]. Each VD is singly connected and bounded by polygonal faces. However, their algorithm can terminate at a sub-optimal solution after several iterations since the RCVT function actually is defined on a discrete space because of their use of face clustering. In contrast, we treat the triangular mesh as a continuous space, i.e, our optimization domain is a piecewise continuous submanifold. In our implementation the boundary of VDs can cross the faces of the mesh such that CVT energy can vary smoothly. Here we compare our result with Valette's algorithm using the accurate energy function. To allow a fair comparison, and eliminate mesh dependence that their algorithm suffers from, we choose a geodesic dome as our model. Valette's face based method stops after 10 iterations, In this setting the Lloyd method with face splitting stops after 100 iterations ($\|\delta_{\mathbf{x}}\| < 1e - 5$). From Figure 5.7, we can see that face-splitting method produces a better result.

Our VD computation consists of the following steps:

- 1. Initialization :** We select n distinct points randomly from the input triangular mesh. They serve as the initial seeds. Each seed is assigned a seed ID.
- 2. Clustering vertices and faces:** We first build a kd-tree from the seed points which is used for fast distance computation. For each vertex of the input mesh, its clustering ID is the ID of the nearest seed point from it. For each face, if the clustering IDs of its

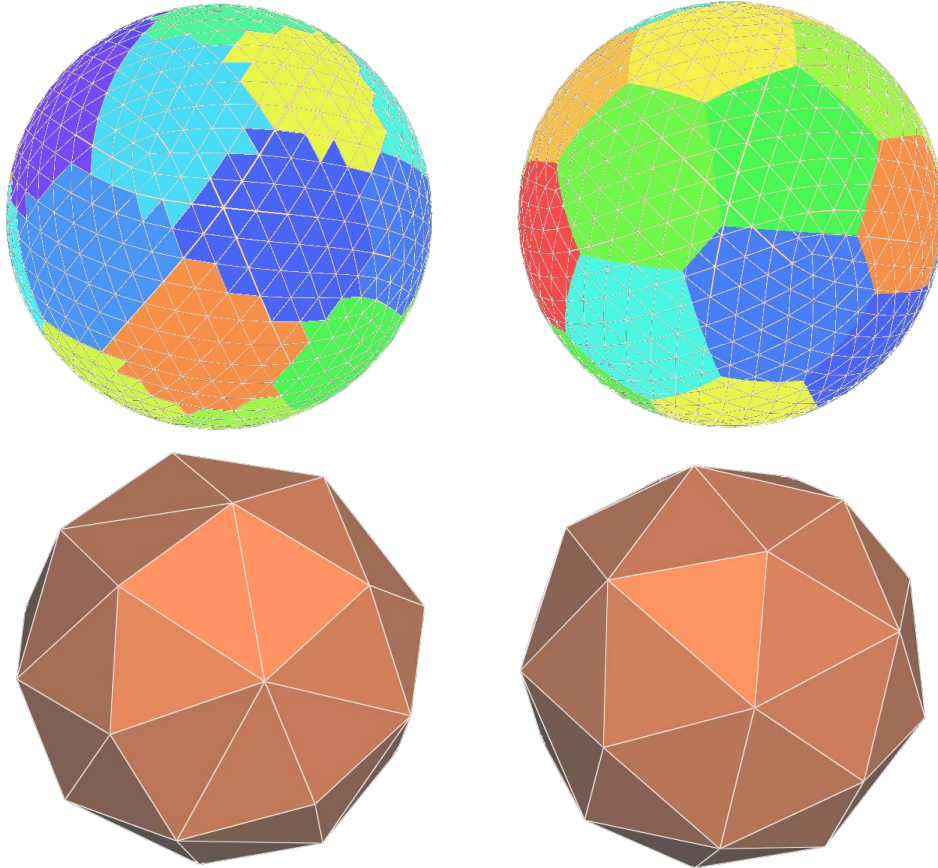


FIGURE 5.7: Voronoi diagrams and their dual. Left: face clustering; Right: face splitting. Face clustering produces a 7-sided polygonal VD. Our result is more regular.

three vertices are the same, we assign this same clustering ID to this face; otherwise, the face is labeled as a “split” face. For short, we denote the corresponding seeds of vertex and face as s_v and s_f .

3. 3D Voronoi Computation: We choose 26 points from a twice-enlarged bounding box B of the input mesh as the boundary points (from the corners, the middle points of edges, the center of each faces of B). These points with the seeds serve as input for 3D Voronoi computation done with QHULL. Therefore we obtain the bounded Voronoi cells for each seed, we denote the Voronoi cell of s as $P(s)$, which is a convex polyhedron.

4. Start Voronoi Region tracing on the mesh: A “split” face $f = \Delta u_1 u_2 u_3$ and a seed s are picked, where s is one of the clustering seeds of f ’s vertices. Since at least one of faces of $P(s)$ intersects with f , we test the intersection for each face of $P(s)$ until one face M intersects f . The intersection point located on one edge of f is pushed into a stack.

5. Tracing by flooding: An entry is popped from the stack. It contains an intersection point p and the corresponding M and f . Starting from the intersected point, we trace the

intersecting line of M and f . There are only two cases: (1) M intersects with another edge e of f at q ; (2) one edge E of M passes through f at r . For the first case, we continue the tracing from q along the adjacent face of the input mesh on the other side of e . For the second case, we store r and push the adjacent unprocessed faces of all the Voronoi cells around E into the stack. The current M and f are labeled “processed”. [Note: in (5) each intersection point with the corresponding f and s is stored in the stack]. The above process is executed until the stack is empty. Then we check whether each “split” face is “processed”; if not, we process it by goto (4).

6. Voronoi Region construction by splitting: For each “split” face f , we can obtain the computed intersection points and the associated seeds on it from steps (4) and (5). For each associated s , its corresponding intersection points on f and the vertices of f are actually the boundary vertices of a convex part of f . We compute the convex hull of these points and vertices of f which belongs to s to get the corresponding sub-region for s .

Gradient computation

The quasi-Newton method involves gradient computation. The gradient of RCVT is same as that in 2D and 3D CVT.

$$\frac{\partial F}{\partial \mathbf{x}_i} = 2m_i (\mathbf{x}_i - \mathbf{c}_i) \quad (5.5)$$

where \mathbf{c}_i is the centroid of the Voronoi region Ω_i of \mathbf{x}_i .

For CCVT, since it is also RCVT but the seed points are constrained on surface, its gradient has the following form:

$$\left. \frac{\partial F}{\partial \mathbf{x}_i} \right|_{\Omega} = \frac{\partial F}{\partial \mathbf{x}_i} - \left(\frac{\partial F}{\partial \mathbf{x}_i} \cdot \mathbf{N}(\mathbf{x}_i) \right) \cdot \mathbf{N}(\mathbf{x}_i) \quad (5.6)$$

where $\mathbf{N}(\mathbf{x}_i)$ is the normal of the surface Ω at \mathbf{x}_i . In our implementation, the updated \mathbf{x}_i is always projected to the surface according to the property of the constrained centroid.

5.4.4 Experiments

We compare our method with Lloyd’s method in the 3D models (David in Figure. 5.8, Homer in Figure. 5.9). It shows our L-BFGSB algorithm is much better than Lloyd’s method like we did in 2D and 3D cases.

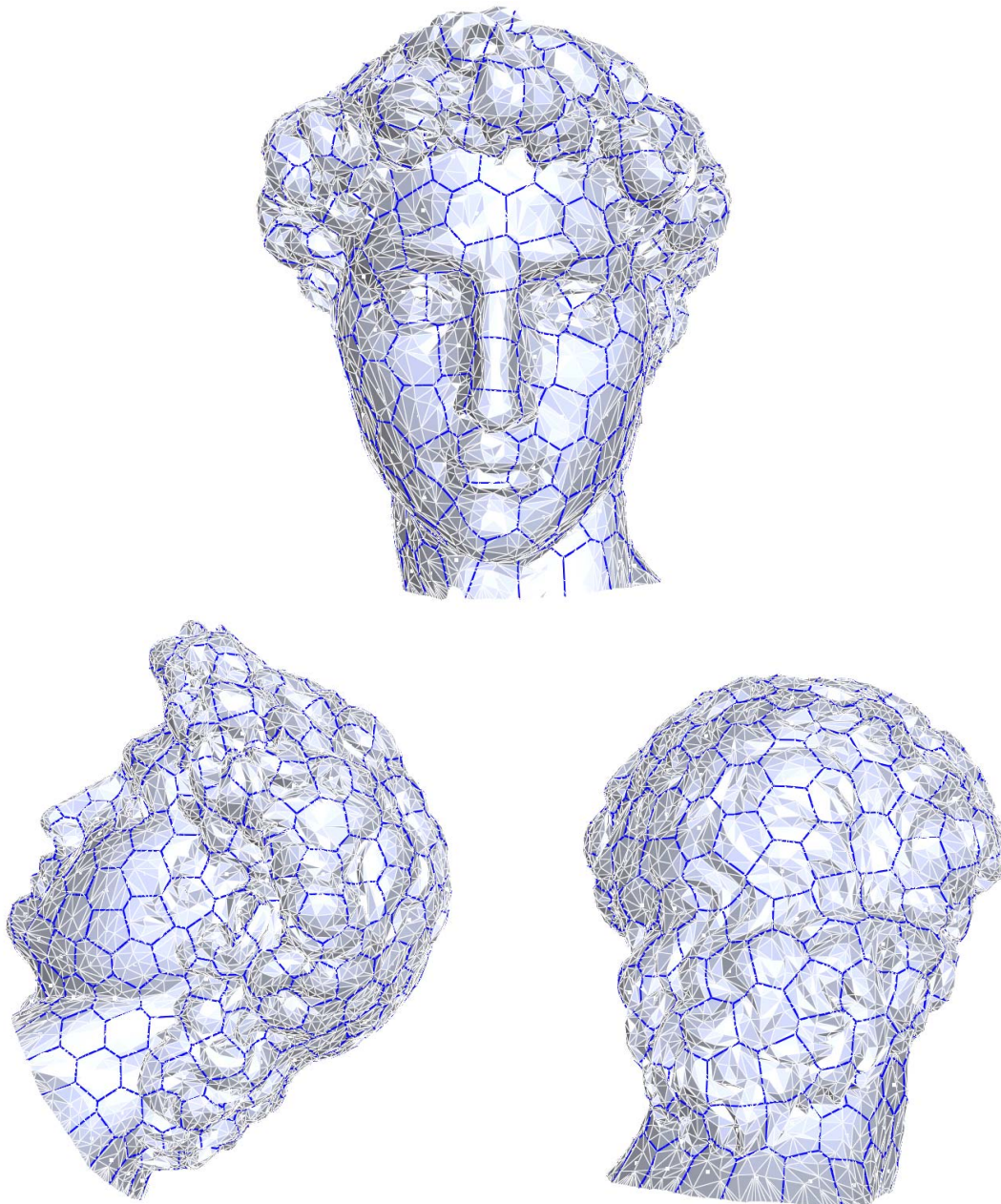


FIGURE 5.8: David model (Faces: 15836, vertices: 8014, seeds: 400). Our result after 80 iterations by L-BFGSB (200 seconds).

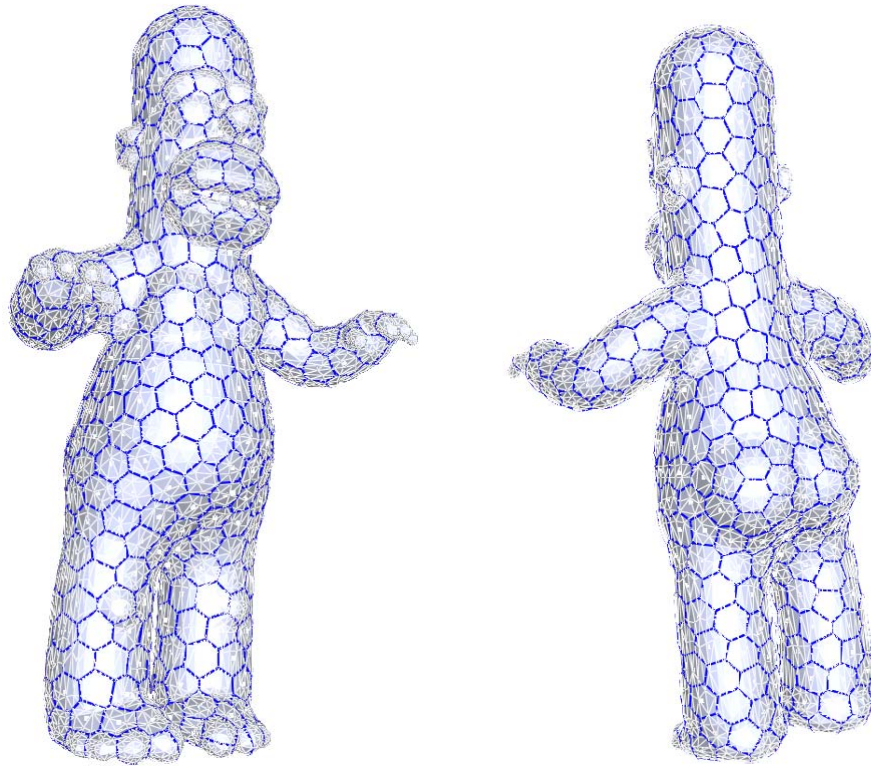
5.5 Appendix: Proof of Theorem 5.2

Let $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n$ be an *ordered set* of n seeds in a connected compact region $\Omega \subset \mathbb{R}^2$.

The Voronoi region Ω_i of \mathbf{x}_i is defined as

$$\Omega_i = \{z \in \Omega \mid \|z - \mathbf{x}_i\| < \|z - \mathbf{x}_j\|, \forall j \neq i\}.$$

The Voronoi regions Ω_i of all the seeds form the Voronoi diagram (VD) of \mathbf{X} . We define $F_i(\mathbf{X}) = \int_{z \in \Omega_i} \rho(z) \|z - \mathbf{x}_i\|^2 dz$ for each seed \mathbf{x}_i , where $\rho(\cdot) \geq 0$ is a density function and



CVT Energy Comparison

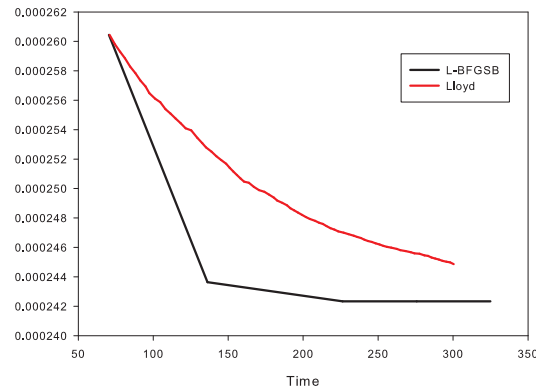


FIGURE 5.9: Homer model (Faces: 10202, vertices: 5103, seeds: 600). Our result after 80 iterations by L-BFGSB (220 seconds); Lower: $F(\mathbf{X})$ vs time.

$\sup_{z \in \Omega} \rho(z) \leq \gamma$ for some constant $\gamma > 0$. The CVT problem is to find the minimizer of the CVT function, or CVT energy,

$$F(\mathbf{X}) = \sum_{i=1}^n F_i(\mathbf{X}). \quad (5.7)$$

Here the variable $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ is a $2 \times n$ -dimensional point in $\Gamma := \Omega^n \subset \mathbb{R}^{2n}$. A natural assumption is that no two seed points are equal, i.e., $\mathbf{x}_i \neq \mathbf{x}_j, \forall i \neq j$. Hence, we shall consider the behavior of $F(\mathbf{X})$ in $\Gamma_c := \{\mathbf{X} \in \Gamma \mid \mathbf{x}_i \neq \mathbf{x}_j, \forall i \neq j\}$.

We first introduce the configuration space of the VD of \mathbf{X} for all $\mathbf{X} \in \Gamma_c$. An ordered

set $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n$ is *non-degenerate* if there do not exist four co-circular seeds in it. The set of all non-degenerate \mathbf{X} is denoted by $\Gamma_\mu \subsetneq \Gamma_c$. The VD of a non-degenerate set \mathbf{X} defines a distinct topological configuration, denoted as $CF(\mathbf{X})$, whose dual is the unique Delaunay triangulation of the seeds in \mathbf{X} . Hence, this Delaunay triangulation encodes the structure of configuration $CF(\mathbf{X})$, for instance as a set of integer triplets that correspond to the triangles of Delaunay triangulation. Two non-degenerate sets \mathbf{X}_1 and \mathbf{X}_2 are *equivalent* if $CF(\mathbf{X}_1) = CF(\mathbf{X}_2)$, and this relationship induces equivalence classes in $\Gamma_\mu \subset \mathbb{R}^{2n}$. Conversely, for a fixed non-degenerate \mathbf{X}' , any configuration C induces a ‘triangulation’ of the seeds in \mathbf{X}' , which is not necessarily the Delaunay triangulation of \mathbf{X}' and may not even be a valid triangulation of the seeds in \mathbf{X} in the sense of plane tessellation, since it merely records how the seeds in \mathbf{X}' are connected; it becomes the Delaunay triangulation of \mathbf{X} only when $C = CF(\mathbf{X}')$. The *configuration space*, denoted as \mathcal{C} , is the collection of the configurations of all order sets \mathbf{X} in Γ_μ . Define $G(C) = \{\mathbf{X} \in \Gamma_c | CF(\mathbf{X}) = C\}$ for any configuration C . Then the collection $\{G(C) | C \in \mathcal{C}\}$ induces a partition of Γ_μ and is a covering of Γ_c .

Since the VD of a degenerate set $\mathbf{X} \in \Gamma_c$ is a degenerate geometric realization of more than one configurations, we say that a degenerate point $\mathbf{X} \in \Gamma_c$ is associated with multiple configurations. This reflects the fact that the Delaunay triangulation of a set of co-circular seeds is not unique.

Every configuration $C_k \in \mathcal{C}$ has its associated expression $\Phi(\mathbf{X}; C_k)$, called the *evaluation function*, for defining the CVT evaluation, in the sense that the CVT function $F(\mathbf{X}) = \Phi(\mathbf{X}; C_k)$ if $CF(\mathbf{X}) = C_k$. Note that $\Phi(\mathbf{X}; C_k)$ is well-defined and infinitely smooth over Γ_c , even when $CF(\mathbf{X}) \neq C_k$. This can be understood as follows. When $CF(\mathbf{X}) = C_k$, the integration domain for each seed \mathbf{x}_i is a convex polygon, denoted as P_i , with their vertices defined by the circumcenters of certain triples of seeds. When $CF(\mathbf{X}) \neq C_k$, the circumcenters of the same triples of seeds will still be used to define the polygon P_i , which is not necessarily convex or even simple but will be treated as polygon with oriented sides. Then the evaluation function $\Phi(\mathbf{X}; C_k)$ is consistently defined by performing the integration in the oriented polygon P_i . Clearly, we just need to analyze the smoothness of $F(\mathbf{X})$ at a degenerate set \mathbf{X}_0 of seeds, which is a transition point between different configurations. Since \mathbf{X}_0 is degenerate, its VD is a geometric realization of multiple configurations associated with \mathbf{X}_0 , and we denote the set of these configurations by $H(\mathbf{X}_0)$. Since $F(\mathbf{X}_0) = \Phi(\mathbf{X}_0; C)$ for all $C \in H(\mathbf{X}_0)$, we need to show that the evaluation functions of any two configurations in $H(\mathbf{X}_0)$ have C^2 contact at \mathbf{X}_0 .

The degenerate set \mathbf{X}_0 contains several groups of seeds with each group containing at least four co-circular seeds whose Voronoi cells have a common point, and the seeds of different groups are on different circles. The Delaunay triangulation of the seeds of each

group is not unique, since any triangulation of these seeds is a Delaunay triangulation. The triangulations of different groups are independent of each other, as can be seen as follows. Two different groups can have at most two common seeds, since three different points determines a circle. If two groups have two common seeds, say u and v , then the two seeds must be adjacent seeds on the two circles of the two groups, for otherwise the circles would not be empty, violating the property of a Voronoi diagram. Therefore, the edge \overline{uv} belongs to any Delaunay triangulation of all the seeds in \mathbf{X}_0 . Hence, the triangulations of different groups of co-circular points do not interfere with each other.

Now it is clear that the configurations in $H(\mathbf{X}_0)$ differ from each other only in the way how the points in different groups of co-circular points are triangulated. Two configurations C_1 and C_2 are said to be *adjacent* if their dual triangulations differ by a single edge flip. It is well known that any two triangulations of a set of points in $2D$ can reach each other through a series of edge flips [23]. In our present specific setting, it is evident that any two configurations in $H(\mathbf{X}_0)$ are connected via a chain of adjacent configurations, as the consequence of the independence among the triangulations of different groups of co-circular points and the fact that any two triangulations of the same group of co-circular points are connected by a series of edge flips. Therefore we just need to show that the evaluation functions of any two adjacent configurations in $H(\mathbf{X}_0)$ have C^2 contact at \mathbf{X}_0 . Then the C^2 contact of the evaluation functions of any two configurations in $H(\mathbf{X}_0)$ will be implied by transitivity.

Lemma 5.6. *For any two adjacent configurations C_1 and C_2 in $H(\mathbf{X}_0)$, their evaluation functions $\Phi(\mathbf{X}; C_1)$ and $\Phi(\mathbf{X}; C_2)$ have C^2 contact at \mathbf{X}_0 .*

PROOF: Suppose that an arbitrary perturbation $\delta_{\mathbf{X}}$ of order $O(h)$, where $h > 0$ is arbitrarily small, is applied to \mathbf{X}_0 to yield $\mathbf{X}_1 = \mathbf{X}_0 + \delta_{\mathbf{X}}$. Consider the second order Taylor expansions of $\Phi(\mathbf{X}; C_1)$ and $\Phi(\mathbf{X}; C_2)$ at \mathbf{X}_0 : $\Phi(\mathbf{X}_1; C_1) = Q_1(\mathbf{x}_0; \delta_{\mathbf{X}}) + O(h^3)$ and $\Phi(\mathbf{X}_1; C_2) = Q_2(\mathbf{x}_0; \delta_{\mathbf{X}}) + O(h^3)$. In order to show that $\Phi(\mathbf{X}; C_1)$ and $\Phi(\mathbf{X}; C_2)$ have C^2 contact at \mathbf{X}_0 , it suffices to show that $\Phi(\mathbf{X}_1; C_2) - \Phi(\mathbf{X}_1; C_1) = O(h^3)$, since this implies that $Q_1(\mathbf{X}_0; \delta_{\mathbf{X}}) = Q_2(\mathbf{X}_0; \delta_{\mathbf{X}})$ for any perturbation $\delta_{\mathbf{X}}$ of order $O(h)$, i.e., all the derivatives of $\Phi(\mathbf{X}; C_1)$ and $\Phi(\mathbf{X}; C_2)$, up to the second order, agree with each other at \mathbf{X}_0 .

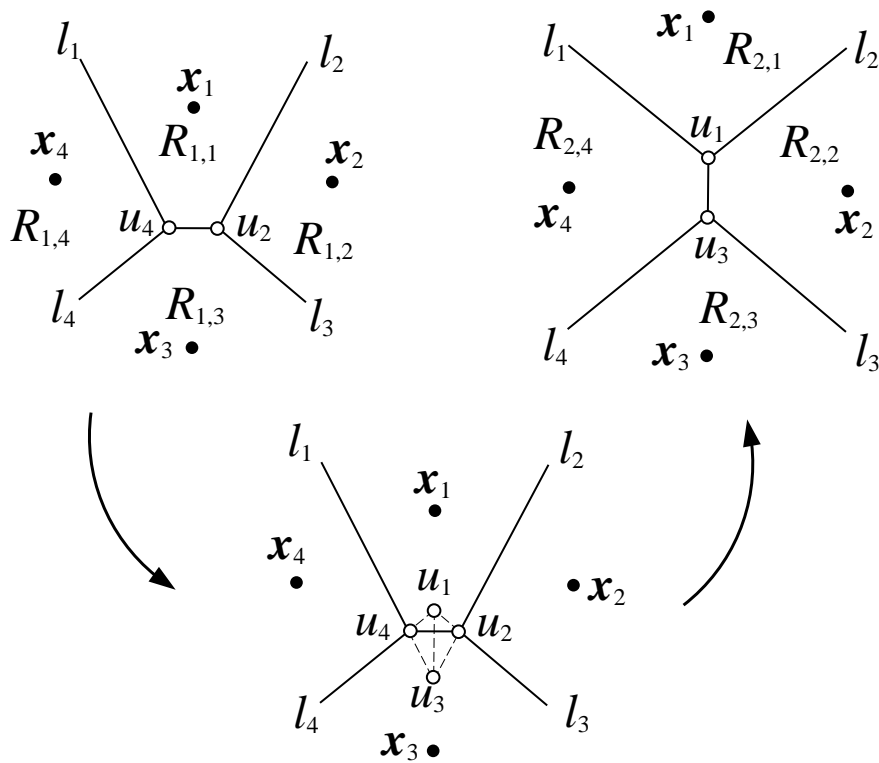
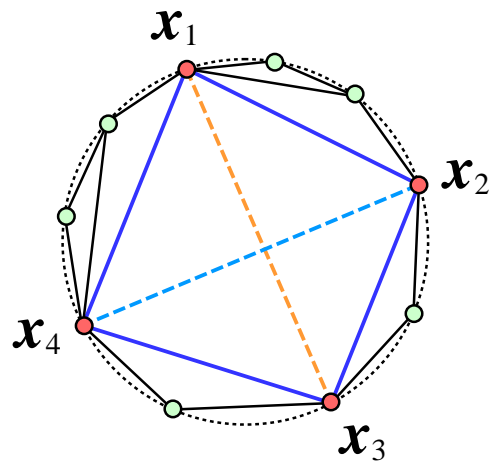


FIGURE 5.10: Geometric analysis of the CVT energy function F 's continuity during an edge flip.

Since $C_1, C_2 \in H(\mathbf{X}_0)$ are adjacent, there exist four co-circular seeds in $\{\mathbf{x}_j\}_{j=1}^k$ such that the triangulation of C_1 differs from that of C_2 by flipping the diagonals of the quadrilateral formed by these four co-circular seeds. Without loss of generality, we suppose that those are \mathbf{x}_i , $i = 1, 2, 3, 4$ (see the figure on the right). The local structures of the VD's of C_1 and C_2 as determined by the four points \mathbf{x}_i are shown in Figure 5.10). Denote the integration domain of the seed \mathbf{x}_i by $R_{1,i}$ and



$R_{2,i}$ respectively as they appear in C_1 and C_2 , $i = 1, 2, 3, 4$. Since C_1 and C_2 are adjacent, their structures are identical everywhere except for at the regions $R_{1,i}$ and $R_{2,i}$; hence, the difference between $\Phi(\mathbf{X}; C_1)$ and $\Phi(\mathbf{X}; C_2)$ can only be caused by the difference of their integration domains $R_{1,i}$ and $R_{2,i}$, $i = 1, 2, 3, 4$. Denote $g_i(z) = \rho(z)\|z - \mathbf{x}_i\|^2$, and define $\phi_{1,i} = \int_{z \in R_{1,i}} g_i(z) dz$ and $\phi_{2,i} = \int_{z \in R_{2,i}} g_i(z) dz$, $i = 1, 2, 3, 4$. Then $\Phi(\mathbf{X}; C_2) - \Phi(\mathbf{X}; C_1) = \sum_{i=1}^4 [\phi_{2,i} - \phi_{1,i}]$.

Let T_i denote the following oriented triangles: $T_1 : \Delta u_1 u_2 u_4$, $T_2 : \Delta u_1 u_3 u_2$, $T_3 : \Delta u_2 u_3 u_4$ and $T_4 : \Delta u_1 u_4 u_3$ (see Figure 5.10), where u_1, u_2, u_3, u_4 are the circumcenters of $\Delta \mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_4$, $\Delta \mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3$, $\Delta \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4$ and $\Delta \mathbf{x}_1 \mathbf{x}_3 \mathbf{x}_4$ respectively.

Clearly, $R_{2,i} = R_{1,i} \oplus T_i$, $i = 1, 2, 3, 4$, where \oplus denotes the union of oriented areas. Furthermore, we observe that $T \equiv \bigoplus_{i=1}^4 T_i = \emptyset$, i.e., the union of the four oriented triangles is the empty set. About the integrand $g_i(z)$ we have the following observation : at \mathbf{X}_0 , since the seeds \mathbf{x}_i , $i = 1, 2, 3, 4$, are co-circular, we have $\|\mathbf{x}_i - O\| = r$, where O is the center and r the radius of the circle \mathbf{S} containing the four seeds \mathbf{x}_i . Clearly, after the perturbation, at \mathbf{X}_1 we have $\|z - O\| = O(h)$ and, consequently, $g_i(z) = \rho(z)\|z - \mathbf{x}_i\|^2 = \rho(z)[r^2 + O(h)]$ for any $z \in T_i$, $i = 1, 2, 3, 4$. Finally, we note that the area of each triangle T_i is $\int_{z \in T_i} dz = O(h^2)$. From these observations it follows that

$$\begin{aligned}
\Phi(\mathbf{X}; C_2) - \Phi(\mathbf{X}; C_1) &= \sum_{i=1}^4 [\phi_{2,i} - \phi_{1,i}] \\
&= \sum_{i=1}^4 \left[\int_{z \in R_{2,i}} g_i(z) dz - \int_{z \in R_{1,i}} g_i(z) dz \right] \\
&= \sum_{i=1}^4 \int_{z \in T_i} g_i(z) dz = \sum_{i=1}^4 \int_{z \in T_i} \rho(z) [r^2 + O(h)] dz \\
&= r^2 \sum_{i=1}^4 \int_{z \in T_i} \rho(z) dz + \sum_{i=1}^4 \int_{z \in T_i} O(h) dz \\
&= r^2 \int_{z \in \bigoplus_{i=1}^4 T_i} \rho(z) dz + O(h^3) \\
&= r^2 \int_{z \in \emptyset} \rho(z) dz + O(h^3) = O(h^3).
\end{aligned}$$

So far we have shown the C^2 smooth of the CVT function F when a perturbation of the seeds do not affect or involve the boundary $\partial\Omega$ of the domain Ω . For the convex domain Ω , it is not difficult to prove that F is still C^2 when the boundary $\partial\Omega$ is involved in the boundary update of a Voronoi cell, following the same idea of the proof above. We will not repeat the detail here. \square

Hence, Theorem 5.2 is proved.

Chapter 6

Geometric Modeling with Conical Meshes and Developable Surfaces

6.1 Introduction

The original motivation for this research comes from architecture, where freeform shapes are becoming increasingly popular, but the actual construction poses new demands on the underlying geometry. Gehry Partners and Schlaich Bergermann and Partners [56] argue why freeform glass structures with planar quadrilateral facets are preferable over structures built from triangular facets or non-planar quads. The authors also show a few simple ways to construct quad meshes with planar faces. However, despite the huge amount of work on mesh processing and the interest in discrete differential geometry [40], we are not aware of a thorough investigation of this topic from the perspective of geometry processing.

The study of quad meshes with planar faces – called *PQ meshes* henceforth – will lead us to interesting geometric results, in particular to *conical meshes*, a discrete counterpart of principal curvature lines which have not been considered before. Algorithms which perturb a quad mesh into a PQ mesh can nicely be combined with subdivision. This makes subdivision a promising tool for architectural design and also provides a new and elegant approach to modeling and approximation with developable surfaces.

6.1.1 Previous work

Discrete differential geometry.

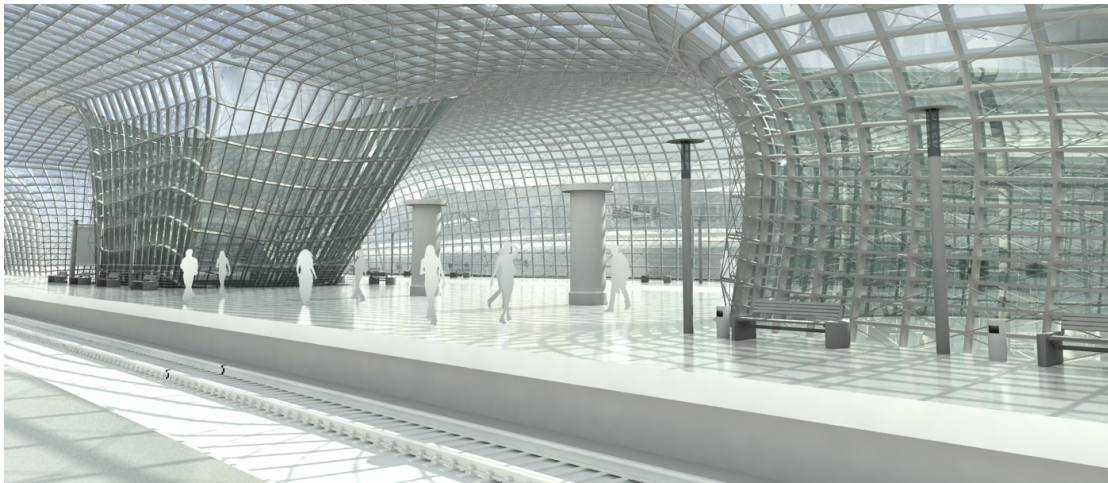
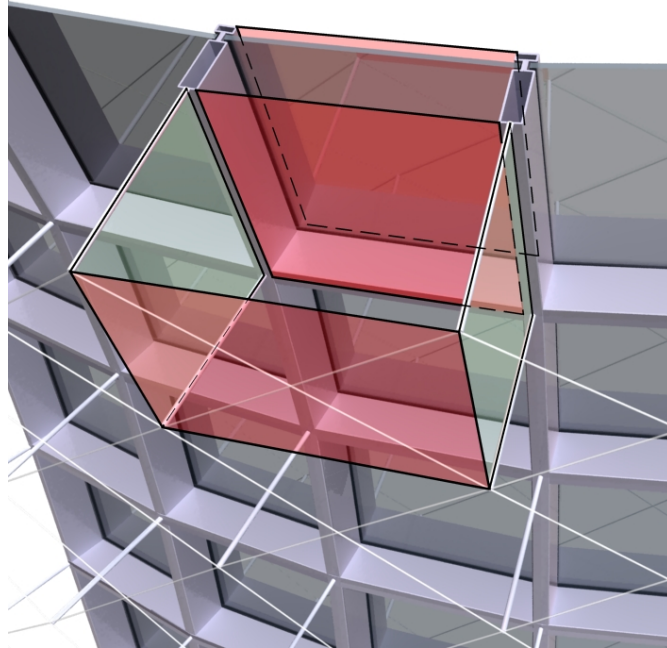


FIGURE 6.1: Conical meshes are planar quad meshes which discretize principal curvature lines, possess offset meshes at a constant distance as well as planar connecting elements supporting the offset meshes (upper). Therefore they are especially suited for architectural design with glass structures (lower). This student project of a railway station by B. Schneider was generated by a subdivision-type process (see also Figure. 6.14).

PQ meshes have first been systematically addressed by R. Sauer, as summarized in his monograph [126] on *difference geometry*, one of the precursors of discrete differential geometry [19, 40, 62, 108]. It has been observed that PQ meshes are a discrete counterpart of conjugate curve networks on surfaces. They appear in the mathematics literature under the name of *quadrilateral* meshes, which actually means quad meshes with the additional property that all quads are planar. The interesting case of *circular meshes* where all quads possess a circumcircle has been introduced in [93]. Circular meshes are discrete analogues of the network of principal curvature lines. Pointers to the literature on PQ meshes and circular meshes, especially to higher-dimensional generalizations, are given in [19] and [22]. Convergence of circular meshes towards the network of principal curvature lines is the topic of [21].

Quad meshes.

The computation of quad-dominant meshes from smoothed principal curvature lines has been presented in [3]. Although the faces of these meshes are not exactly planar, one should expect that they are at least approximately planar. Thus such meshes can serve as an input to algorithms presented below, which compute numerically precise PQ meshes and conical meshes by optimization. Variational shape approximation according to [36] aims at the optimal placement of a given number of planar faces, which, in general, are not quadrilaterals. Other recent contributions to quadrilateral remeshing (see e.g. [43, 92, 123]) do not try to achieve planarity of quads.

Developable surfaces.

An arrangement of n planar quads in a single row (see Figure. 6.2) is a discrete representation of a *developable surface*. In this way the study of PQ meshes is related to the computational geometry of developable surfaces. Recall a few facts from differential geometry [42, 115]: A developable surface Γ is the envelope of a one-parameter family of planes. Each of these planes touches the surface along a straight line, a so-called ruling. There are three main types: Either rulings are parallel (Γ is a cylinder surface), or they pass through a fixed point \mathbf{s} (Γ is a cone with vertex \mathbf{s}), or they are tangents of a space curve r (Γ is a tangent surface and r is its singular curve). Because developable surfaces can be mapped into the plane without distortion, they possess a variety of applications, for example, in sheet-metal and plate-metal based industries and architecture. Modeling with developable surfaces is a nontrivial task, which is only weakly included in current 3D modelers. Several ways of geometric design with developables have been proposed. One can use B-spline ruled surfaces and express developability via nonlinear constraints [9, 32]. Via duality such constraints can be avoided, at the cost of a less intuitive plane-based control structure [115]. There are also contributions based on constrained triangle meshes [55, 95, 150]. Singularities in crumpled sheets have also

received attention, see e.g. [29, 55]. Recently there has been interest in developable surfaces for mesh parameterization [72] and mesh segmentation [157].

Surfaces in architecture and aesthetic design.

Freeform geometries are becoming increasingly popular in architecture, thus demanding adapted modeling methods which take the actual construction and fabrication into consideration. The Smart Geometry group (<http://www.smartgeometry.com>) promotes research in this direction; a good overview of the state of the art may be found in [75]. Geometric modeling for aesthetic design and ‘optimal geometry’ are the topics of [136] and [139].

6.1.2 Contributions and overview

- We introduce *conical meshes* and demonstrate their superiority over other types of meshes for architectural design and other applications where planarity and exact offset property are demanded. The conical mesh is a new type of *principal meshes* and it possesses the property that offsetting the face planes by a constant distance yields a planar mesh of the same connectivity, which is again a conical mesh. This is a very useful property in layer composition constructions for architecture, where each layer has to be covered by planar panel elements and the geometry of the outermost layer should also be valid for the offsets which represent the layer composition (see Figures 6.1, 6.11, 6.12a, and 6.15).
- We propose the *PQ perturbation algorithm* for computing a PQ mesh from an input quadrilateral mesh. By combining PQ perturbation with a surface subdivision scheme we obtain a powerful tool for modeling not only conical meshes, but also circular meshes and general PQ meshes. When applied to PQ strips, it leads to an effective and elegant approach to modeling developable surfaces.

In Section. 6.2 we elaborate on the relation between PQ meshes to *conjugate curve networks* for understanding the variety of PQ meshes. Section. 6.2.2 discusses the *PQ perturbation algorithm*. In Section. 6.3 we combine subdivision and PQ perturbation to get a *hierarchical construction of PQ meshes*. In particular, we obtain *developable subdivision surfaces*. *Conical meshes* are introduced in Section. 6.4, and their main properties are derived. Section. 6.5 discusses how to approximate given data by a conical mesh via optimization of a quad mesh, possibly derived from robustly computed principal curves on an appropriate scale. We discuss our results in Section. 6.6 and conclude the chapter with some pointers to future research in Section. 6.7.

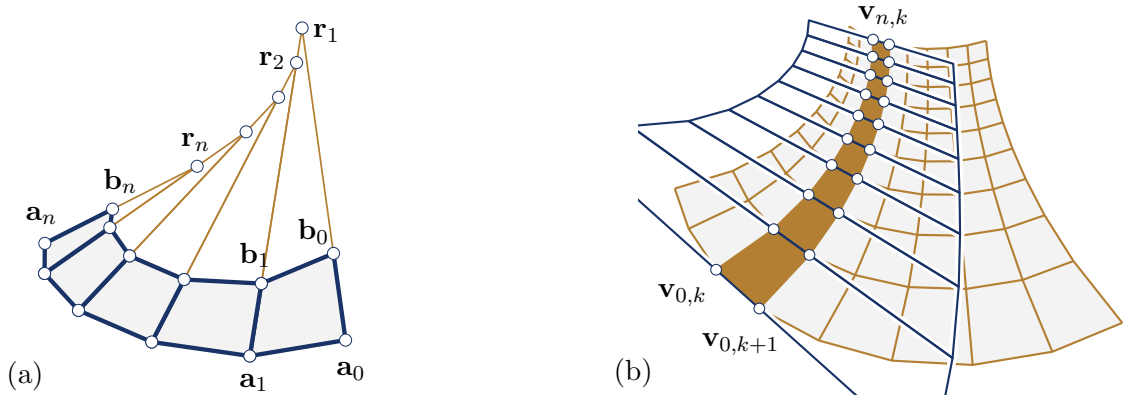


FIGURE 6.2: (a) PQ strip as discrete model for a developable surface. (b) Discrete developable tangent to PQ mesh along a row of faces.

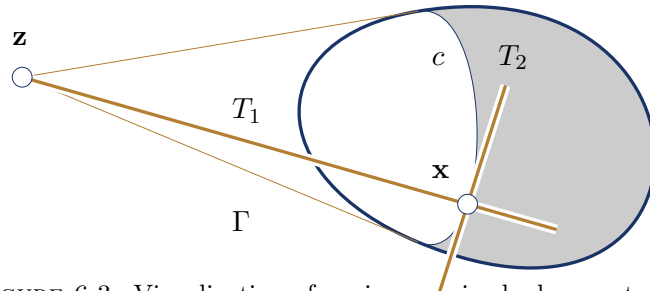


FIGURE 6.3: Visualization of conjugacy via shadow contours.

6.2 PQ Meshes and PQ Perturbation

Conjugate Curves.

Quad meshes with planar faces may be seen as a discrete version of so-called conjugate curve networks on a surface [126]. First we explain *conjugate surface tangents* at a point x of a surface Φ (see Figure. 6.3): Suppose that the straight line T_1 is tangent to the surface at x . Choose a light source z on T_1 . Then the line T_2 tangent to the shadow contour (contour generator) c at x is conjugate to T_1 . T_1 is contained in the conical surface Γ of surface tangents passing through the light source z . Here we could also use a parallel illumination, with z at infinity. An alternative definition of conjugate directions in terms of the second fundamental form of a surface is given by [42, p. 150].

The above is a special case of the following more general property: If Γ is the developable surface enveloped by the tangent planes along a curve $c \subset \Phi$, and T_1 is a ruling of Γ passing through the point $x \in c$, then the line T_2 tangent to the curve c at the point x is conjugate to T_1 . This relation turns out to be symmetric (see e.g. [115]). *Asymptotic* directions are self-conjugate. A *conjugate network of curves* consists of two one-parameter families A, B of curves which cover a given surface Φ such that for each point $p \in \Phi$ there is a unique curve of A and a unique curve of B which pass through x , and furthermore, the tangents of these two curves at x are conjugate. We may prescribe

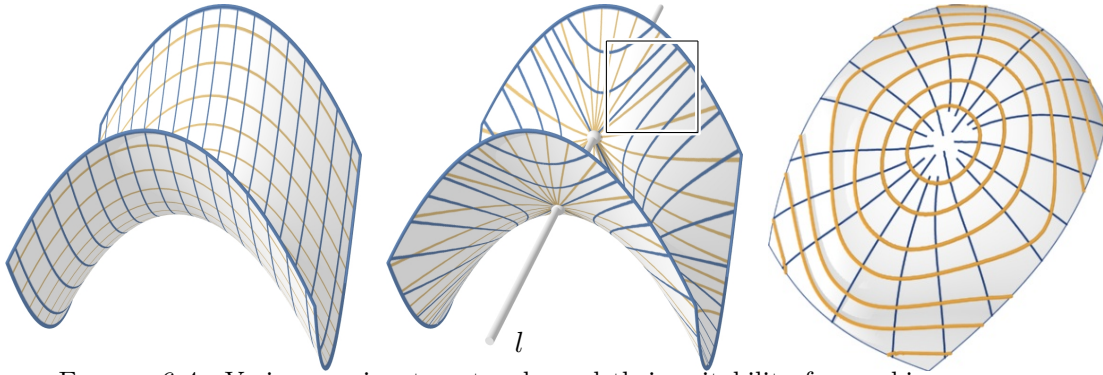


FIGURE 6.4: Various conjugate networks and their suitability for meshing purposes. Left: The network of generating curves in a translational surface Φ is conjugate. Center: For any surface Φ , the intersection curves (yellow) of Φ with planes through a fixed line l and the contour generators (blue) for viewpoints on l form a conjugate network. Right: Isophotes (yellow) and curves of steepest descent (blue). Such networks may be unsuitable for meshing even for simple surfaces, if its curves do not intersect transversely. This is caused by asymptotic directions (see frame).

family A and get family B by integration of the vector field of directions conjugate to the tangents of family A .

Examples of conjugate networks on surfaces are:

- The network of principal curvature lines is always conjugate.
- In a translational surface of the form $\mathbf{x}(u, v) = \mathbf{p}(u) + \mathbf{q}(v)$, generated by a translatory motion of a profile curve $\mathbf{p}(u)$ along a directrix curve $\mathbf{q}(v)$, or vice versa, the isoparameter lines form a conjugate network (Figure. 6.4, left).
- The movement of a viewpoint \mathbf{z} along some curve in space produces a family of *contour generators* “ $c(\mathbf{z})$ ” on a given surface Φ , where \mathbf{z} is interpreted as a light source. The curves conjugate to the $c(\mathbf{z})$ ’s are called *epipolar curves* and are found by integrating the field of light rays tangent to the surface Φ . These curves arise in 3D surface reconstruction from apparent contours in an image sequence [33]. Figure. 6.4 (center) shows the case where \mathbf{z} moves along a straight line l .
- The condition that the surface normals form a constant angle with the z -axis defines an *isophotic curve*. These isophotes are conjugate to the system of *curves of steepest descent* with respect to the z -axis (see Figure. 6.4, right, and e.g. [115]).

6.2.1 PQ meshes

Let us start with a *PQ strip*, which means a single row of planar quadrilateral faces. The two rows of vertices are denoted by $\mathbf{a}_0, \dots, \mathbf{a}_n$ and $\mathbf{b}_0, \dots, \mathbf{b}_n$ (see Figure. 6.2). It is obvious and well known that such a mesh is a discrete model of a developable surface (see e.g. [115, 126]). This surface is cylindrical, if all lines $\mathbf{a}_i\mathbf{b}_i$ are parallel. If the lines $\mathbf{a}_i\mathbf{b}_i$ pass through a fixed point \mathbf{s} , we obtain a model for a conical surface with vertex

s. Otherwise the PQ strip is a patch on the tangent surface of a polyline $\mathbf{r}_1, \dots, \mathbf{r}_n$, as illustrated by Figure. 6.2: consecutive lines $\mathbf{a}_i\mathbf{b}_i$ and $\mathbf{a}_{i+1}\mathbf{b}_{i+1}$ are co-planar and thus intersect in a point \mathbf{r}_{i+1} . It follows that both \mathbf{a}_i and \mathbf{b}_i are contained in the line $\mathbf{r}_i\mathbf{r}_{i+1}$. This property is the direct analogue of the well known fact that, in general, a developable surface is part of the tangent surface of a space curve. The lines $\mathbf{r}_i\mathbf{r}_{i+1}$ serve as the rulings of the discrete tangent surface, which carries the given PQ strip. The planar faces of the strip represent tangent planes of the developable surface.

Now we consider a general PQ mesh with vertices $\mathbf{v}_{i,j}$, $i = 0, \dots, n$, $j = 0, \dots, m$. For theoretical investigations we will always assume that interior mesh vertices have valence four; vertices with valence $\neq 4$ are like singularities in a curve network and require special treatment. In practice, meshes will not consist of quads only – n -gons with $n \neq 4$ likewise are treated as singularities.

Recall the property mentioned above which characterizes conjugate curve networks: the envelope of tangent planes along a curve of family A is a developable surface, whose rulings are tangent to curves of family B . We can easily see that the row and column polylines of a PQ mesh enjoy a discrete version of this property: Each row of faces $\mathbf{v}_{i,j}$ (we let $j = k, k + 1$) is a PQ strip, which represents a discrete developable surface tangent to the mesh (Figure. 6.2). The row of vertices $\mathbf{v}_{0,k}, \dots, \mathbf{v}_{n,k}$ can be seen as the polyline of tangency between the mesh and this developable surface. The rulings of the developable surface are spanned by the edges $\mathbf{v}_{i,k}, \mathbf{v}_{i,k+1}$ for $i = 1, \dots, n$. The same lines occur as tangents of the column polylines $\mathbf{v}_{i,0}, \dots, \mathbf{v}_{i,m}$. It follows that the system of row and column polylines are a *discrete conjugate network* of polylines. Moreover, a discrete developable surface tangent to a PQ mesh along a polyline is given by a row (or a column) of quad faces.

Consequently, *if a subdivision process, which preserves the PQ property, refines a PQ mesh and produces a curve network in the limit, then the limit is a conjugate curve network on a surface.*

6.2.2 PQ perturbation

Given a quad mesh with vertices $\mathbf{v}_{i,j}$, we want to minimally perturb the vertices into new positions such that the resulting mesh is a PQ mesh. One way to solve this problem is by a Sequential Quadratic Programming method (SQP, see e.g. [89]), which minimizes fairness and closeness functionals subject to the planarity condition. Another way is a penalty method which optimizes a linear combination of functionals which express planarity, fairness, and closeness to the original mesh, weighted in a way which ensures numerically exact planarity.

In order to express planarity of a quad face Q_{ij} , we consider the four angles $\phi_{i,j}^1, \dots, \phi_{i,j}^4$ enclosed by the edges of Q_{ij} , measured in the interval $[0, \pi]$. It is known that Q_{ij} is planar and convex if and only these angles sum up to 2π . We use the notation

$$c_{pq,i,j} := \phi_{i,j}^1 + \dots + \phi_{i,j}^4 - 2\pi = 0. \quad (6.1)$$

Below we need sums of the form $\sum_{i,j} \lambda_{pq,i,j} c_{pq,i,j}$, which we write as $\lambda_{pq}^T c_{pq}$, i.e., the inner product of the vectors $\lambda_{pq} = (\lambda_{pq,i,j})$ and $c_{pq} = (c_{pq,i,j})$.

For modeling developable surfaces it is important that the planarity criterion also works for a thin planar quad which converges to a straight line segment. Here, the constraints in (6.1) serve to maintain convexity and thereby avoid singularities, but they cannot express planarity in the limit (the angle sum will tend to 2π in any case, assuming convexity). Therefore we add another planarity term: Denote the unit vectors along the edges in quad $Q_{i,j}$ by $\mathbf{e}_{i,j} := (\mathbf{v}_{i,j+1} - \mathbf{v}_{i,j}) / \|\mathbf{v}_{i,j+1} - \mathbf{v}_{i,j}\|$, $\mathbf{e}_{i+1,j}$, $\mathbf{f}_{i,j} := (\mathbf{v}_{i+1,j} - \mathbf{v}_{i,j}) / \|\mathbf{v}_{i+1,j} - \mathbf{v}_{i,j}\|$, and $\mathbf{f}_{i,j+1}$. Then, using the four vertices in an equal way, the planarity of $Q_{i,j}$ is enforced by the following constraints,

$$\begin{aligned} c_{det,i,j}^1 &:= \det(\mathbf{e}_{i,j}, \mathbf{e}_{i+1,j}, \mathbf{f}_{i,j}) = 0, & c_{det,i,j}^2 &:= \det(\mathbf{e}_{i,j}, \mathbf{e}_{i+1,j}, \mathbf{f}_{i,j+1}) = 0, \\ c_{det,i,j}^3 &:= \det(\mathbf{e}_{i,j}, \mathbf{f}_{i,j}, \mathbf{f}_{i,j+1}) = 0, & c_{det,i,j}^4 &:= \det(\mathbf{e}_{i+1,j}, \mathbf{f}_{i,j}, \mathbf{f}_{i,j+1}) = 0. \end{aligned}$$

A linear combination of these constraints as used below is denoted by $\lambda_{det}^T c_{det}$. Note that these terms are included as effective planarity constraints only when computing PQ strips.

In addition, we introduce two energy terms to ensure that the resulting PQ mesh has a fair shape and stays close to the input mesh. For aesthetic design we use the fairness term f_{fair} , which includes simplified bending energies of the mesh's row and column polygons.

$$f_{fair} := \sum_{i,j} [(\mathbf{v}_{i+1,j} - 2\mathbf{v}_{i,j} + \mathbf{v}_{i-1,j})^2 + (\mathbf{v}_{i,j+1} - 2\mathbf{v}_{i,j} + \mathbf{v}_{i,j-1})^2].$$

At the boundary not all vertices required by the sum exist, so in addition we define that any undefined square is set to zero. For the PQ mesh to remain close to the surface Φ defined by the original mesh, we need to minimize the distances of the perturbed mesh vertices from the original mesh surface Φ by minimizing

$$f_{close} := \sum_{i,j} \|\mathbf{v}_{i,j} - \mathbf{y}_{i,j}\|^2,$$

where $\mathbf{y}_{i,j}$ is the closest point (i.e., footprint) on Φ to $\mathbf{v}_{i,j}$. We put the above terms together and define the Lagrangian function

$$f_{PQ} := w_1 f_{fair} + w_2 f_{close} + \lambda_{pq}^T c_{pq} + \lambda_{det}^T c_{det}. \quad (6.2)$$

Note that the term $\lambda_{det}^T c_{det}$ is needed only when computing a PQ strip. SQP minimizes the energy term $w_1 f_{fair} + w_2 f_{close}$ subject to the constraints $c_{pq} = 0$ and $c_{det} = 0$. That is, the minimizer gives a PQ mesh that has a fair shape and is close to the original surface Φ . The desired minimum is a stationary point of the Lagrangian f_{PQ} . Note that λ_{pq} and λ_{det} are determined automatically by the SQP method, while w_1 and w_2 are user specified constants to control relative weighting of fairness and geometric fidelity.

SQP uses a sequence of Newton-like iterations. In each round we compute the Hessians and gradients of the four terms which occur in the Lagrangian f_{PQ} of (6.2) to form a local quadratic approximation of f_{PQ} at the current point. Computation of the Hessians is straightforward, except for the squared distance term $\|\mathbf{v}_{i,j} - \mathbf{y}_{i,j}\|^2$ in f_{close} , which involves the footpoints $\mathbf{y}_{i,j}$ as dependent variables, since $\mathbf{v}_{i,j} - \mathbf{y}_{i,j}$ is always perpendicular to the tangent plane of Φ at $\mathbf{y}_{i,j}$. We use $[(\mathbf{v}_{i,j} - \mathbf{y}_{i,j}) \cdot \mathbf{n}_{i,j}]^2$ as a quadratic approximation of $\|\mathbf{v}_{i,j} - \mathbf{y}_{i,j}\|^2$. This approximation arises from Gauss-Newton minimization of the squared distance of $\mathbf{v}_{i,j}$ from Φ and has been successfully used for registration [30] and curve and surface approximation (see e.g. [15]).

Rewrite f_{PQ} in (6.2) in the form $f_{PQ}(x, \lambda) = f(x) - \lambda^T c(x)$, where x denotes the unknown vertex coordinates, $f = w_1 f_{fair} + w_2 f_{close}$ and $-\lambda^T c(x) = \lambda_{pq}^T c_{pq} + \lambda_{det}^T c_{det}$. Let J denote the Jacobian matrix of the constraints $c(x)$ and H denote the Hessian matrix of $f_{PQ}(x, \lambda)$ w.r.t. x ; (note that the contribution to H by the f_{close} term is a Gauss-Newton approximation). The update step $x \rightarrow x + h$ is solved from

$$\begin{bmatrix} H & -J^T \\ -J & 0 \end{bmatrix} \begin{bmatrix} h \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ c(x) \end{bmatrix}. \quad (6.3)$$

We use a soft line-search strategy [89] to determine the actual update step size αh , $0 < \alpha \leq 1$, to ensure stable convergence and sufficient descent — so x is updated by



FIGURE 6.5: PQ perturbation without a closeness term applied to a highly un-planar mesh consisting of only a few quads.

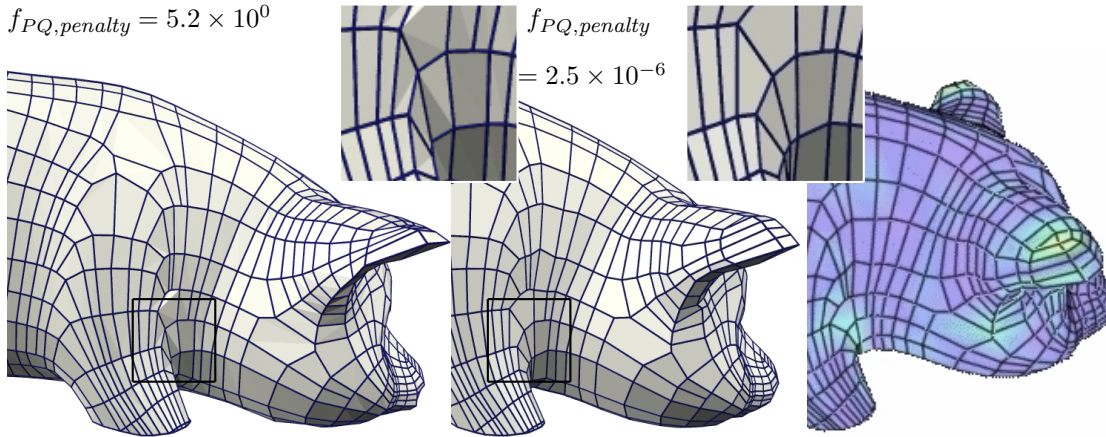


FIGURE 6.6: PQ perturbation acting on a quad-dominant mesh (left) extracted from principal curves (cf. Section 6.5). All faces, not only quads, are planarized (planarity is visualized via flat shading). The shape change noticeable at the ears is due to concentration of highly nonplanar quads in the original mesh. Higher geometric fidelity at the cost of fairness is easily possible. At right: color coded deviation from original mesh (max. 3% of object size).

$$x^* = x + \alpha h.$$

The above coefficient matrix is highly sparse and has the size $(3M + 5N) \times (3M + 5N)$ or $(3M + N) \times (3M + N)$ if not including the terms $\lambda_{det}^T c_{det}$, where M is the number of vertices and N the number of faces of the input mesh. We use the sparse matrix packages *TAUCS* and *UMFPACK*. Our implementation of SQP works efficiently for meshes of small or medium size (up to 1000 vertices). Our experience shows that for larger meshes it is more efficient to use a penalty method: We combine the angle constraints in (6.1) in the function

$$f_{angle} := \sum_{i,j} (\phi_{i,j}^1 + \dots + \phi_{i,j}^4 - 2\pi)^2, \quad (6.4)$$

and similarly the determinant terms in f_{det} . Then, we minimize the objective function

$$f_{PQ,penalty} = w_1 f_{fair} + w_2 f_{close} + \mu f_{det} + f_{angle}. \quad (6.5)$$

This is an unconstrained least squares problem which can be solved effectively by the Gauss-Newton method with L-M regularization [74]. We let $\mu = 1$ when considering a PQ strip, and $\mu = 0$ otherwise. For stability reasons, the coefficients w_1 and w_2 have higher values at the start of the iteration. Later, we let w_1 and w_2 tend to zero, so that planarity can be achieved with high accuracy. The following strategy works in practice: At the beginning, w_1, w_2 are chosen such that f_{angle} dominates in (6.5), and they are divided by 2 in every iteration. In case of small values of w_1 and w_2 , near-singular linear systems are solved via SVD.

In many cases the method exhibits much faster convergence than the SQP method. An

inherent problem of the penalty method are stability deficiencies near the optimum. Therefore in practice we use the penalty method to quickly derive an initial mesh near a local minimum, and then employ SQP. PQ perturbation works very well if the input quad mesh is reasonably close to a PQ mesh (cf. Figure. 6.6). Figure. 6.5 shows an input mesh which is far from planar, so PQ perturbations results in large deviations from the original. In order to planarize n -gons with $n > 4$, we use the fact that condition (6.1) generalizes to n -gons.

6.3 Subdividing Developables and PQ Meshes

To generate a PQ mesh from a coarse control mesh, we combine the PQ perturbation algorithm with a quad based subdivision algorithm like Doo-Sabin or Catmull-Clark in an alternating way: We subdivide a given PQ mesh once, and then apply PQ perturbation to make the resulting faces planar (see Figure. 6.7). These two steps are iterated to generate a hierarchical sequence of PQ meshes.

A single PQ strip can be subdivided by applying a curve subdivision rule like Chaikin's to its boundaries, and subsequent application of PQ perturbation in order to achieve face planarity. Alternating application of these two steps is a *subdivision algorithm which generates developable surfaces*. Because of our treatment of PQ perturbation as a black box it is in general not possible to write down the limit of this subdivision process explicitly. Nevertheless it is a much simpler design tool than developable B-spline surfaces, whose control points have to satisfy a set of nonlinear constraints.

As illustrated in Figure. 6.8, the relation of the input PQ strip to the final developable surface is very intuitive – certainly more so than the dual control structure in terms of tangent planes, which can be used to avoid nonlinear constraints (cf. [115]).

In the perturbation phase of the algorithm, the term f_{det} in (6.5) is important for maintaining planarity. The term f_{angle} discourages self-intersecting quads and thus acts against the common problem that the singular curve enters the designed patch. Finally, f_{fair} helps to prevent a zig-zag effect in adjacent quads.

6.4 Conical Meshes

Principal curvature lines form a special network of conjugate curves on a surface. Apart from umbilic points, where this network possesses singularities, it behaves nicely, since its curves intersect at right angles. This is not necessarily true for an arbitrary conjugate

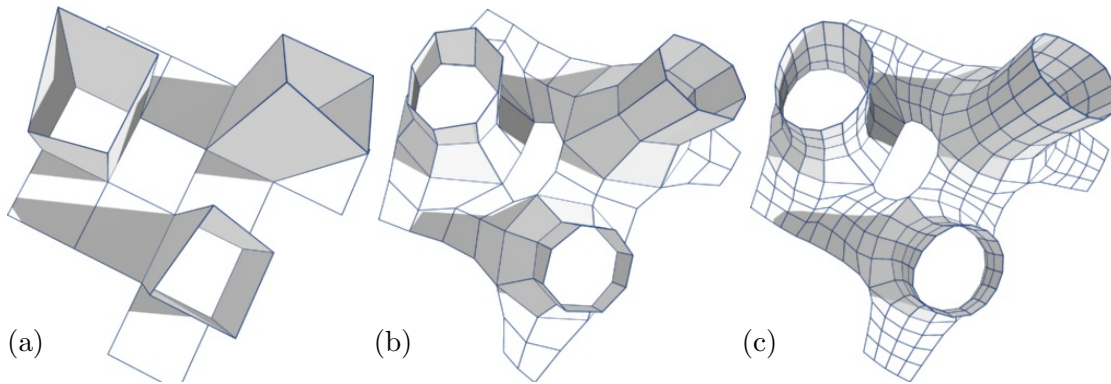


FIGURE 6.7: (a)–(c): Hierarchy of PQ meshes obtained by iterative application of Catmull-Clark subdivision and PQ perturbation.

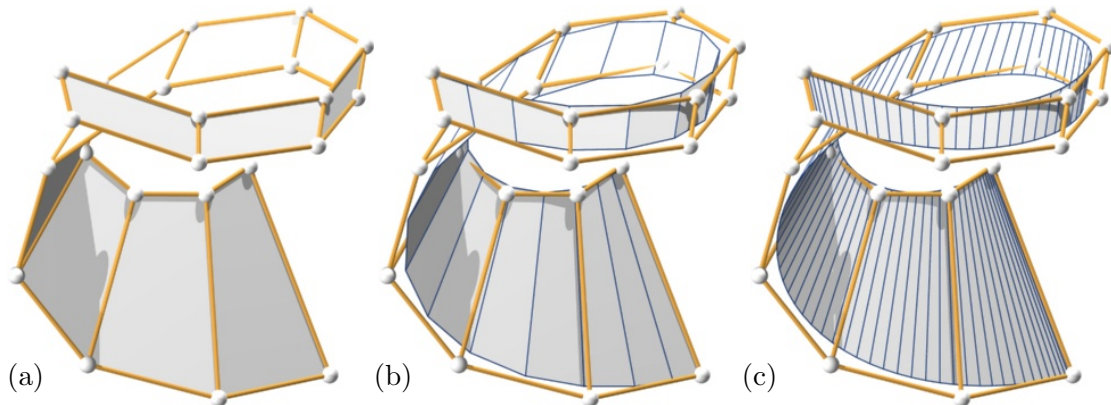


FIGURE 6.8: Developable subdivision surfaces generated with the perturbed cubic Lane-Riesenfeld algorithm; this nonlinear subdivision scheme keeps the planarity of quads and thus achieves developability of the limit. The control entity (a) is a piecewise-planar PQ strip. (b) and (c): 1 and 3 rounds of subdivision.

curve network; asymptotic (self-conjugate) directions give rise to degenerate situations that make such networks unsuitable for meshing purposes (Figure. 6.4).

A particular discretization of the network of principal curvature lines are the *circular meshes*, which are quad meshes whose quads are not only planar, but also have a circumcircle [19, 93]. It is however easy to extend our PQ perturbation algorithm to the computation of circular meshes (see Section 6.5 and Figure. 6.17). It turns out that another discrete analogue of the principal curvature lines – the conical meshes to be introduced in this section – have geometric properties essential for architectural design of freeform structures. For their computation via an augmented PQ perturbation algorithm, see Section. 6.5.

A vertex \mathbf{v} of a quad mesh is a *conical vertex* if all the four (oriented) face planes meeting at \mathbf{v} are tangent to a common (oriented) sphere. This is equivalent to saying that these oriented face planes are tangent to a common *oriented cone of revolution* Γ (see Figure. 6.10a). The axis G of Γ can be regarded as a discrete surface normal at that vertex.

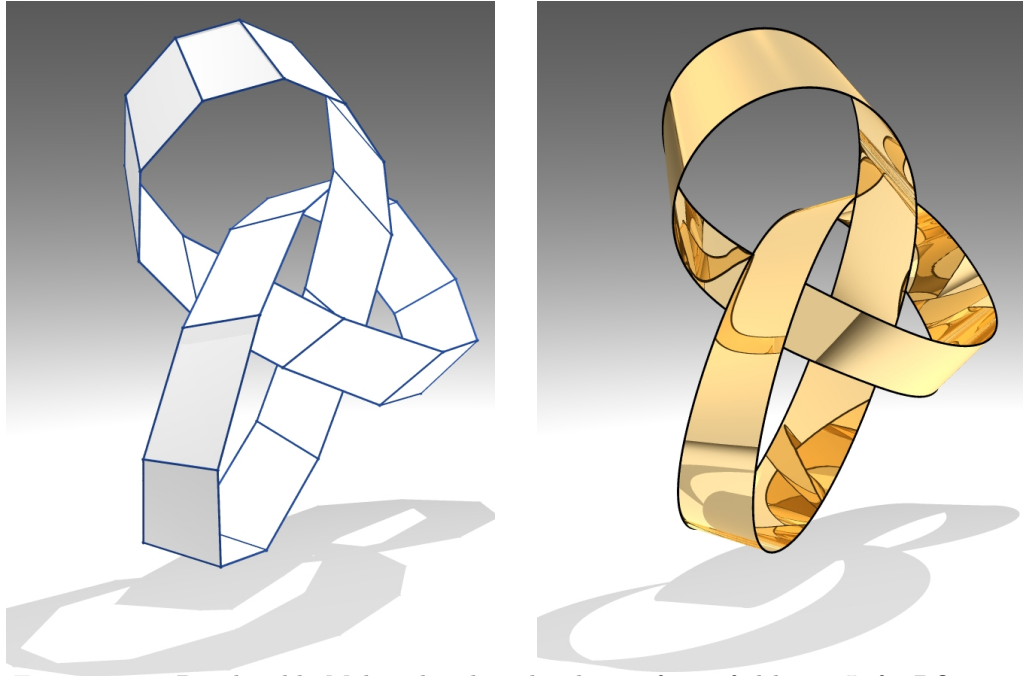


FIGURE 6.9: Developable Möbius band in the shape of a trefoil knot. Left: PQ strip as control structure. Right: Result of subdivision augmented by PQ perturbation. Numerical smoothness is C^2 , as seen from smooth reflection lines ($f_{PQ,penalty} = 2.9 \times 10^{-11}$).

We call a PQ mesh a *conical mesh* if all of its vertices of valence four are conical. For theoretical investigations, we consider only regular quadrilateral meshes whose vertices have valence 4, except for valence-2 or valence-3 vertices on the boundary. A conical mesh is in some sense dual to a circular mesh [114]. Instead of requiring the four vertices of a quad to be co-circular, we require that the four (oriented) faces incident with a mesh vertex be tangent to an (oriented) cone of revolution. We will see that conical meshes,

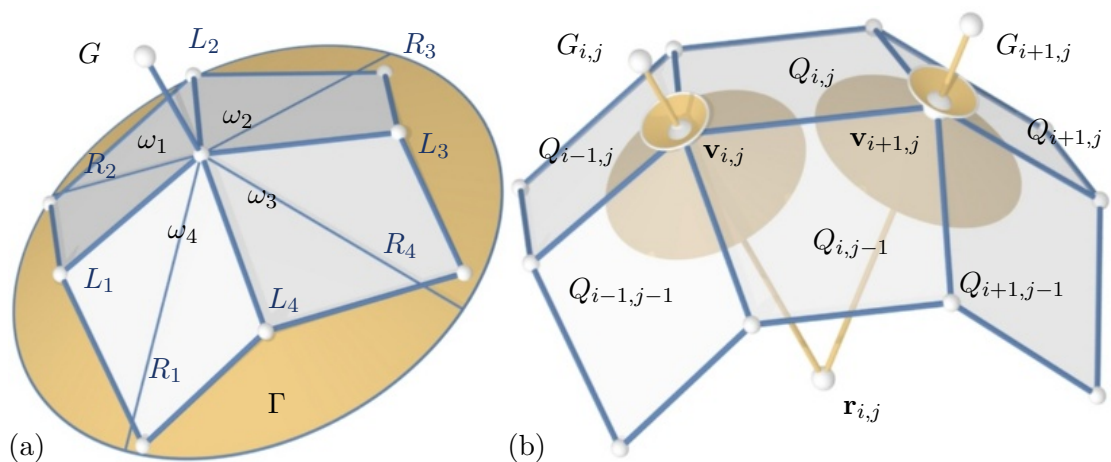


FIGURE 6.10: (a) Configuration of the faces of a conical mesh at a vertex. The faces touch the common cone Γ along rulings R_1, \dots, R_4 , and have interior angles $\omega_1, \dots, \omega_4$. (b) Faces of a conical mesh at two adjacent vertices $\mathbf{v}_{i,j}$ and $\mathbf{v}_{i,j+1}$, and the intersection point $\mathbf{r}_{i,j}$ of neighboring axes $G_{i,j}, G_{i,j+1}$.

like circular meshes, discretize the network of principal curvature lines.

There are exactly three types of conical mesh vertices, which can be characterized geometrically as follows. A small sphere S centered in a mesh vertex \mathbf{v} intersects the mesh in a simple 4-sided spherical polygon P . If the four vertices \mathbf{p}_i of P cannot be contained in the same hemisphere, \mathbf{v} is of the *hyperbolic* type. Otherwise (i.e., the four vertices \mathbf{p}_i are contained a hemisphere) \mathbf{v} is either of *elliptic type* (see Figure. 6.10a) or of *parabolic type*, depending on whether P is convex or not. These three types of mesh vertices are discrete analogues of hyperbolic points, elliptic points and parabolic points on a smooth surface.

An angle criterion for conical meshes.

There is a simple condition characterizing a conical mesh in terms of the interior angles of its quads. This characterization is also important for computing conical meshes (see Equation. (6.6) in Section 6.5).

Geometry Fact 1. A vertex of a quad mesh is a conical vertex if and only if the angle balance $\omega_1 + \omega_3 = \omega_2 + \omega_4$ is satisfied (see Figure. 6.10 for notation).

Here we assume that no two adjacent faces incident with a mesh vertex are co-planar, for otherwise the vertex is always conical. The (oriented) great circles that carry the edges of the spherical polygon P are tangent to a common (oriented) circle if and only if the vertex is conical. For elliptic vertices, Geometry Fact 1 follows from a result by A. J. Lexell which states that a convex spherical quadrilateral has an incircle if and only if the sums of opposite sides are equal. A proof of Geometry Fact 1 for all types of mesh vertices is given in Section 6.8. As an example we now use the elliptic vertex in Figure. 6.10a to illustrate why the angle balance holds in this case.

Consider the right circular cone Γ tangent to all the four faces incident with \mathbf{v} , the vertex of Γ . Suppose that the face plane Q_i touches Γ along the ruling R_i . Let L_i denote the intersection line of Q_i and Q_{i+1} . Denote $\alpha_i = \sphericalangle(L_i, R_i)$. Then, by symmetry, $\sphericalangle(L_i, R_{i+1}) = \sphericalangle(L_i, R_i) = \alpha_i$. Since $\omega_i = \alpha_i + \alpha_{i+1}$, we have $\omega_1 + \omega_3 = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = \omega_2 + \omega_4$.

Offsetting conical meshes.

Meshes with planar faces (including triangle meshes, cf. e.g. [76]) in general do not have the property that offsetting all faces by a fixed distance leads again to a mesh with the same connectivity, since planes meeting at a common point in general do not meet again at a common point after offsetting. Conical meshes, however, have this property — they possess conical quad meshes as offset meshes, as illustrated by Figures 6.11 and 6.12: The faces of a conical mesh incident with a vertex $\mathbf{v}_{i,j}$ are tangent to an oriented

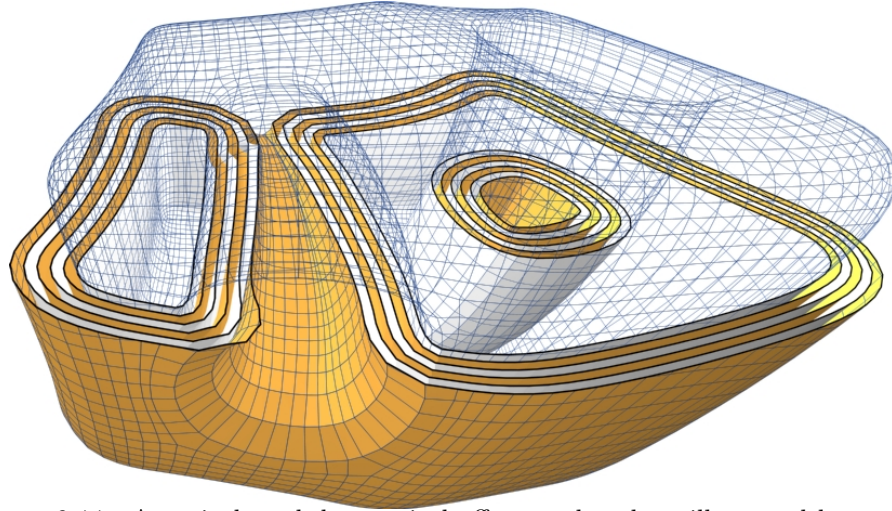


FIGURE 6.11: A conical mesh has conical offset meshes, here illustrated by a planar cut through a sequence of offsets.

cone with axis $G_{i,j}$. After offsetting, they are still tangent to a cone with the *same* axis. This behavior of the discrete surface normal $G_{i,j}$ is consistent with the behavior of the ordinary surface normal of a smooth surface under offsetting (which also does not change).

Remark: It is easy to show that any PQ mesh having the offsetting property is a conical mesh; that is, the offsetting property is a characterizing property of conical meshes. Offsetting planes by a fixed distance along their normal vector is a simple instance of a Laguerre transformation [28]. It is not difficult to see that general Laguerre transformations map conical meshes to conical meshes, whereas the property of a mesh being circular is preserved under Möbius transformations [19].

The normals of a conical mesh.

Starting from a planar mesh with quads $Q_{i,j}$, we construct a mesh in the unit sphere whose vertices \mathbf{n}_{ij} are the unit normal vectors of Q_{ij} . It is called the *spherical image* of the original PQ mesh. For conical meshes, the spherical image has special properties: As the four faces which meet in a vertex $\mathbf{v}_{k,l}$ are tangent to a common cone $\Gamma_{k,l}$, the normal vectors of these faces enclose the same angle with the cone's axis. Thus these four normal vectors lie in a circle contained in the unit sphere; the spherical center of this circle gives the unit direction vector of $G_{k,l}$. Thus *the spherical image of a conical mesh is a circular mesh*. This property is a discrete analogue to the well known fact that the spherical image of the network of principal curvature lines is an orthogonal curve network on the sphere.

Support structures of conical meshes.

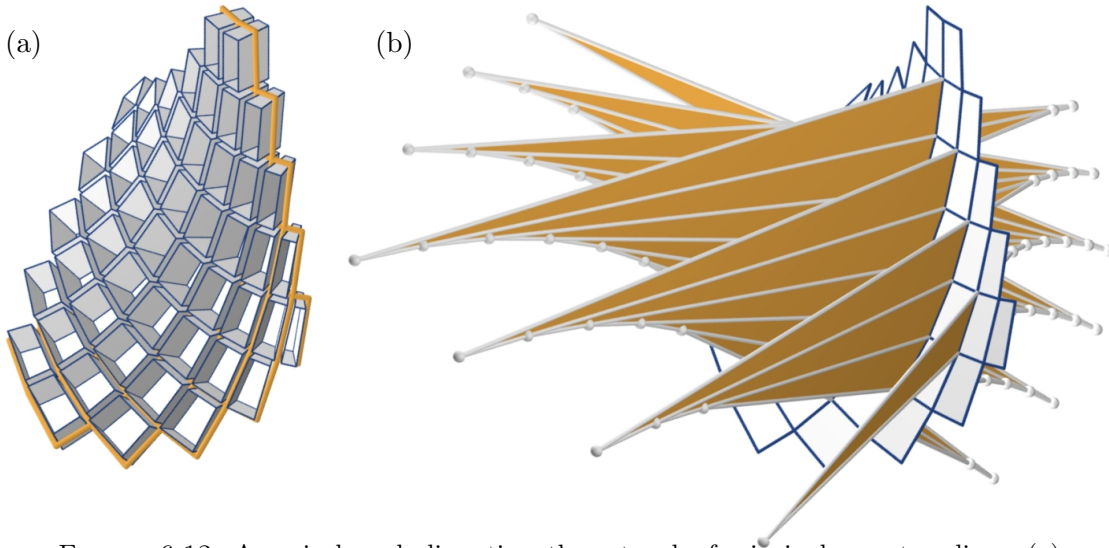


FIGURE 6.12: A conical mesh discretizes the network of principal curvature lines. (a) A conical mesh has conical offset meshes and a discretely orthogonal support structure connecting the offsets. (b) Cone axes of neighboring vertices intersect in the discrete principal curvature centers. Connecting these axes leads to discrete row and column developables orthogonal to the mesh.

Figure. 6.10b illustrates two neighboring vertices $\mathbf{v}_{i,j}$ and $\mathbf{v}_{i+1,j}$ of a conical mesh. There are two faces $Q_{i,j}$ and $Q_{i,j-1}$ containing both vertices. These two faces are tangent to both cones $\Gamma_{i,j}$ and $\Gamma_{i+1,j}$. It follows that their axes $G_{i,j}$ and $G_{i+1,j}$ lie in the bisector plane of the oriented faces $Q_{i,j}$ and $Q_{i,j-1}$. The important fact derived here is that neighboring axes (discrete surface normals) are co-planar, and they are contained in a plane orthogonal to the mesh in a discrete sense.

It follows that an edge of the mesh, the discrete normals at its endpoints, and the corresponding edge of any offset mesh, lie on a common plane. This property can be used to build ‘orthogonal’ support structures as shown in Figures 6.1, 6.12 and 6.15, which are important for the construction of freeform glass structures based on conical meshes. Co-planarity of axes $G_{i,j}$ and $G_{i,j+1}$ implies:

Geometry Fact 2. Successive discrete normals of a conical mesh along a row or column are co-planar and therefore form a discrete developable surface (see Figure. 6.12).

Recall that the surface normals of a smooth surface along a curve constitute a developable surface if and only if that curve is a principal curvature line. Fact No. 2 is a discrete analogue of this classical result, and shows the following important property:

Geometry Fact 3. If a subdivision process, which preserves the conical property, refines a conical mesh and in the limit produces a curve network on a smooth surface, then this limit curve network is the network of principal curvature lines.

Focal surfaces with conical meshes.

In a conical mesh, neighboring axes (discrete surface normals) in a row intersect, and so do neighboring axes in a column. These intersection points are discrete row and column *curvature centers*, which define, in general, a two-sheet discrete focal surface. It is easy to see that the two quad meshes defined by the row centers and the column centers are actually PQ meshes, and that singularities of discrete offsets occur at these two discrete focal sheets. This is analogous to the smooth case [110].

Remark: We might ask if there are meshes which are both circular and conical. The answer is in the affirmative, and it is not difficult to construct some. Interesting examples of conical meshes of constant cone opening angle, which are at the same time circular of constant circle radius, are derived from the discrete surfaces of constant negative Gaussian curvature of [156]. One of them is shown in Figure. 6.12. However, meshes with both properties may be too inflexible to be useful for modeling and approximation.

6.5 Computing Conical Meshes

We would like to approximate a surface Φ , which is given in any representation, by a conical (or circular) mesh. Since both types of meshes converge to principal curvature lines under refinement, it is a good choice to use a quad mesh extracted from principal curvature lines (e.g., the meshes from [3]) as input for an optimization algorithm which achieves the conical or circular property by perturbing the vertices as little as possible.

Robust computation of principal curves.

For computing principal curves, we employ a method different from previous approaches [34, 35]. In view of the desired average size of faces in a principal mesh, we find it appropriate to use as input *robust principal curves on a given scale r* (Figure. 6.13), which are computed as follows. The procedure analyzes neighborhoods of points \mathbf{p} of the given surface Φ . We choose a kernel radius r , which defines the scale on which we would like to work. The domain of space which, locally around \mathbf{p} , lies to one side of the surface is denoted by D . For each point $\mathbf{p} \in \Phi$ we perform a principal component analysis (PCA) of the set $N^r(\mathbf{p}) = B^r(\mathbf{p}) \cap D$, where $B^r(\mathbf{p})$ is a ball of radius r centered in \mathbf{p} . This means that we compute the barycenter \mathbf{s}^r of N^r and the eigenvectors $\mathbf{t}_1^r, \mathbf{t}_2^r, \mathbf{t}_3^r$ and corresponding eigenvalues $\lambda_1^r \leq \lambda_2^r \leq \lambda_3^r$ of the covariance matrix $J := \int_{N^r} (\mathbf{x} - \mathbf{s}^r) \cdot (\mathbf{x} - \mathbf{s}^r)^T d\mathbf{x}$. In the limit $r \rightarrow 0$, \mathbf{t}_3^r converges to the surface normal at \mathbf{p} , and $\mathbf{t}_1^r, \mathbf{t}_2^r$ converge to the principal directions. What we actually compute is a kind of average of these geometric characteristics over a small neighborhood of \mathbf{p} . Most importantly, directions $\mathbf{t}_1^r, \mathbf{t}_2^r$ are more robust against noise and minor perturbations than those of classical differential geometry or than those in [34], which are computed via PCA on the surface patch

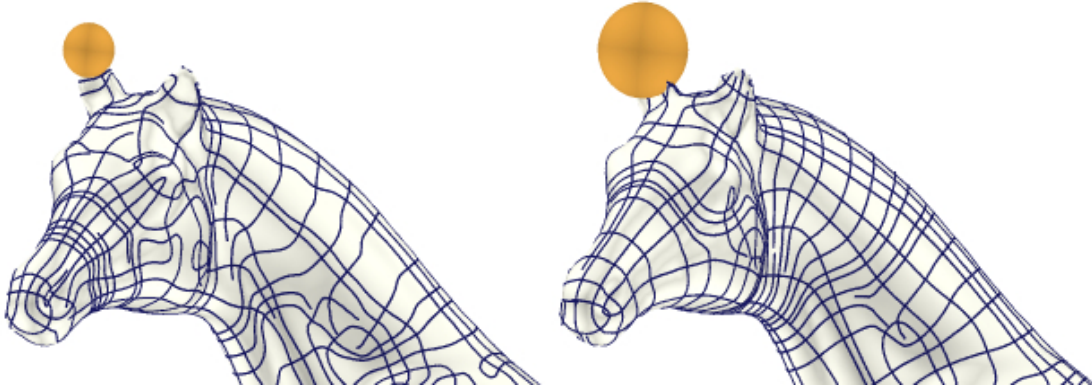


FIGURE 6.13: Principal curves computed with different kernel radii.

neighborhood $B^r(\mathbf{p}) \cap \Phi$. For proofs and details on efficient implementation we refer to [118].

The directions $\mathbf{t}_1^r, \mathbf{t}_2^r$ need not be tangent to the given surface at \mathbf{p} . However, we can still obtain meaningful principal directions at \mathbf{p} if we just project them onto the tangent plane at \mathbf{p} . The direction of this projection shall be given by the third eigenvector \mathbf{t}_3^r , which estimates the surface normal. The projected directions do not have to be orthogonal anymore, which is actually no loss and rather enhances stability when we now integrate these two vector fields to obtain *principal curves at the chosen scale r* (see Figure. 6.13). Our algorithm for vector field integration and quad mesh extraction is based on ideas in [3, 43, 92].

Conical and circular optimization.

It is not difficult to modify the PQ perturbation algorithm from Section. 6.2.2 so that it produces conical meshes. For perturbation into a conical mesh, we keep the constraints of (6.1) and, according to Geometry Fact No. 1, for each vertex add the constraint

$$\omega_{i,j}^1 + \omega_{i,j}^3 - \omega_{i,j}^2 - \omega_{i,j}^4 = 0. \quad (6.6)$$

The perturbation algorithm for computing a circular mesh is similar. A quad is planar, convex, and has a circumcircle, if and only if the four angles enclosed by its four edges have the property

$$\phi_{i,j}^1 + \phi_{i,j}^3 - \pi = 0, \quad \phi_{i,j}^2 + \phi_{i,j}^4 - \pi = 0. \quad (6.7)$$

We therefore replace the planarity constraint $\phi_{i,j}^1 + \dots + \phi_{i,j}^4 - 2\pi = 0$ in Equation. (6.1) by the two constraints in (6.7). The modifications of the penalty method proceed along the same lines. We do not enforce the conical condition for vertices of valence greater than four, as in architectural applications such vertices are expected to get special treatment anyway. We could however easily make such a vertex conical by imposing the conical condition for any four faces adjacent to the vertex. Similarly, for circular meshes the

circular condition is not enforced for n -gons with $n > 4$, except for artificial 5-gons which arise from quads at T junctions, where the original quad's vertices are made cocircular.

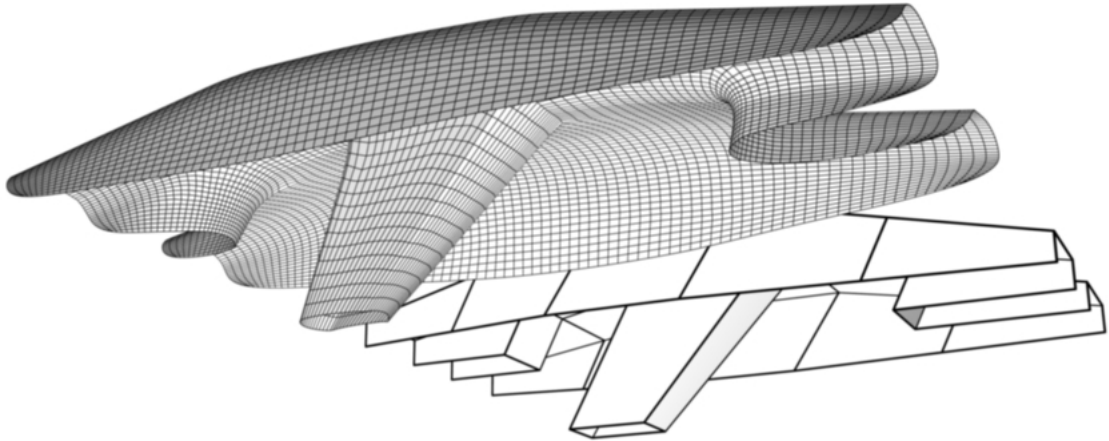


FIGURE 6.14: The conical mesh in front was obtained by a combination of Catmull-Clark subdivision and conical optimization from the control mesh behind. This conical mesh is the basis for the glass structure in Figure. 6.1.

6.6 Results and Discussion

Developable surfaces. Our experiments show that the proposed subdivision approach to developable surface modeling is a powerful new tool (see Figures 6.9 and 6.16). The elimination of the singular curve from the actually designed patch is simplified by the multiscale approach inherent to subdivision: Planarization results in convex quads and thus eliminates singularities from the designed patch at each subdivision level (Figure. 6.16). This multiscale elimination of the singular curve appears to be more efficient than the methods known in the literature [115].

The PQ perturbation method described in Section 6.2.2 makes use of a reference surface. If only a coarse PQ strip is available as a control structure for a smooth developable, such a reference surface may be generated by applying the *unperturbed* subdivision rule to the strip. Numerical evidence for the C^2 smoothness of the perturbed cubic Lane-Riesenfeld rule is furnished by the apparent smoothness of reflection lines in Figure. 6.9.

Conical and circular meshes.

The combination of subdivision and conical/circular perturbation produces high quality meshes suitable for aesthetic design (see Figures 6.1, 6.14, and 6.15). Without subdivision, it is essential that perturbation which aims at principal meshes (circular or conical) is applied to a mesh which is not too far away from a principal mesh. This is achieved by deriving a mesh from principal curves (cf. Figure. 6.17). Otherwise, we either get large deformations, or – if the surface is subject to further constraints such as fixed points

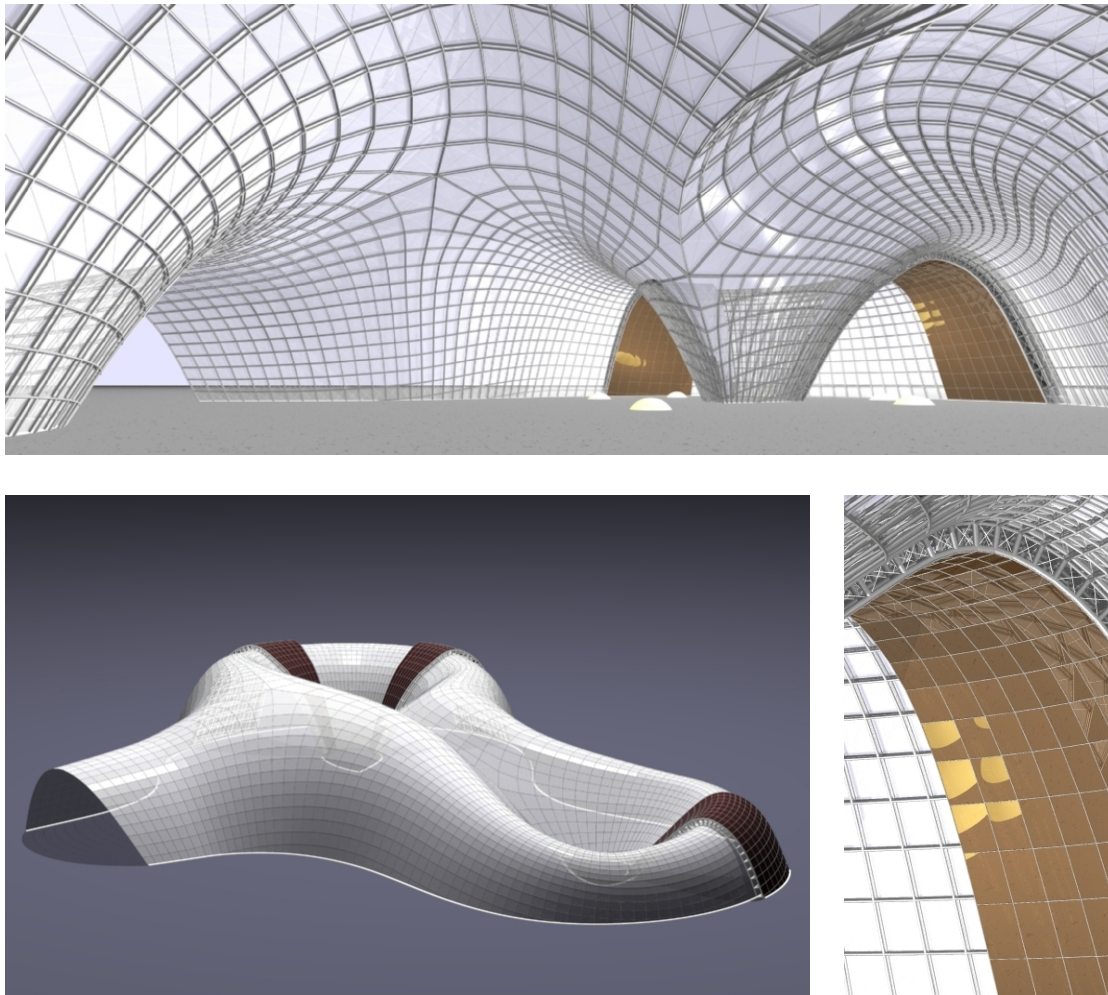


FIGURE 6.15: Design studies with conical meshes and their offset meshes produced by subdivision and conical perturbation. The figure shows a wide-angle perspective of the interior (upper), an exterior view (lower left), and an offset detail (lower right).

or closeness to a reference surface – we may obtain self-intersections, creases, and other undesirable effects.

Efficiency.

The performance of our non-optimized PQ perturbation code depends not only on the size of the input data, but also on the geometry and the nature of nearness constraints. To give a few numbers, on a 2 GHz PC we experienced computation times of 0.08 (penalty) and 0.75 seconds (sequential quadratic programming) for PQ perturbation applied to the trefoil knot with 336 faces in Figure. 6.9. Thus interactive modeling of developable surfaces is easily possible. Total mesh computation time for Figure. 6.15 was 13 seconds, of which 80% was for 4 iterations of conical perturbation (penalty) at the finest subdivision level with 5951 vertices. There is still room for improvement when processing large meshes. The number of iterations can be as high as 20–50 for the SQP method with a closeness term present. Surface approximation by conical and

circular meshes would certainly benefit from even better initial meshes, but this aspect of quad-dominant remeshing is not a focus of the current work.

Convergence. It should be mentioned that there are meshes where PQ perturbation fails because of topological obstructions. On the other hand we did not encounter problems with meshes based on principal curves, and PQ perturbation is capable of large deformations when that is necessary for achieving planarity.

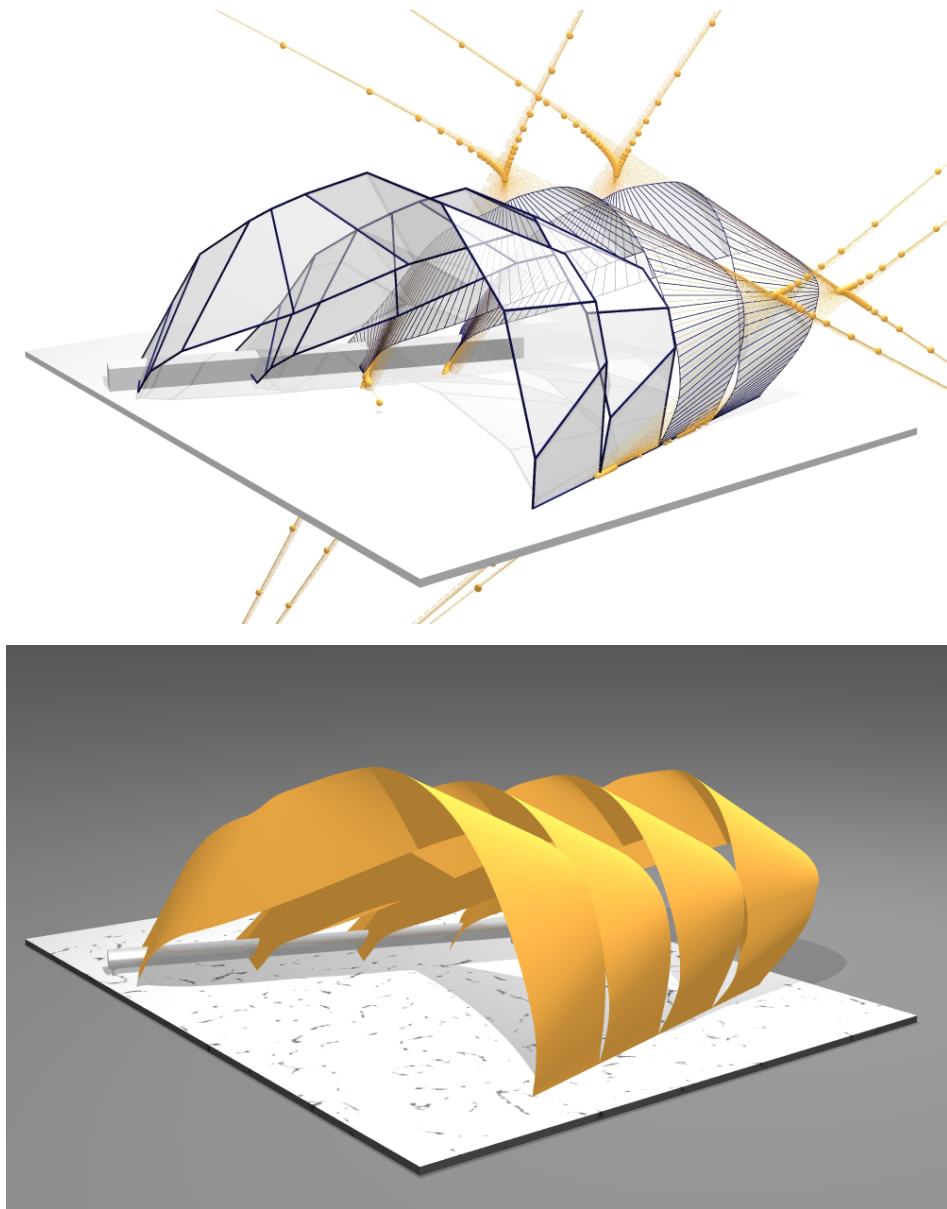


FIGURE 6.16: Design studies with developable surfaces. Two of four developable strips in the top figure show control structures, and the other two show the result of subdivision together with singular curves.

6.7 Conclusion and Future Work

We have shown how to construct and approximate surfaces with meshes composed of planar quadrilaterals. To our knowledge, approximation with conical, circular, or even just PQ meshes has not been treated before. Combining an optimization algorithm for the computation of these PQ meshes with quad-based subdivision algorithms results in a powerful modeling tool. It adapts subdivision for applications in architecture and also provides a new way of modeling developable surfaces. In particular, we have introduced and studied conical meshes, which discretize the network of principal curvature lines. They are well suited for designing free-form glass structures in architecture, and provide a simple and natural offsetting operation and the construction of a support structure from discrete surface normals.

Claim: the presented material of this chapter has been published in [83].

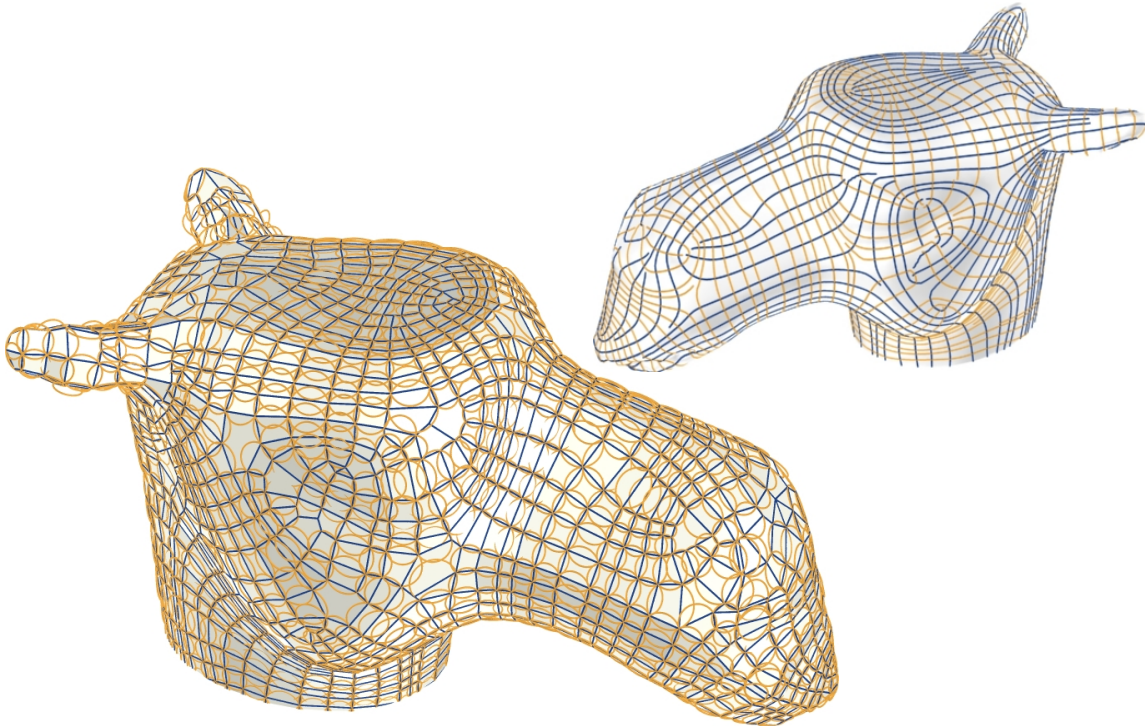


FIGURE 6.17: Circular mesh generated by optimizing a mesh generated from principal curves (top right).

6.8 Appendix: An Angle Criterion for Conical Mesh Vertices

Introduction

The concept of conical meshes was introduced in preceding sections, thus laying the foundation for significant new research in the geometry of freeform designs realized as steel/glass constructions (see Chapter 6 and the textbook [119]). This section deals with an important detail, namely the proof that the angle-based condition which characterizes conical meshes (Theorem 6.1 below), is indeed true in all cases.

6.8.1 Conical vertices

In a mesh with planar faces, each face is equipped with a unit normal vector. These normal vectors can be consistently oriented only if the mesh surface is orientable, but anyway a consistent orientation is possible for the faces adjacent to a fixed vertex \mathbf{v} . A mesh vertex is said to be *conical* if the oriented planes adjacent to \mathbf{v} are tangent to a common oriented cone of revolution. The axis of this cone can be regarded as a discrete surface normal at the vertex \mathbf{v} .

Consider a vertex \mathbf{v} of valence 4 as shown in Figure 6.18. Let L_i be the edges incident with \mathbf{v} , $i = 1, 2, 3, 4$. Let ω_i denote the unsigned angle formed by L_i and L_{i+1} (indices mod 4), $i = 1, 2, 3, 4$. We assume that no face is degenerate, i.e., any two consecutive edges are not parallel, and $\omega_i > 0$. The main result of this note is the following geometric fact.

Theorem 6.1. *A vertex \mathbf{v} of valence 4 is conical if and only if the sums of opposite angles are equal, i.e., $\omega_1 + \omega_3 = \omega_2 + \omega_4$.*

Before giving the proof of Theorem 6.1, as preparation, we shall first present several results concerning the conical property, the existence of offset meshes of a mesh, and spherical quadrilaterals which have an incircle.

A mesh with planar faces usually does not have an offset mesh consisting of planar faces which are at constant distance from the original ones. That is because planes intersecting in a common point in general lose this property when each of them is moved by a fixed distance. The following result shows that the existence of an offset mesh is equivalent to the property that all vertices of the mesh are conical.

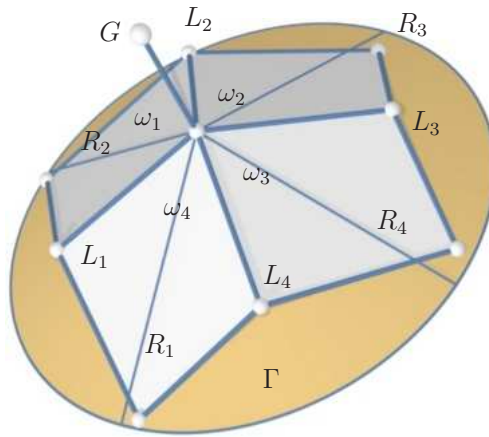


FIGURE 6.18: Conical vertex of valence four. The faces touch the common cone Γ with axis G along rulings R_1, R_2, R_3, R_4 , and have interior angles $\omega_1, \omega_2, \omega_3, \omega_4$.

Theorem 6.2. *Suppose that planes $\varepsilon_1, \dots, \varepsilon_k$, $k \geq 4$, with unit normal vectors $\mathbf{n}_1, \dots, \mathbf{n}_k$ contain the faces of a mesh which are incident with a common vertex \mathbf{v} . Translating each plane ε_i in the direction of \mathbf{n}_i by a fixed distance $d \neq 0$ yields its offset plane ε_i^d . Then the following statements are equivalent:*

1. *The offset planes ε_i^d have a point in common for some $d \neq 0$;*
2. *The offset planes ε_i^d have a point in common for all d ;*
3. *The planes $\varepsilon_1, \dots, \varepsilon_k$ are tangent to a common cone of revolution, including the plane as limit case (the limit case of a straight line does not occur).*
4. *The normal vectors $\mathbf{n}_1, \dots, \mathbf{n}_k$, regarded as points on the unit sphere S^2 , satisfy a linear equation $\langle \mathbf{n}_i, \mathbf{x}_0 \rangle = d$, for some $\mathbf{x}_0 \neq 0$ (the case $d = 0$ does not occur).*

Proof. We first show the equivalence of statements 3 and 4. Note that the oriented planes ε_i are tangent to a common oriented cone of revolution (including the limit cases of line and plane) if and only if the unit normal vectors \mathbf{n}_i lie on a circle contained in the unit sphere S^2 , including the limit case of zero radius. This happens if and only if they satisfy a linear equation $\langle \mathbf{n}_i, \mathbf{x}_0 \rangle = d$ for some $\mathbf{x}_0 \neq 0$.

The case $d = 0$ is a limit case where the cone degenerates into a line, and the circle in question is a great circle. This would imply that all the edges of the mesh emanating from the vertex are parallel, which cannot happen. Therefore, the case of $d = 0$ does not occur.

We are now going to show $1 \iff 4$. We choose a coordinate system such that $\mathbf{v} = 0$. The equation of the plane ε_i is $\mathbf{x} \in \varepsilon_i \iff \langle \mathbf{n}_i, \mathbf{x} \rangle = 0$, where \mathbf{n}_i is the oriented unit normal vector. The offset plane ε_i^d has the equation $\langle \mathbf{n}_i, \mathbf{x} \rangle = d$. Clearly, the k planes

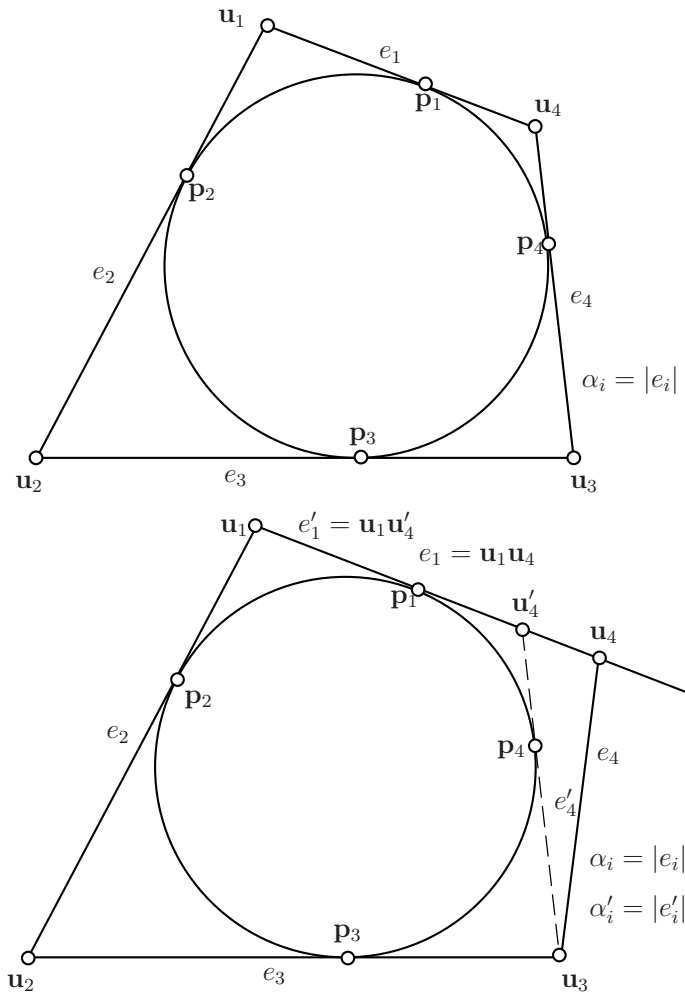


FIGURE 6.19: (a) A convex spherical quad having an incircle. This schematic illustration shows great circles as straight lines. (b) A convex spherical quad satisfying the angle criterion.

ε_i^d have a common point \mathbf{x}_0 if and only if the k normal vectors \mathbf{n}_i satisfies the equation $\langle \mathbf{n}, \mathbf{x}_0 \rangle = d$. Assuming $1, d \neq 0$ implies $\mathbf{x} \neq 0$, so 4 follows. Conversely, 4 implies that $\mathbf{x}_0 \in \varepsilon_i^d$ for all i . For all $\lambda \neq 0$, the equations $\langle \mathbf{n}_i, \lambda \mathbf{x}_0 \rangle = \lambda d$ are equivalent. This shows that $1 \iff 2$. □

6.8.2 Convex spherical quadrilaterals

Let S be a sphere centered at a mesh vertex \mathbf{v} of valence 4. Then the four faces incident with \mathbf{v} cut out four circular arcs on S which form a spherical quadrilateral $Q(\mathbf{v})$. We choose units such that S is the unit sphere. Clearly, the vertex \mathbf{v} is convex if and only if $Q(\mathbf{v})$ is a convex spherical quadrilateral. In this connection, the next result relates to the special case of Theorem 6.1 where the vertex under consideration is convex.

Theorem 6.3. *Suppose that a spherical convex quadrilateral with consecutive sides e_1, \dots, e_4 has an incircle. Let α_i be the length of the side e_i . Then $\alpha_1 + \alpha_3 = \alpha_2 + \alpha_4$. Conversely, a convex spherical quadrilateral with the property $\alpha_1 + \alpha_3 = \alpha_2 + \alpha_4$ has an incircle.*

According to p. 1038 of [159], the first part of Theorem 6.3 together with its dual version (i.e., a convex spherical quadrilateral has a circumcircle if and only if the two sums of opposite angles are equal) is due to Anders Johan Lexell [81], and the converse is due to M. J. B. Durrande [51]. However, we found it difficult to locate recent references, and so for the sake of completeness we give a proof below. As the proof of Theorem 6.3 does not refer to properties of the sphere which are different from those of the Euclidean plane, this result is also true in Euclidean geometry, as well known. For brevity, we will often use *quad* for *quadrilateral*.

Proof. We begin with a convex quad that has an incircle. Suppose that the incircle touches the four sides e_i at the points $\mathbf{p}_i \in e_i$, as shown in Figure. 6.19a. Let \mathbf{u}_i denote the vertex which is the intersection of the sides e_i and $e_{i+1} \pmod{4}$. Let $\overline{\mathbf{ab}}$ denote the spherical distance between two points \mathbf{a} and \mathbf{b} , which is the angle of the smallest arc of a great circle on S^2 connecting \mathbf{a} and \mathbf{b} .

Because the two sides incident with a vertex are tangents of the same incircle, we have

$$\overline{\mathbf{u}_1\mathbf{p}_1} = \overline{\mathbf{u}_1\mathbf{p}_2}, \quad \overline{\mathbf{u}_2\mathbf{p}_2} = \overline{\mathbf{u}_2\mathbf{p}_3}, \quad \overline{\mathbf{u}_3\mathbf{p}_3} = \overline{\mathbf{u}_3\mathbf{p}_4}, \quad \overline{\mathbf{u}_4\mathbf{p}_4} = \overline{\mathbf{u}_4\mathbf{p}_1}.$$

It follows that

$$\begin{aligned} \alpha_1 + \alpha_3 &= \overline{\mathbf{u}_1\mathbf{p}_1} + \overline{\mathbf{u}_4\mathbf{p}_1} + \overline{\mathbf{u}_2\mathbf{p}_3} + \overline{\mathbf{u}_3\mathbf{p}_3} \\ &= \overline{\mathbf{u}_1\mathbf{p}_2} + \overline{\mathbf{u}_4\mathbf{p}_4} + \overline{\mathbf{u}_2\mathbf{p}_2} + \overline{\mathbf{u}_3\mathbf{p}_4} = \alpha_2 + \alpha_4. \end{aligned}$$

Conversely, suppose that

$$\alpha_1 + \alpha_2 = \alpha_3 + \alpha_4. \tag{6.8}$$

We shall prove by contradiction that the quad $Q : \mathbf{u}_1\mathbf{u}_2\mathbf{u}_3\mathbf{u}_4$ under consideration has an incircle. Assume that Q does not have an incircle. Consider the family of the circles that are contained in the convex quad Q and tangent to e_2 and e_3 . Obviously this family either contains a circle, denoted by C , which is tangent to e_1 but not e_4 , or a circle tangent to e_4 but not e_1 . Without loss of generality, suppose that the former case occurs (see Figure. 6.19b).

Let \mathbf{u}'_4 be the unique point on e_1 between \mathbf{p}_1 and \mathbf{u}_4 such that the side $e'_4 = \mathbf{u}_3\mathbf{u}'_4$ is tangent to the circle C at \mathbf{p}_4 . Then, by the first part of the proof, the convex quad

$Q' : \mathbf{u}_1 \mathbf{u}_2 \mathbf{u}_3 \mathbf{u}'_4$ satisfies the angle criterion, i.e.,

$$\alpha'_1 + \alpha_2 = \alpha_3 + \alpha'_4, \quad (6.9)$$

where $\alpha'_1 = \overline{\mathbf{u}_1 \mathbf{u}'_4}$ and $\alpha'_4 = \overline{\mathbf{u}_3 \mathbf{u}'_4}$. Subtracting Equation. (6.9) from Equation. (6.8) yields

$$\alpha_1 - \alpha'_1 = \alpha_4 - \alpha'_4.$$

It follows that

$$\alpha_4 = \alpha'_4 + \alpha_1 - \alpha'_1 = \alpha'_4 + \overline{\mathbf{u}_4 \mathbf{u}'_4}.$$

On the other hand, by the triangle inequality, we have

$$\alpha_4 < \alpha'_4 + \overline{\mathbf{u}_4 \mathbf{u}'_4}.$$

This is a contradiction, implying that the quadrilateral Q has an incircle. This completes the proof. \square

6.8.3 General spherical quadrilaterals

We consider in this section general conical vertices of valence 4, i.e., vertices incident with four planar faces. There are three types of mesh vertices of valence 4, as defined below.

Definition 6.4. Consider a mesh vertex \mathbf{v} of valence 4 and its associated quadrilateral $Q(\mathbf{v})$.

1. \mathbf{v} is an *elliptic vertex* if the vertices of $Q(\mathbf{v})$ are contained in a hemisphere, and no vertex is contained in the spherical triangle formed by the other three vertices (note that the interior of a spherical triangle is naturally defined once we restrict ourselves to a hemisphere).
2. \mathbf{v} is a *parabolic vertex* if it is not elliptic but the vertices of $Q(\mathbf{v})$ are still contained in a hemisphere.
3. \mathbf{v} is a *hyperbolic vertex* if the four vertices of $Q(\mathbf{v})$ are not contained in any hemisphere.

Examples of these three types are shown in Figure. 6.20. Note that $Q(\mathbf{v})$ is convex if and only if \mathbf{v} is elliptic.

The four planar faces incident with \mathbf{v} have consistent normal vectors, which give rise to four oriented planes with the same normal vectors. These planes intersect the sphere S

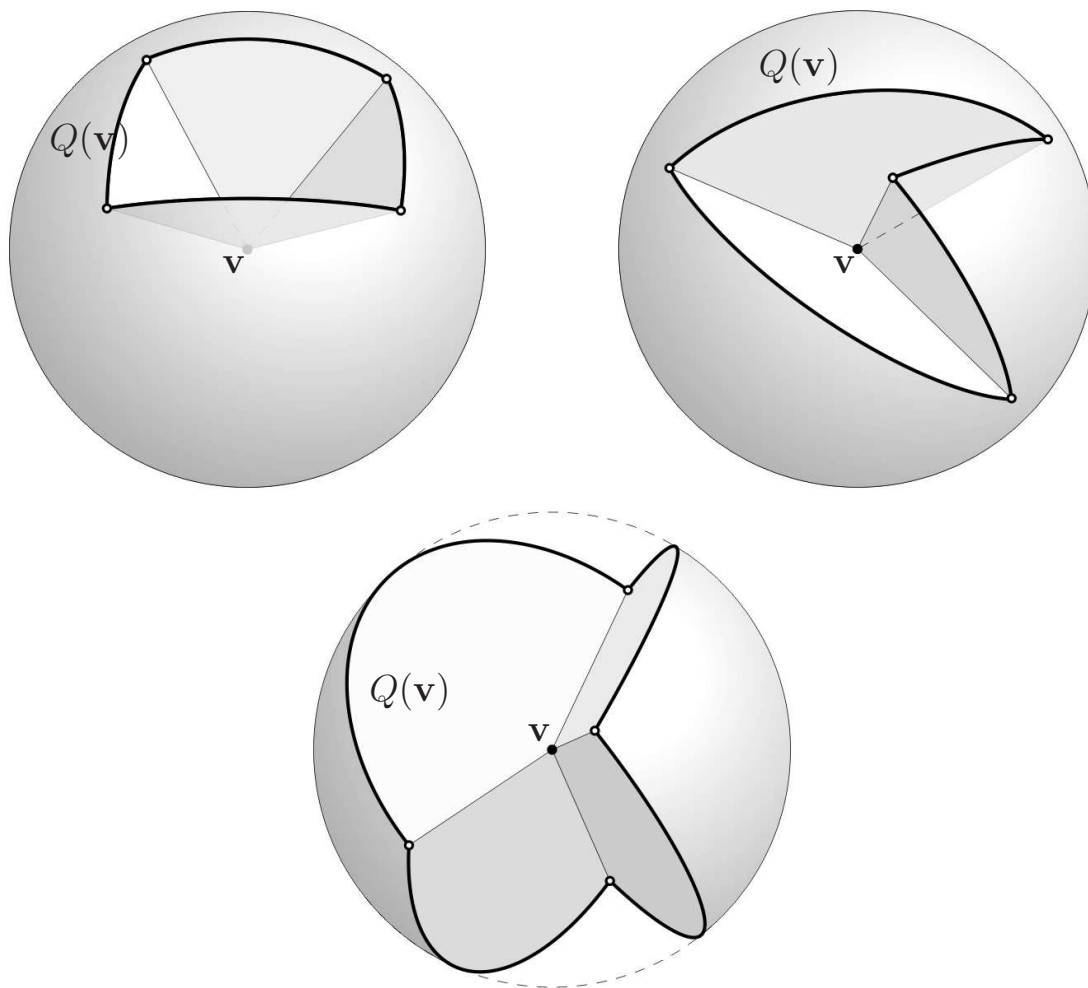


FIGURE 6.20: From left: elliptic(upper left), parabolic(upper right), and hyperbolic (lower) vertices, and their associated spherical quadrilaterals $Q(\mathbf{v})$. The vertex \mathbf{v} is the center of the sphere.

in four oriented great circles, denoted by C_i , $i = 1, 2, 3, 4$. These four circles cut each other into a number of oriented circular arcs. The two sides of each arc are distinguished as the outside and the inside, as indicated by the orientation of the plane containing the arc.

Definition 6.5. A quadrilateral with sides e_1, \dots, e_4 contained in the oriented great circles C_1, \dots, C_4 is *admissible* if the orientations of the four sides are consistent. This means the following: The quadrilateral decomposes the unit sphere into two connected components. Then it is required that the normal vectors of the four oriented planes containing to C_1, \dots, C_4 , when positioned along the sides e_1, \dots, e_4 , point consistently towards one of these two components.

Figure 6.21 shows some of admissible spherical quads. Here the sphere S is mapped onto the plane via stereographic projection, which maps the four oriented great circles

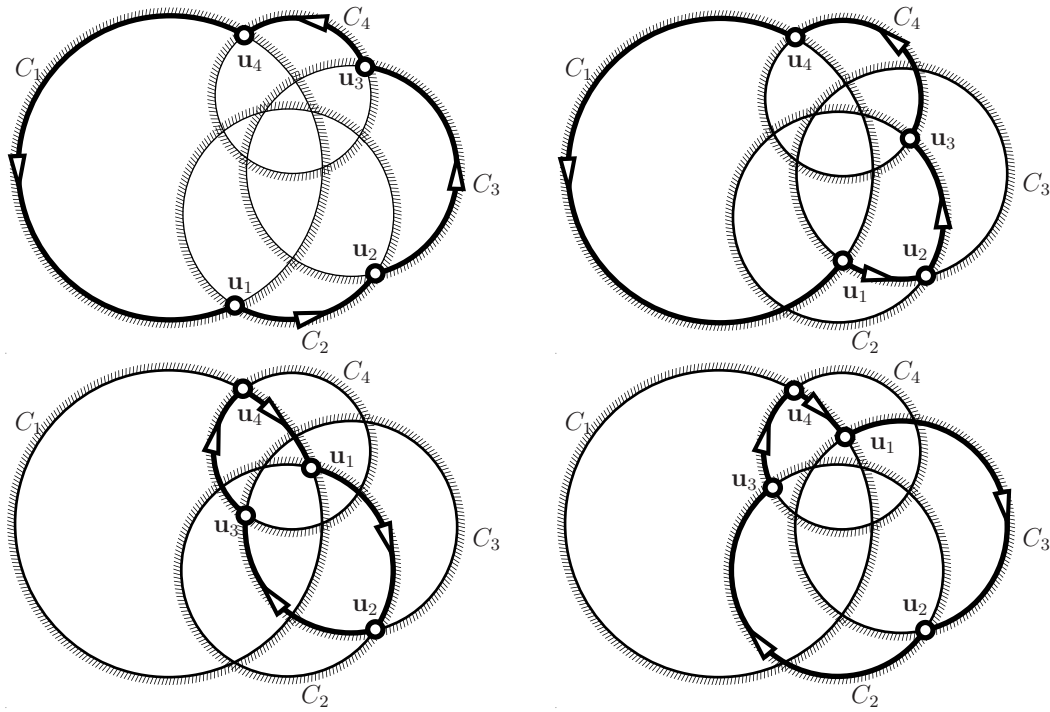


FIGURE 6.21: Stereographic images of four oriented great circles on the unit sphere. These four oriented circles correspond to four oriented planes incident with a vertex v . The four figures show 4 of the total 12 admissible quads. Circle orientations are indicated by hatched boundaries.

on S to four oriented circles in the plane, indicated by hatched boundaries. The center of projection is chosen not to be on any of the four great circles.

By saying that an admissible spherical quad has an incircle, we mean that the four oriented planes containing the four sides of the quad are tangent to a common oriented cone. Then the following is obvious: The vertex in question is conical \iff the four oriented face planes are tangent to an oriented cone \iff the oriented great circles are tangent to an oriented circle \iff the convex ones among the admissible quads have an incircle \iff the convex admissible quads satisfy the angle criterion. The last equivalence in this statement follows from Theorem 6.3.

The next theorem states that any admissible spherical quad has an incircle if and only if it satisfies the angle criterion.

Theorem 6.6. *For four oriented great circles C_1, C_2, C_3, C_4 , in the unit sphere there are in total 12 admissible quadrilaterals, including reflections in the center of the unit sphere. If the four oriented planes which carry C_1, \dots, C_4 are tangent to a common oriented cone, then all these 12 quads satisfy the angle criterion, i.e.,*

$$\omega_1 + \omega_3 = \omega_2 + \omega_4,$$

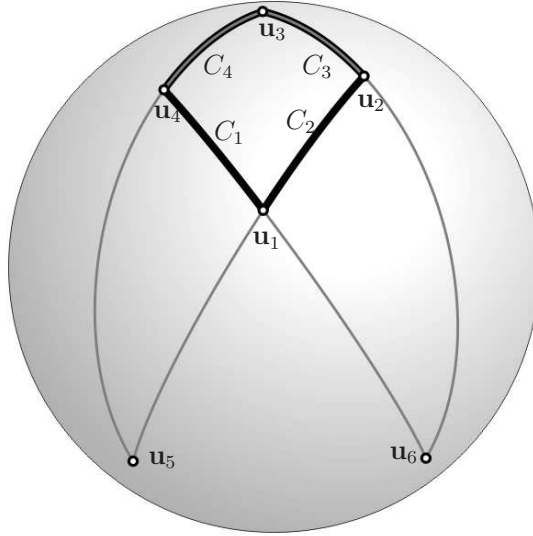


FIGURE 6.22: Converting an elliptic configuration $Q_e : \mathbf{u}_1\mathbf{u}_2\mathbf{u}_3\mathbf{u}_4$ into a parabolic configuration $Q_{1,p} : \mathbf{u}_1\mathbf{u}_6\mathbf{u}_3\mathbf{u}_5$.

where ω_i is the unsigned length of the i -th side of an admissible quad. Also the reverse implication is true: If any of the 12 admissible quads satisfies the angle criterion, the four oriented planes are tangent to a common oriented cone.

Proof. The four circles C_i have in total 12 pairwise intersection points, since $2 \cdot \binom{4}{2} = 2 \cdot 6 = 12$. Pick one of these 12 intersection points. Without loss of generality, suppose that this point is $\mathbf{u}_4 \in C_1 \cap C_4$. Now we count how many admissible quad contain \mathbf{u}_4 . In view of the assumption of consistent orientations, there are two ways to choose the arcs of C_1 and C_4 which start at \mathbf{u}_4 and are part of an admissible quad. For each of these choices, we have either a quad with sides traversing the four circles in the order $C_1C_2C_3C_4$ or in the order $C_1C_3C_2C_4$ (see Figure. 6.21). Thus, there are in total 4 quads passing through \mathbf{u}_4 . Since there are 12 pairwise intersection points among the four circles, we have counted $12 \cdot 4 = 48$ admissible quads. Since each quad has four vertices, it is counted 4 times. So the number of distinct admissible quads is $48/4 = 12$. This proves the first part of the theorem.

Obviously, there is a convex quad among the 12 admissible ones; in fact, there are two, which are reflections of each other. We are going to show that all admissible quads can be obtained from a convex admissible quad Q_e by operations which preserve the angle balance in both directions.

Let Q_e be a convex admissible quad with vertices \mathbf{u}_i , $i = 1, 2, 3, 4$. Suppose that the sides e_i of Q_e are on the circles C_1, C_2, C_3, C_4 (in this order) and that $\mathbf{u}_i \in C_i \cap C_{i+1}$ (indices modulo 4). Then a parabolic admissible quad $Q_{1,p} : \mathbf{u}_1\mathbf{u}_6\mathbf{u}_3\mathbf{u}_5$ can be derived from Q_e

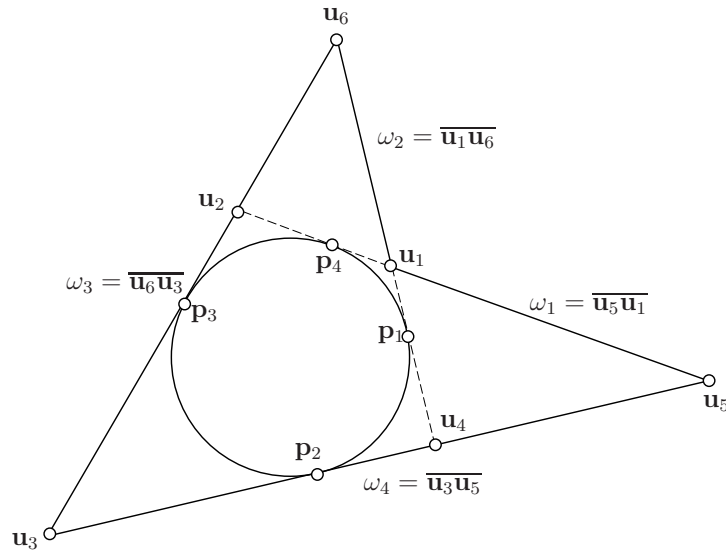


FIGURE 6.23: An admissible parabolic quad having an incircle.

by traversing the circles in the order C_1, C_3, C_4, C_2 , thereby making \mathbf{u}_1 a concave vertex (see Figure. 6.22). Similarly, we can derive three other admissible parabolic quads $Q_{2,p}$, $Q_{3,p}$, and $Q_{4,p}$. By reflecting the $Q_{i,p}$ in the center of the sphere, $i = 1, 2, 3, 4$, we obtain in total 8 parabolic quads.

Now we derive hyperbolic quads from Q_e . We replace the vertices \mathbf{u}_1 and \mathbf{u}_3 of Q_e by their diametrically opposite points \mathbf{u}_1^* and \mathbf{u}_3^* and arrive at the admissible quad $Q_h : \mathbf{u}_1^* \mathbf{u}_2 \mathbf{u}_3^* \mathbf{u}_4$ of hyperbolic type. If we flip \mathbf{u}_2 and \mathbf{u}_4 instead, we get the quad $Q_h^* : \mathbf{u}_1 \mathbf{u}_2^* \mathbf{u}_3 \mathbf{u}_4^*$, which is the reflection of Q_h . In this way, 2 hyperbolic quads are derived.

Together with Q_e and its reflection Q_e^* , we have obtained 12 admissible quads, which, in view of the total number 12 shown earlier, already exhaust the set of all admissible quads. Hence, we conclude that any admissible quad can be obtained from a convex admissible quad with the above operations.

Next we are going to show that, for any of the nonconvex admissible quads obtained above, the angle criterion characterizes the property that C_1, \dots, C_4 are tangent to a common oriented circle. First consider the case of parabolic admissible quads, using the quad $Q_{1,p} : \mathbf{u}_1 \mathbf{u}_2 \mathbf{u}_3 \mathbf{u}_5$ in Figure. 6.23 for illustration. Denote the lengths of the sides of $Q_{1,p}$ by $\omega_1 = \overline{\mathbf{u}_5 \mathbf{u}_1}$, $\omega_2 = \overline{\mathbf{u}_1 \mathbf{u}_6}$, $\omega_3 = \overline{\mathbf{u}_6 \mathbf{u}_3}$ and $\omega_4 = \overline{\mathbf{u}_3 \mathbf{u}_5}$. First suppose that an incircle exists, which means that the convex quad $Q_e : \mathbf{u}_1 \mathbf{u}_2 \mathbf{u}_3 \mathbf{u}_4$ has an incircle. Using the fact that the two tangents from a vertex to a circle have equal lengths, we have

$$\begin{aligned} \omega_1 + \omega_3 &= \overline{\mathbf{u}_5 \mathbf{u}_1} + \overline{\mathbf{u}_3 \mathbf{u}_6} = \overline{\mathbf{p}_4 \mathbf{u}_5} - \overline{\mathbf{p}_4 \mathbf{u}_1} + \overline{\mathbf{u}_3 \mathbf{p}_3} + \overline{\mathbf{p}_3 \mathbf{u}_6} = \overline{\mathbf{p}_2 \mathbf{u}_5} - \overline{\mathbf{p}_1 \mathbf{u}_1} + \overline{\mathbf{u}_3 \mathbf{p}_2} + \overline{\mathbf{p}_1 \mathbf{u}_6} \\ &= \overline{\mathbf{p}_1 \mathbf{u}_6} - \overline{\mathbf{p}_1 \mathbf{u}_1} + \overline{\mathbf{u}_3 \mathbf{p}_2} + \overline{\mathbf{p}_2 \mathbf{u}_5} = \overline{\mathbf{u}_1 \mathbf{u}_6} + \overline{\mathbf{u}_3 \mathbf{u}_5} = \omega_2 + \omega_4. \end{aligned}$$

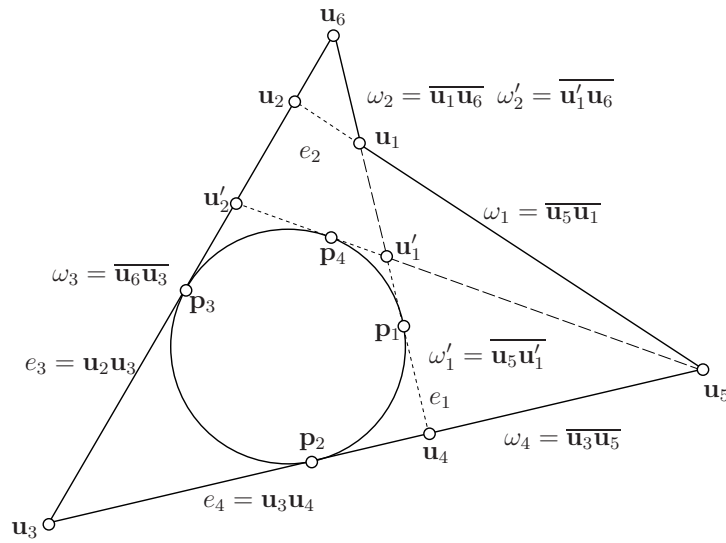


FIGURE 6.24: A parabolic admissible quad satisfying the angle criterion.

It follows that $Q_{1,p}$ satisfies the angle criterion.

Conversely, suppose that $Q_{1,p}$ satisfies the angle criterion, i.e.,

$$\omega_1 + \omega_3 = \omega_2 + \omega_4. \quad (6.10)$$

We shall prove by contradiction that $Q_{1,p}$ has an incircle. Assume otherwise, i.e., Q_e has no incircle. Similar to the proof of Theorem 6.3, consider the family of the circles that are contained in the convex quad Q_e and tangent to $e_3 = \mathbf{u}_2\mathbf{u}_3$ and $e_4 = \mathbf{u}_3\mathbf{u}_4$. Then, there is a circle C in this family that is either tangent to e_1 but not e_2 or tangent to e_2 but not e_1 . Without loss of generality, we suppose the former to be the case, as shown in Figure. 6.24.

Let \mathbf{u}'_1 be the unique point on the side $\mathbf{u}_1\mathbf{p}_1$ such that the great circle containing the side $\mathbf{u}'_1\mathbf{u}_5$ is tangent to the circle C inside the convex quad Q_e . Then the new convex quad $Q'_e : \mathbf{u}'_1\mathbf{u}_2\mathbf{u}_3\mathbf{u}_4$ has an incircle. By the preceding argument, the new parabolic quad $Q'_{1,p} : \mathbf{u}'_1\mathbf{u}_6\mathbf{u}_3\mathbf{u}_5$ satisfies the angle criterion, that is,

$$\omega'_1 + \omega_3 = \omega'_2 + \omega_4, \quad (6.11)$$

where $\omega'_1 = \overline{\mathbf{u}_5\mathbf{u}'_1}$ and $\omega'_2 = \overline{\mathbf{u}'_1\mathbf{u}_6}$. Subtracting Eq. (6.11) from Eq. (6.10) yields

$$\omega_1 - \omega'_1 = \omega_2 - \omega'_2.$$

It follows that

$$\omega'_1 = \omega_1 + \omega'_2 - \omega_2 = \omega_1 + \overline{\mathbf{u}'_1\mathbf{u}_1}.$$

On the other hand, by the triangle inequality, we have

$$\omega'_1 < \omega_1 + \overline{\mathbf{u}'_1 \mathbf{u}_1}.$$

This is a contradiction, implying that $Q_{1,p}$ has an incircle. It follows that the angle criterion characterizes the existence of an incircle also for parabolic quads.

Next we consider the case of hyperbolic quads. We are going to show that $Q_h : \mathbf{u}_1^* \mathbf{u}_2^* \mathbf{u}_3^* \mathbf{u}_4^*$, which is constructed from Q_e , has an incircle if and only if Q_h satisfies the angle criterion. This is easier than in the parabolic case, because the side lengths of Q_h are given by $\pi - \alpha_i$, where the α_i are the side lengths of Q_e . Hence, Q_h satisfies the angle criterion if and only if Q_e does, i.e., if and only if an incircle exists. This completes the proof. \square

Remark 1: It is possible that a non-admissible quad also enjoys the angle balance, because it might be admissible for a different assignment of orientations, and so the four corresponding planes are tangent to a different oriented cone. This, however, does not diminish the value of Theorem 6.6 for applications if we consider admissible quads only. This is the case if the quads under consideration come from a consistently oriented quad mesh.

Remark 2: In a quad mesh with planar faces which approximates a smooth surface and where almost all vertices have valence four, vertices are typically elliptic or hyperbolic, whereas parabolic vertices occur not as often. This is similar to the distribution of parabolic points in smooth surfaces. The fact that most of the 12 admissible quads discussed in Theorem 6.6 are parabolic does not contradict this behavior.

Proof of Theorem 6.1

Proof. The planar faces incident with a mesh vertex \mathbf{v} of valence 4 are consistently oriented such that the spherical quad $Q(\mathbf{v})$ is admissible in the sense of Definition 6.5. The side lengths of the spherical quad $Q(\mathbf{v})$ are equal to the interior angles of the faces mentioned in the statement of Theorem 6.1. Hence, the proof follows from Theorem 6.6. \square

Claim: the presented material in this section has been published in [152].

Chapter 7

Geometry of Multi-layer Freeform Structures for Architecture

7.1 Introduction

Freeform shapes in architecture is an area of great engineering challenges and novel design ideas. Obviously the design process, which involves shape, feasible segmentation into discrete parts, functionality, materials, statics, and cost, at every stage benefits from a complete knowledge of the complex interrelations between geometry requirements and available degrees of freedom. Triangle meshes – the most basic, convenient, and structurally stable way of representing a smooth shape in a discrete way – do not support desirable properties of meshes relevant to building construction (most importantly, “torsion-free” nodes). Alternatives, namely quad-dominant and hexagonal meshes tend to have less weight, and can be constructed with geometrically optimized nodes and beams. However, the geometry of such meshes is more difficult. Especially challenging are aesthetic layout of edges and the geometric constraints of planar faces and optimized nodes.

Only recently, researchers have become interested in the geometric basics of single- and multilayer freeform structures which are not based on triangle meshes. Existing literature has been motivated by problems in the fabrication of steel/glass and other structures and mostly aims at the realizations of freeform shapes by meshes with planar faces [39, 56, 83, 133]. The latter paper (cf. Chapter 6) introduced *conical meshes* which have planar faces and possess offset meshes at constant face-face distance from the base mesh. They can serve as the basis of multi-layer constructions, and so for the first time the problem of multilayered realization of a freeform surface by means of planar parts was solved in principle.

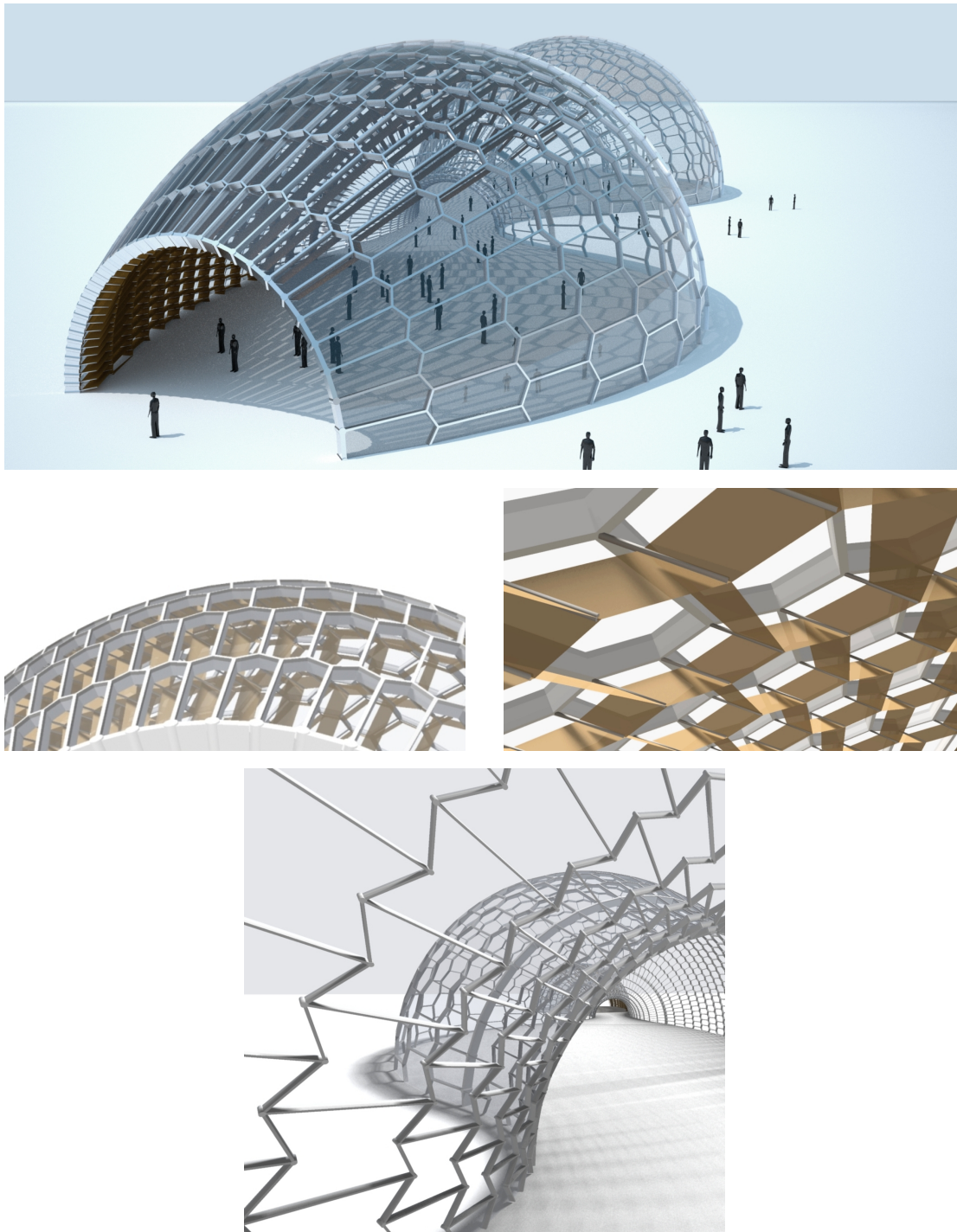


FIGURE 7.1: This architectural free form structure – built of beams of constant height meeting in optimized nodes and covered with planar glass facets – was designed using the theory and algorithms presented in this chapter. Our method also allows for the construction of secondary parallel offsets at a variable distance, here physically realized as a structure designed to cast shadows which is optimized to reduce heat load for particular sun positions.

Until now the wealth of interesting geometry relevant to the construction of freeform structures in architecture has been explored only to a small extent. It is the aim of our work to show how the *local structure* of single- and multi-layer constructions can be analyzed with *mesh parallelism* as the main tool. This concept allows us to encode the existence of node axes and offsets in a discrete Gauss image, and to define discrete curvatures in a natural way. Optimization in the linear space of meshes parallel to a given mesh yields a modeling tool. A particularly important and interesting type of meshes are those possessing *edge offsets*. We show how mesh parallelism establishes a connection between meshes with edge offsets and Koebe polyhedra. Thus the research presented here is situated at the meeting point of discrete differential geometry, modeling, geometry processing, and architectural design.

Previous work in discrete differential geometry.

Most of the work relevant to the present study concerns quadrilateral meshes with planar faces, which discretize so-called conjugate curve networks on surfaces [126]. They are a basic concept in the integrable system viewpoint of discrete differential geometry [19]. Both the *circular* meshes and the *conical* meshes are special cases which possess particularly nice geometric properties, and which correspond to those conjugate curve networks which are also orthogonal, i.e., the principal curvature lines. For convergence of such meshes to the network of principal curvature lines, see [19]. The fact that principal curvature lines are a concept of Lie sphere geometry has a discrete manifestation in the unified treatment of circular and conical meshes as principal meshes of Lie sphere geometry [20]. Elementary relations between circular and conical meshes, and meshes which enjoy both properties are discussed by Pottmann and Wallner [114]. Special cases of the parallel meshes which are the topic of our work have been considered by [126]. Our work on curvature extends results of Schief [128], who defined a mean and Gaussian curvature for circular meshes via surface areas of discrete offset surfaces. That method apparently has first been applied to simplicial surfaces by Nishikawa et al. [98] in a different context. Another classical definition of the mean curvature through variation of surface area of simplicial surfaces has been investigated by Polthier [108]. Discrete minimal surfaces realized as circular and conical meshes are the topic of several contributions (e.g. [17, 22]). Attempts to construct discrete minimal surfaces with planar quad meshes have been made by [109]. The discrete minimal surfaces of [22] are particularly interesting because they provide a class of polyhedral surfaces with the *edge offset property*. It turns out in Section 7.3.2 that the edge offset property is closely related to results on orthogonal circle patterns [18, 134].

Previous work in geometry processing.

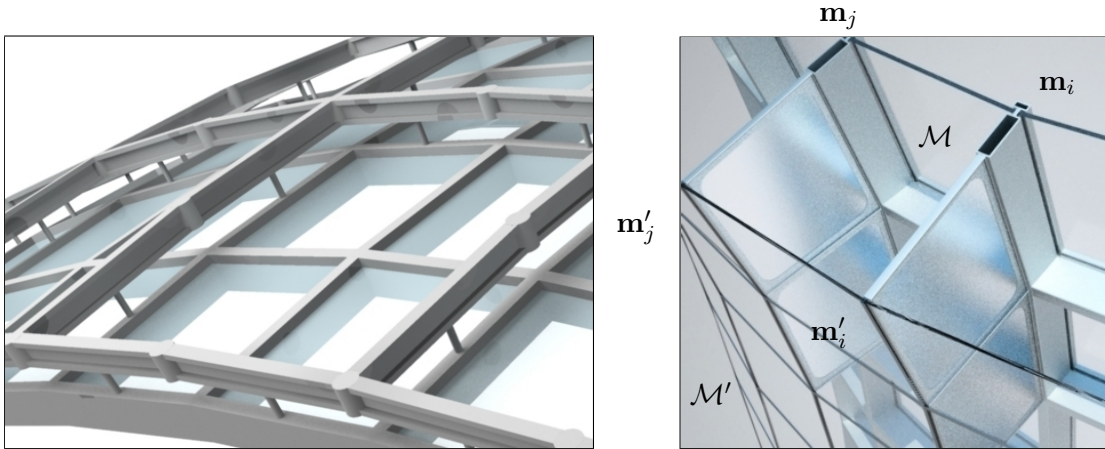


FIGURE 7.2: Multi-layer constructions based on the geometric support structure defined by two parallel meshes \mathcal{M} , \mathcal{M}' at approximately constant distance. On the left, the lower layer of the glass roof is suspended from the upper layer which has a structural function. The right hand image shows a rudimentary construction of a glass facade where the closed space between layers has an insulating function.

Only some of the papers mentioned above address the computation of the meshes they deal with [18, 22, 83]. The latter paper (cf. Chapter 6) demonstrates how to design meshes with planar faces, circular meshes, and conical meshes by subdivision and optimization, and also how to approximate a given shape by a circular or conical mesh. As input for such mesh optimization algorithms any mesh aligned along a network of principal curvature lines may be used (see e.g. [3, 142]). Approximation of smooth surfaces by meshes with planar faces, without a focus on support structures and multilayer constructions can be achieved by variational shape approximation [36]. Cutler and Whiting [39] modified this method with regard to aesthetics and architectural design. More generally, various research projects on geometry for architecture in general are promoted by the Smart Geometry group [137].

Meshes whose faces are mostly planar 5-gons or 6-gons have certain desirable properties, but such meshes have received considerably less attention in the graphics community than triangle meshes and quad meshes. Some notable exceptions are papers on refinement processes by Akleman et al. [2], and on combined primal/dual subdivision [102]. However, they do not consider planarity of faces or other aspects relevant to building construction.

7.2 Mesh Parallelism for Architecture

7.2.1 Motivation and introduction

Glass panels and multilayer metal sheets for roofing structures are planar as a rule, with only a few exceptions. The reason for this is mostly the prohibitive cost of manufacturing

them otherwise. Obviously, every implementation of a freeform shape in terms of flat primitives faces the problem of approximating the given surface by a mesh with planar faces. If *triangle meshes* are employed, the geometry part of the solution of this problem consists of choosing the vertices and deciding which vertices to connect by edges. If faces can have more than three vertices, this approximation task is not so simple, because the condition of planar faces is no longer fulfilled automatically. It should be mentioned that also statics is simpler if we stay with triangle meshes. There are, however, the following issues which make other solutions attractive:

- In a steel/glass or other construction based on a triangle mesh, typically six beams meet in a node. This means a significantly higher node complexity compared to other types of meshes (see Figure. 7.4, right).
- Experience shows that the per area cost of triangular glass panels is higher than that of quadrilateral panels. This is mainly due to the fact that quadrilaterals fill their smallest rectangular bounding boxes better than triangles do.
- Generally one aims at less steel, more glass, and less weight, which also suggests the use of non-triangular faces.
- For the actual construction, torsion-free nodes are preferred. For this concept, see Figure 7.4 and the text below. The geometric theory however tells us that for triangle meshes in general torsion-free nodes do not exist.
- Apart from trivial cases, triangle meshes do not possess offsets at constant face-face or edge-edge distance; neither is it possible to use triangle meshes as basis of a multilayer construction where only the basic requirement of parallelity of layers is imposed.

This section shows how the concept of *parallelism*, which applies to meshes with planar faces, can be used to gain a unified view of these issues, especially geometrically optimal nodes and offset properties [26]. Consider two meshes \mathcal{M} and \mathcal{M}' which are combinatorially equivalent, i.e., there is a 1-1 correspondence between vertices and edges. We call the meshes $\mathcal{M}, \mathcal{M}'$ *parallel*, if corresponding edges are parallel (see Figure 7.3). Before we consider parallel meshes from the mathematical viewpoint, we first describe some geometric problems connected with discrete surfaces in architecture where this concept occurs naturally.

Multilayer constructions.

Structures like those schematically depicted by Figure 7.2 include not only one mesh, but several meshes, corresponding to the different layers of the construction. It is natural to demand that meshes which correspond to different layers are parallel.

Geometrically optimal nodes.

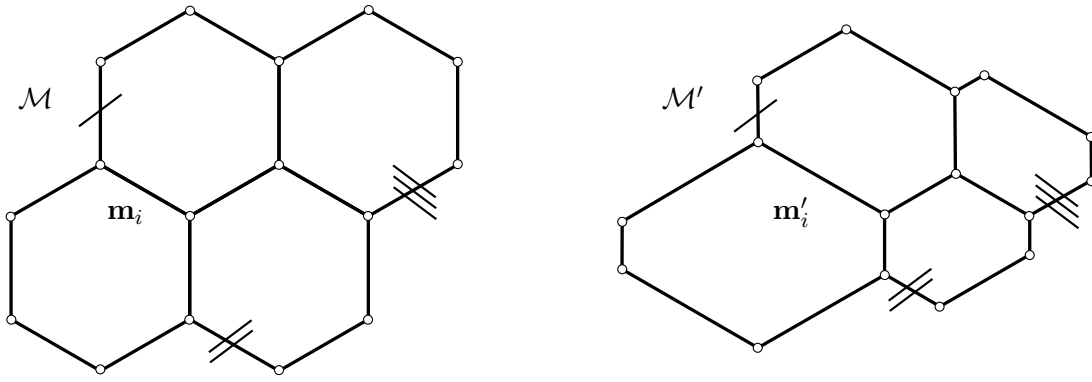


FIGURE 7.3: Meshes $\mathcal{M}, \mathcal{M}'$ with planar faces are parallel if they are combinatorially equivalent, and corresponding edges are parallel.

In the actual realization of a polyhedral surface \mathcal{M} as a steel/glass roof, planar glass panels are held by prismatic *beams* following the edges of \mathcal{M} (see Figure 7.4). A beam is symmetric with respect to its *central plane* which passes through the edge corresponding to the beam. A *node* corresponds to a vertex \mathbf{m}_i and connects incoming beams in a way which supports the force flow imposed by the overall statics of the structure. Node construction and manufacturing are greatly simplified if there is a *node axis* A_i , which is contained in the central planes of incoming beams (see Figure 7.4, left). Figure 7.4, right, shows the case of a welded node which does not have a node axis. Obviously the handling and manufacturing of such ‘nodes with torsion’ is more complicated than the case with a node axis. Geometrically, lines A_i passing through the vertices \mathbf{m}_i of a given mesh are a collection of node axes, if and only if

$$\mathbf{m}_i \mathbf{m}_j \text{ is an edge} \implies \text{node axes } A_i, A_j \text{ are co-planar.}$$

To avoid pathological cases, we forbid that node axes lie in edges.

The following simple but fundamental proposition establishes a property of a collection of node axes associated with a mesh \mathcal{M} . It relates node axes to an auxiliary mesh \mathcal{M}' which is parallel to the given mesh, and is illustrated by Figure 7.4, left.

Proposition 7.1. *If the meshes $\mathcal{M}, \mathcal{M}'$ with vertices $\mathbf{m}_i, \mathbf{m}'_i$ ($\forall i : \mathbf{m}_i \neq \mathbf{m}'_i$) are parallel, then the lines $A_i = \mathbf{m}_i \vee \mathbf{m}'_i$ serve as node axes for the mesh \mathcal{M} (provided no line A_i contains an edge).*

Conversely assume that a simply connected mesh \mathcal{M} is equipped with node axes A_i passing through its vertices \mathbf{m}_i . Then there exists a mesh \mathcal{M}' parallel to \mathcal{M} , such that A_i is spanned by corresponding vertices $\mathbf{m}_i, \mathbf{m}'_i$.

Proof. This is shown in [120]. Part (i) is elementary. For part (ii) we start with a vertex $\mathbf{m}'_{i_0} \in A_{i_0}$ and construct further vertices of \mathcal{M}' by the requirement that corresponding edges of \mathcal{M} and \mathcal{M}' are parallel, and that $\mathbf{m}'_i \in A_i$ for all i . \square

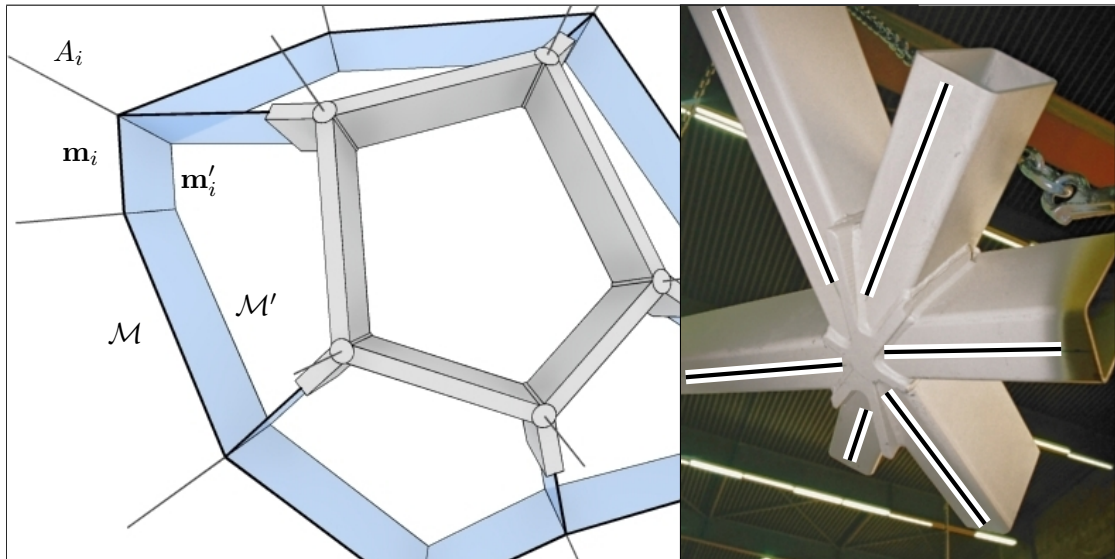


FIGURE 7.4: *Left:* This figure shows nodes, supporting beams, and the underlying geometric support structure of a steel/glass construction, based on a mesh \mathcal{M} (vertices \mathbf{m}_i) and its parallel mesh \mathcal{M}' (vertices \mathbf{m}'_i). All beams are symmetric with respect to their central plane (blue); at an optimized (*torsion-free*) node \mathbf{m}_i the central planes of supporting beams pass through the node axis $A_i = \mathbf{m}_i \vee \mathbf{m}'_i$. *Right:* A node without axis, with geometric torsion.

Geometric support structure.

There is a point of view which unifies the different cases where parallel meshes occur – regardless of whether a parallel mesh is physically realized, or is only used in the definition of node axes. This point of view is that we emphasize the construction elements which *connect* different layers – physically present or not – and which are transverse to the mesh \mathcal{M} under consideration. In Figure 7.2, right, these construction elements are the glass panels which connect the two parallel meshes $\mathcal{M}, \mathcal{M}'$. In Figure 7.4, left, these construction elements are the beams. We shrink those elements until they have zero width (this is schematically indicated in Figure 7.4, left). They then become planar quadrilaterals transverse to the mesh \mathcal{M} , passing through the edges of \mathcal{M} . Those transverse quads which are adjacent to a node \mathbf{m}_i , have a common edge which lies in the node axis A_i . Such a collection of quads is called a *geometric support structure* of the mesh \mathcal{M} (see Figures 7.1, 7.2, 7.4, 7.6, 7.13, and 7.14).

Obviously, a collection of node axes for \mathcal{M} almost uniquely defines the quads of a geometric support structure for \mathcal{M} – the only degree of freedom left is the quad boundaries opposite to the edges of \mathcal{M} . By Proposition 7.1, any geometric support structure joins the edges of \mathcal{M} with the respective corresponding edges of a mesh \mathcal{M}' parallel to \mathcal{M} . Every node axis A_i joins corresponding vertices $\mathbf{m}_i, \mathbf{m}'_i$.

Specific problems

We are going to discuss meshes with certain geometric properties relevant to architectural design, especially support structures. An important property of this kind is that a mesh

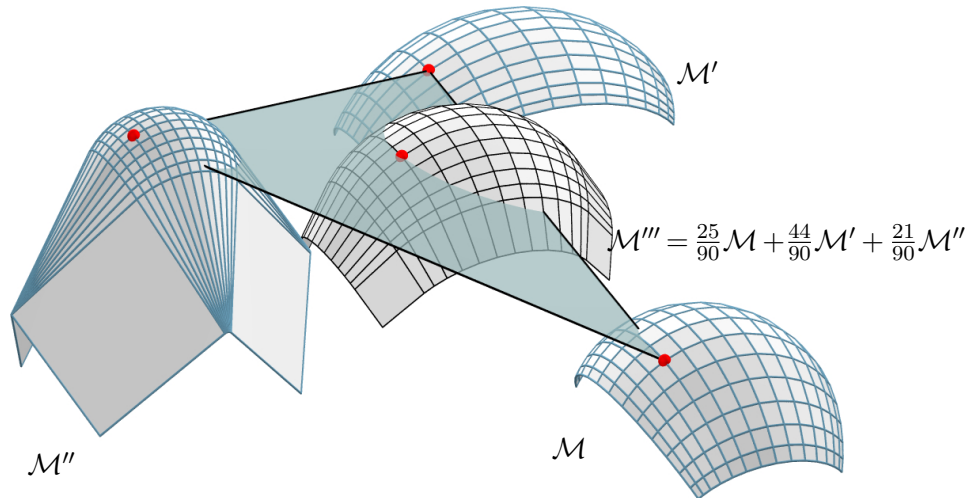


FIGURE 7.5: The set $\mathcal{P}(\mathcal{M})$ of meshes parallel to a given mesh \mathcal{M} is a linear space and can be explored by a linear blend of some of its elements.

possesses a support structure whose beams are of constant height – this is the class of *edge offset meshes* defined later. Generally speaking, the more properties we require a mesh to have, the fewer degrees of freedom are available. We therefore encounter the following problems:

- *The approximation problem.* Is it possible to approximate a given shape by a mesh contained in a specific class of meshes? For example, this is possible for the conical quad meshes, as shown in Chapter 6, but not for the quad meshes with the edge offset property. However we shall see that we can approximate arbitrary surfaces if we relax the requirement of constant height a little bit.
- *The design problem.* How can we explore or even completely describe the set of meshes with a specific geometric property? This question is especially important in cases where the approximation problem is not solvable. For example, we will introduce some geometric transformations which change shape but preserve the edge offset property.

7.2.2 Basics of mesh parallelism

Mesh parallelism and the spaces $\mathcal{C}(\mathcal{M})$ and $\mathcal{P}(\mathcal{M})$.

A mesh \mathcal{M} is represented by the list $(\mathbf{m}_1, \dots, \mathbf{m}_N) \in \mathbb{R}^{3N}$ of vertices and the mesh combinatorics, i.e., the information which vertices belong to common edges and faces. We use $\mathcal{C}(\mathcal{M})$ to denote the linear $3N$ -dimensional space of meshes *combinatorially equivalent* to \mathcal{M} . If $\mathcal{M}', \mathcal{M}''$ have the same combinatorics, a linear combination $\lambda' \mathcal{M}' + \lambda'' \mathcal{M}''$ is defined vertex-wise; this operation corresponds to the linear combination of vectors in \mathbb{R}^{3N} .

Meshes $\mathcal{M}, \mathcal{M}'$ are *parallel*, if $\mathcal{M}' \in \mathcal{C}(\mathcal{M})$ and corresponding edges are parallel (see Figures 7.3 and 7.5). We use that definition only if the faces of \mathcal{M} are planar. Clearly

then corresponding faces of \mathcal{M} and \mathcal{M}' lie in parallel planes (parallelity of planes alone is sufficient to guarantee parallelity of edges, if no pair of adjacent faces are co-planar). We denote the set of meshes parallel to \mathcal{M} by $\mathcal{P}(\mathcal{M})$. To avoid pathological cases we require that the mesh \mathcal{M} which defines the space $\mathcal{P}(\mathcal{M})$ has only nonzero edges. Trivial ways of producing meshes parallel to \mathcal{M} are to translate and scale \mathcal{M} . Since triangles with parallel edges are scaled copies of each other, two parallel triangle meshes are scaled copies of each other [120]. This is the reason why we do not consider triangle meshes.

Suppose $\mathcal{M}', \mathcal{M}'' \in \mathcal{P}(\mathcal{M})$. Then, for each edge $\mathbf{m}_i\mathbf{m}_j$, the vectors $\mathbf{m}'_i - \mathbf{m}'_j$, $\mathbf{m}''_i - \mathbf{m}''_j$ are multiples of $\mathbf{m}_i - \mathbf{m}_j$. It follows that any expression $(\lambda'\mathbf{m}'_i + \lambda''\mathbf{m}''_i) - (\lambda'\mathbf{m}'_j + \lambda''\mathbf{m}''_j)$ is a multiple of $\mathbf{m}_i - \mathbf{m}_j$. This shows that the linear combination $\lambda'\mathcal{M}' + \lambda''\mathcal{M}''$ is also parallel to \mathcal{M} , so $\mathcal{P}(\mathcal{M})$ is a linear subspace of $\mathcal{C}(\mathcal{M})$. The zero vector of both $\mathcal{P}(\mathcal{M})$ and $\mathcal{C}(\mathcal{M})$ is the mesh $\mathbf{o} = 0 \cdot \mathcal{M}$, all of whose vertices coincide with the origin of the coordinate system. Linear blending between three meshes in the space $\mathcal{P}(\mathcal{M})$ is illustrated by Figure 7.5.

The space $\mathcal{P}(\mathcal{M})$ does not only contain ‘nice’ meshes. We may see various undesirable effects such as unevenly distributed faces, sharp edges of regression, or overlapping regions. Nevertheless, it is both theoretically and practically useful to have the entire space $\mathcal{P}(\mathcal{M})$ at our disposal. Even visually unpleasant meshes in $\mathcal{P}(\mathcal{M})$ will turn out to be helpful in the computation of optimal beam layouts (see Figure. 7.14d).

Computing in the space of parallel meshes.

As $\mathcal{P}(\mathcal{M})$ is a linear space, it is important to determine a basis. We observe that a mesh $\mathcal{M}' \in \mathcal{C}(\mathcal{M})$ is contained in $\mathcal{P}(\mathcal{M})$ if and only if

$$\mathbf{m}_i\mathbf{m}_j \text{ is an edge} \implies \mathbf{m}'_i - \mathbf{m}'_j = \lambda_{ij}(\mathbf{m}_i - \mathbf{m}_j). \quad (7.1)$$

We can therefore determine $\mathcal{P}(\mathcal{M})$ as the solution space of the system of equations

$$(\mathbf{m}'_i - \mathbf{m}'_j) \times (\mathbf{m}_i - \mathbf{m}_j) = 0 \quad \text{for all edges } \mathbf{m}_i\mathbf{m}_j. \quad (7.2)$$

A rough count of degrees of freedom (one d.o.f. per edge, two closure conditions per face, 3 d.o.f. for the translations in space) shows that e.g. for open meshes we can expect $\dim \mathcal{P}(\mathcal{M}) = \# \text{ edges} - 2 \times \# \text{ faces} + 3$. The actual solution of (7.2) is done via SVD. Thresholding of small singular values is supported by an a priori estimate for $\dim \mathcal{P}(\mathcal{M})$.

Once a basis is available, the minimization of linear and quadratic functionals (e.g. a fairness functional) under constraints (like fixed points) and linear side conditions (like edge length inequalities) presents no problems.

7.3 Offset Meshes

Meshes are *offsets* of each other, if they are parallel, and in addition their distance from each other is constant throughout the mesh. This notion is similar to the concept of smooth *offset surfaces* [91], but in contrast to the smooth case, for meshes there are various different definitions of *distance*. Consequently, there are also various different notions of offset mesh. Two of them, *face offsets* and *edge offsets* are relevant to problems of architectural design of freeform surfaces.

For multilayer constructions (cf. Figure 7.2) it is natural to require that the distance of corresponding faces is constant. Parallel meshes with this property (*face offsets*) are the topic of Chapter 6.

Another type of offset occurs when we employ parallel meshes for the layout of beams in a steel/glass construction based on a given mesh \mathcal{M} . The beams are a physical realization of a geometric support structure which connects two parallel meshes \mathcal{M} and \mathcal{M}' . If the distance of corresponding *edges* in \mathcal{M} and \mathcal{M}' is constant throughout the

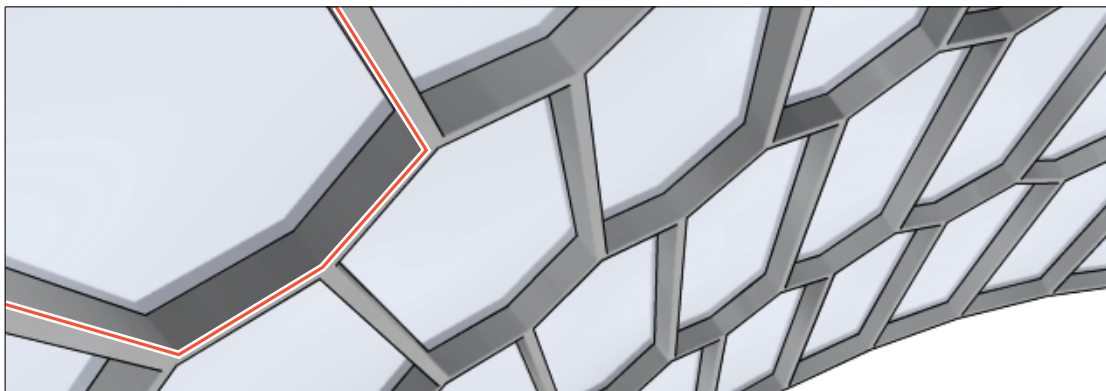


FIGURE 7.6: This construction (a detail of Figure 7.1) is based on an edge offset mesh and has beams of constant height. In the positively curved areas, edges (red) of beams with rectangular cross-section have an exact intersection at the nodes, which follows from Prop. 7.3.

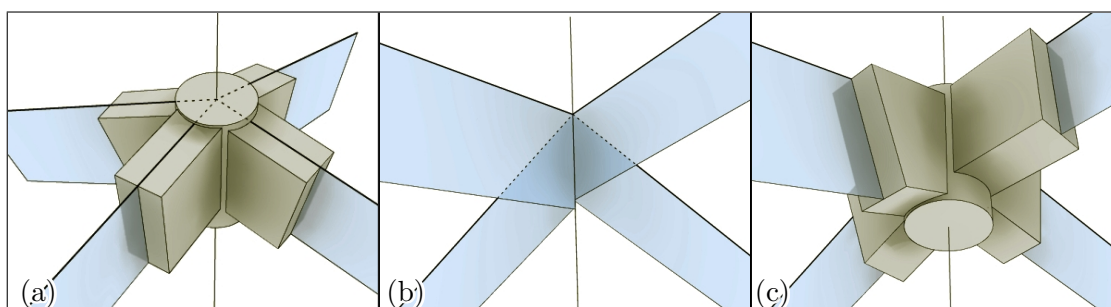


FIGURE 7.7: This geometric support structure is defined by two parallel meshes which are not of constant edge-edge distance. We nevertheless employ beams of constant height to physically realize that support structure. The resulting misalignment is not visible from the outside (a), hardly visible from the beams' mid-sections lying in the respective central planes (b), but is clearly visible from the inside (c). Still, nodes have no torsion and symmetry planes of beams intersect in the node axis.

mesh, then beams of constant height are perfectly aligned on both the upper (outer) and the lower (inner) side of the construction, provided that the mesh is convex (see Figure 7.6).

If, on the other hand, the distance of corresponding edges in \mathcal{M} and \mathcal{M}' is not the same throughout the mesh, then one could still use beams with the same cross-section throughout the mesh, but these beams will not be aligned on both sides (see Figure 7.7). Still, if \mathcal{M} and \mathcal{M}' are at almost constant edge-edge distance, a physical realization may use beams of the same cross-section throughout the mesh, without too much misalignment being visible. As exact distance requirements are sometimes hard or impossible to fulfill, we consider offsets at an exactly constant distance (in this section) as well as offsets at approximately constant distance (in Section 7.4).

7.3.1 Types of exact offset meshes

Recall that a mesh $\mathcal{M}' \in \mathcal{P}(\mathcal{M})$ at constant distance from \mathcal{M} is an *offset* of \mathcal{M} . Different ways to define the precise meaning of “ $\text{dist}(\mathcal{M}, \mathcal{M}') = d$ ” lead to different kinds of offsets:

- *vertex offsets*: The distance of corresponding vertices $\mathbf{m}_i, \mathbf{m}'_i$ is independent of the vertex and equals a constant d .
- *edge offsets*: The distance of corresponding parallel edges (actually, lines which carry those edges) independent of the edge and equals d .
- *face offsets*: The distance of faces (actually, planes which carry faces) is independent of the face and equals d .

Discrete Gauss images. If \mathbf{p} is a point of a smooth surface and \mathbf{n} is the unit normal vector there, then $\mathbf{p}' = \mathbf{p} + d\mathbf{n}$ would be a point of an offset surface at distance d . If \mathbf{p}, \mathbf{p}' are given, we can recover the unit normal vector by $\mathbf{n} = (\mathbf{p}' - \mathbf{p})/d$. If \mathcal{M}' is an offset mesh of \mathcal{M} we can mimic this construction and define a discrete *Gauss image mesh* $\mathcal{S} := (\mathcal{M}' - \mathcal{M})/d$, whose vertices $\mathbf{s}_i = (\mathbf{m}'_i - \mathbf{m}_i)/d$ can be regarded as discrete normal vectors.

Proposition 7.2. *Consider a mesh \mathcal{M} , its offset mesh \mathcal{M}' at distance d , and define the Gauss image mesh $\mathcal{S} = (\mathcal{M}' - \mathcal{M})/d$. Then the following is true:*

1. \mathcal{M}' is a vertex offset of $\mathcal{M} \iff$ the vertices of \mathcal{S} are contained in the unit sphere S^2 . If \mathcal{S} is a quad mesh and no edges degenerate, then \mathcal{M} has a vertex offset if and only if \mathcal{M} is a circular mesh, i.e., each face has a circumcircle.
2. \mathcal{M}' is an edge offset of $\mathcal{M} \iff$ the edges of the Gauss image mesh \mathcal{S} are tangent to S^2 .

3. \mathcal{M}' is a face offset of \mathcal{M} \iff the faces of the Gauss image mesh \mathcal{S} are tangent to S^2 . A mesh has a face offset if and only if it is conical, i.e., the faces around a vertex are tangent to a cone of revolution.

So in all three cases we have the equivalence $\text{dist}(\mathcal{M}, \mathcal{M}') = d \iff \text{dist}(\mathcal{S}, \mathbf{o}) = 1$, which means that the vertices, or the edges, or the faces of \mathcal{S} are at distance 1 from the origin.

Proof. The equivalence $\text{dist}(\mathcal{M}', \mathcal{M}) = d \iff \text{dist}(\mathcal{S}, \mathbf{o}) = 1$ is elementary. The statements about circular and conical meshes are reviewed in [114]. \square

This relation between a pair of offset meshes $\mathcal{M}, \mathcal{M}'$ and the Gauss image mesh \mathcal{S} is illustrated by Figure 7.8. Proposition 7.2 has an important consequence: If the mesh \mathcal{M} has an offset mesh at constant vertex/edge/face distance, then *every* mesh parallel to \mathcal{M} has this property. This is because the Gauss image mesh $\mathcal{S} \in \mathcal{P}(\mathcal{M})$ can be used to construct an offset not only for \mathcal{M} , but for any further mesh in $\mathcal{P}(\mathcal{M})$. Another observation will be important later: We can first construct a mesh whose vertices/edges/faces are at distance 1 from the origin. Then any mesh $\mathcal{M} \in \mathcal{P}(\mathcal{S})$ has offset meshes $\mathcal{M}' = \mathcal{M} + d\mathcal{S}$.

7.3.2 Meshes with edge offsets

We are interested in meshes which have edge offsets (EO meshes) because they can be built with beams of constant height meeting at the nodes in a geometrically optimal way (see Figure 7.6). Proposition 7.2 mentioned that a mesh \mathcal{M} has an edge offset mesh,

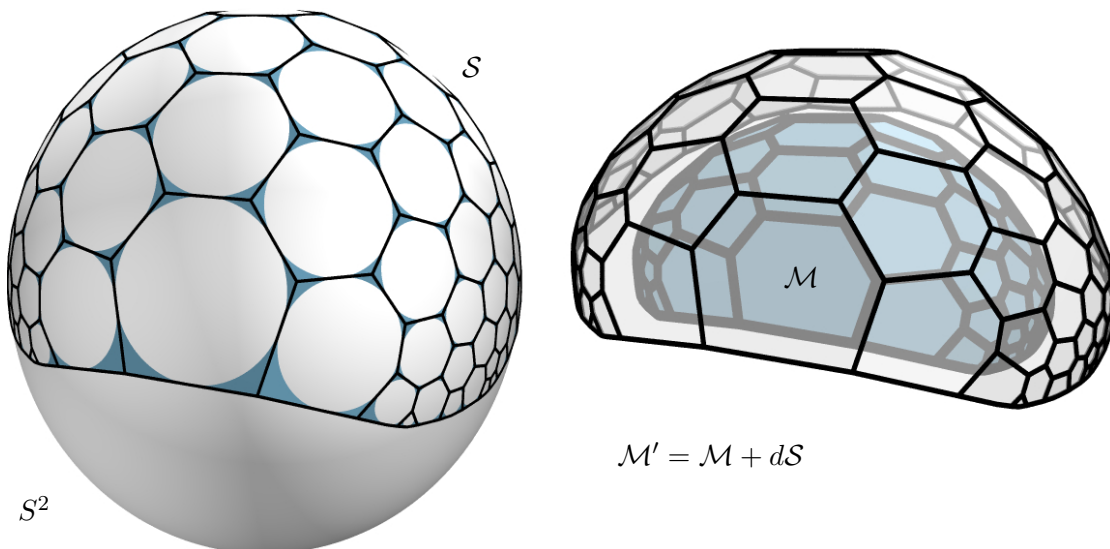


FIGURE 7.8: A mesh \mathcal{M} with an edge offset mesh \mathcal{M}' at distance d has a parallel mesh $\mathcal{S} = (\mathcal{M}' - \mathcal{M})/d$ whose edges are tangent to the unit sphere S^2 . The faces of \mathcal{S} intersect S^2 in a circle packing, cf. Section 7.3.2.

if there is a mesh \mathcal{S} parallel to \mathcal{M} whose edges are tangent to the unit sphere. The following paragraphs deal with the interesting mathematical theory of EO meshes, with a focus on the geometry of \mathcal{S} .

Proposition 7.3. *If a mesh \mathcal{M} has an edge offset \mathcal{M}' , then for each vertex \mathbf{m}_i of \mathcal{M} , the edges emanating from \mathbf{m}_i are contained in a cone of revolution Γ_i . The node axis A_i spanned by corresponding vertices $\mathbf{m}_i \in \mathcal{M}$, $\mathbf{m}'_i \in \mathcal{M}'$ is the axis of the cone Γ_i .*

Proof. The statement about cones is true for the mesh \mathcal{M} if and only if it is true for at least one mesh in $\mathcal{P}(\mathcal{M})$ which does not have zero edges (because corresponding edges are parallel). It is thus sufficient to show it for the Gauss image mesh $\mathcal{S} = (\mathcal{M}' - \mathcal{M})/d$, where $d = \text{dist}(\mathcal{M}, \mathcal{M}')$. According to Proposition 7.2, the edges of \mathcal{S} are tangent to the unit sphere S^2 (see Figure 7.8 and especially Figure 7.9). Obviously, all lines emanating from a vertex \mathbf{s}_i which touch S^2 lie in a cone of revolution $\tilde{\Gamma}_i$, so the statement is true for \mathcal{S} . Consequently it is true for \mathcal{M} . The axis of $\tilde{\Gamma}_i$ passes through the origin, so it is parallel to the vector \mathbf{s}_i . It follows that the axis A_i of the cone Γ_i associated with the vertex \mathbf{m}_i contains the point $\mathbf{m}'_i = \mathbf{m}_i + d\mathbf{s}_i$. \square

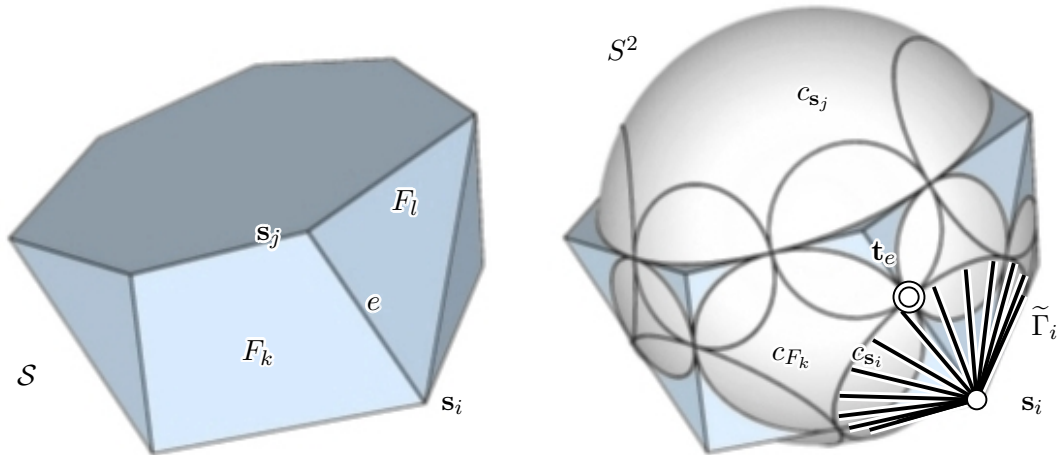


FIGURE 7.9: A Koebe polyhedron and related circles and cones.

EO meshes and Koebe polyhedra.

A mesh \mathcal{S} with planar faces whose edges e touch S^2 in points \mathbf{t}_e (a so-called *Koebe polyhedron*) has very interesting geometry [18, 160]. Each face F intersects S^2 in a circle c_F which touches the boundary edges of F from the inside (see Figures 7.8 and 7.9). For any vertex \mathbf{s}_i , the vertex cone $\tilde{\Gamma}_i$ touches the unit sphere in a circle $c_{\mathbf{s}_i}$. Obviously the edge e has a point of tangency \mathbf{t}_e with S^2 , and two circles of either type pass through \mathbf{t}_e . Circles of the same type touch each other, and circles of different types intersect at 90 degrees. The computation of such circle patterns via minimization of a convex function is known [18] and even possible on-line [135]. Closed Koebe polyhedra are uniquely determined by their combinatorics up to a Möbius transformation (i.e., a projective

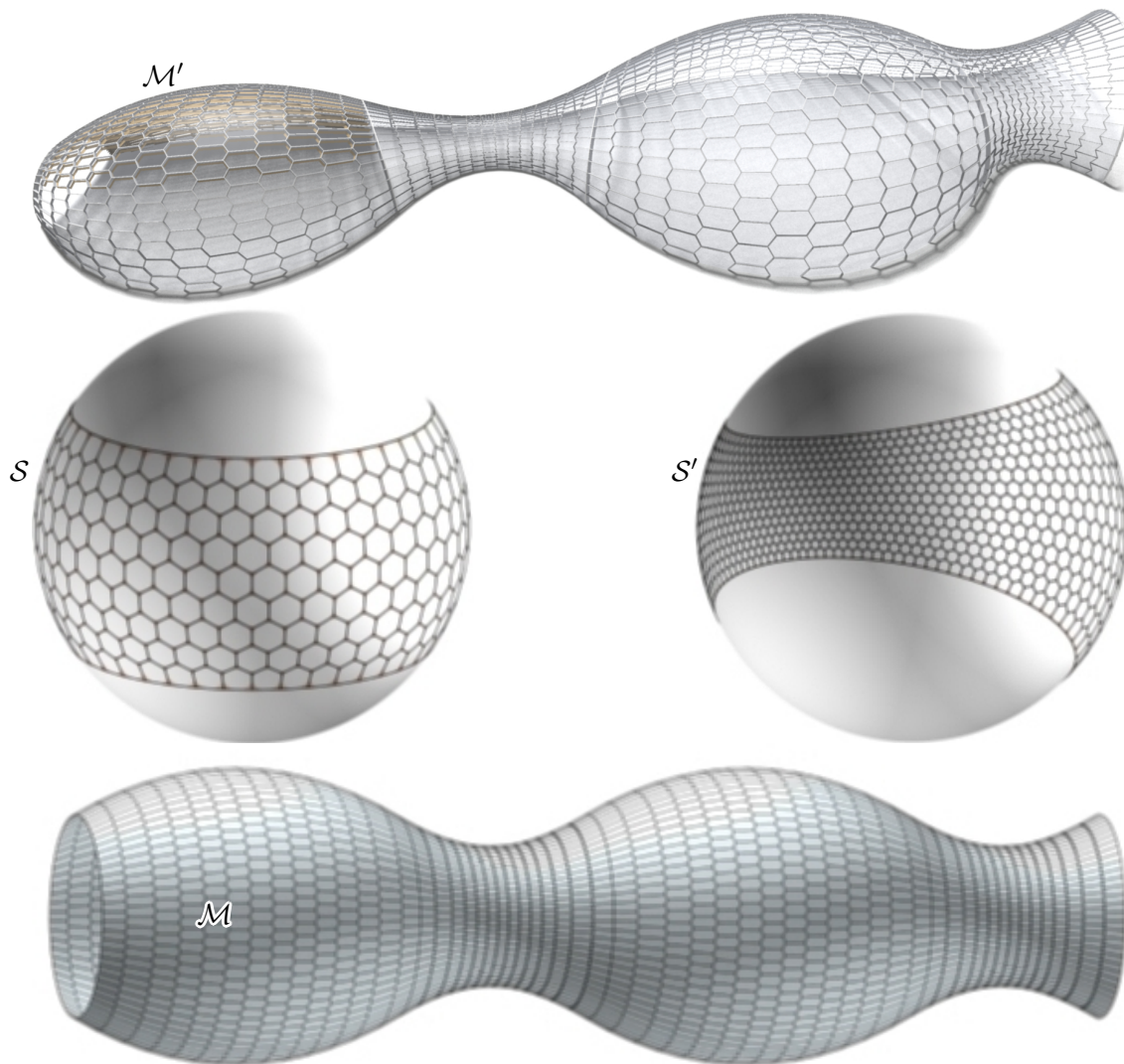


FIGURE 7.10: Creation of the mesh \mathcal{M}' which Figure 7.1 is based on. This example demonstrates that meshes with properties interesting from the mathematical viewpoint can yield aesthetically pleasing results; and that a designer has access to additional degrees of freedom by applying some nonstandard geometric transformations. Here we start with the Koebe polyhedron \mathcal{S} and construct the mesh \mathcal{M} which is of constant mean curvature with respect to \mathcal{S} (the Gauss mapping $\sigma : \mathcal{M} \rightarrow \mathcal{S}$ has an overfolding, with the inflection circle on \mathcal{M} corresponding to the apparent boundary of \mathcal{S}). Applying a Laguerre transformation results in the EO mesh \mathcal{M}' , which has the Gauss image mesh \mathcal{S}' . This L-transform was found interactively.

mapping which transforms S^2 into itself). For open polyhedra there is an additional degree of freedom for each boundary vertex.

7.3.3 Designing with EO meshes

The edge offset property is rather restrictive. Quad-dominant meshes which have vertex or face offsets (i.e., the circular and conical meshes), are capable of approximating arbitrary shapes. This is no longer the case with EO meshes.

Computing EO meshes from Koebe polyhedra.

We may use the following general procedure when designing a mesh \mathcal{M} with the edge offset property: First we determine the combinatorics of the mesh and compute a Koebe polyhedron \mathcal{S} with that combinatorics, using the procedure of [18]. The mesh \mathcal{M} we are looking for is then found within the space $\mathcal{P}(\mathcal{S})$, e.g. by optimization. An example of this is shown by Figure 7.11, where \mathcal{S} is a Koebe polyhedron with pentagonal faces and \mathcal{M} is found by minimizing the fairness functional $f_{Laplacian}$ defined by

$$f_{Laplacian}(\mathcal{M}) = \sum_{\text{vertices } \mathbf{m}_i} \left(\mathbf{m}_i - \frac{1}{\deg(\mathbf{m}_i)} \sum_{\mathbf{m}_j \in \text{star}(\mathbf{m}_i)} \mathbf{m}_j \right)^2, \quad (7.3)$$

under appropriate sign constraints for the factors λ_{ij} of Equation. (7.1) (see the figure caption for more details). Once an EO mesh is found, we may apply geometric transformations to it – this is the topic of the next paragraph. It is obvious that these methods are not useful for geometric modeling in the usual sense, but only for form finding purposes.

Laguerre transformations of EO meshes.

From the various equivalent descriptions of Laguerre geometry [28], the following, which employs the spheres of \mathbb{R}^3 as basic elements, is perhaps shortest: A sphere S with center (m_1, m_2, m_3) and signed radius r is identified with the point $\mathbf{x}_S = (m_1, m_2, m_3, r) \in \mathbb{R}^4$. We think of normal vectors of spheres pointing to the outside if and only if $r > 0$. Points are spheres of zero radius. An L -transformation then has the form $\mathbf{x}_S \mapsto A \cdot \mathbf{x}_S + \mathbf{a}$, where $\mathbf{a} \in \mathbb{R}^4$ and A is a 4×4 matrix with $A^T J A = J$ and $J = \text{diag}(1, 1, 1, -1)$. Every Euclidean transform permutes the set of spheres and can be written as an L-transform. Another simple example of an L-transform ($A = I_4$ and $\mathbf{a} = (0, 0, 0, d)$) is the offsetting operation which increases the radius by the value d . It is well known that the set of spheres tangent to an oriented cone of revolution is mapped by any L-transform α to a set of the same type [28]. Thus, an oriented cone of revolution Γ becomes an entity of Laguerre geometry: Take two spheres S_1, S_2 tangent to Γ and define $\alpha(\Gamma)$ to be tangent to the spheres $\alpha(S_1), \alpha(S_2)$. After these preparations we can state:

Proposition 7.4. *An L-transformation maps an edge offset mesh \mathcal{M} to another edge offset mesh \mathcal{M}' , if both are seen as the respective collection of vertex cones Γ_i, Γ'_i according to Proposition 7.3.*

We use Proposition 7.4 for the modification of edge offset meshes. An example is furnished by the mesh Figure 7.1 is based on; the transformation we use is illustrated by Figure 7.10.

Possible shapes of EO meshes (quad and hex mesh cases).

For a mesh \mathcal{M} with the edge offset property, the Gauss image mesh \mathcal{S} is a Koebe polyhedron. Bobenko et al. [22] show that in case of quad meshes, \mathcal{S} is a so-called s-isothermic mesh and thus the mesh \mathcal{M} is a discrete variant of a curvature line parameterization whose Gauss image is an isothermic curve network. Such “L-isothermic surfaces” are mentioned by Blaschke [16], but not much seems to be known about their shapes. Likewise, *the description of the possible shapes obtainable by quadrilateral EO meshes is an unsolved problem at the present time.*

Hexagonal meshes, which here is a synonym for meshes with planar faces and vertices of valence three, have better approximation properties (cf. e.g. [39]). In order to create a hexagonal EO Mesh \mathcal{M} which approximates a given surface Φ , we could start with any Koebe polyhedron \mathcal{S} which is a hexagonal mesh (cf. Figures 7.10 and 7.18) and determine the vertices \mathbf{m}_i of \mathcal{M} as follows: Parallel translate the three planes which are adjacent to the vertex \mathbf{s}_i in the mesh \mathcal{S} so that they touch Φ , and intersect them. By construction, \mathcal{M} has the edge offset property and the planes which carry its faces touch Φ . Unfortunately it is apparently difficult to guarantee that \mathcal{M} is a nice polyhedral surface without self-intersections, so much work remains to be done before such a procedure can be used as an effective design tool. We does not enter the topic of approximating general surfaces

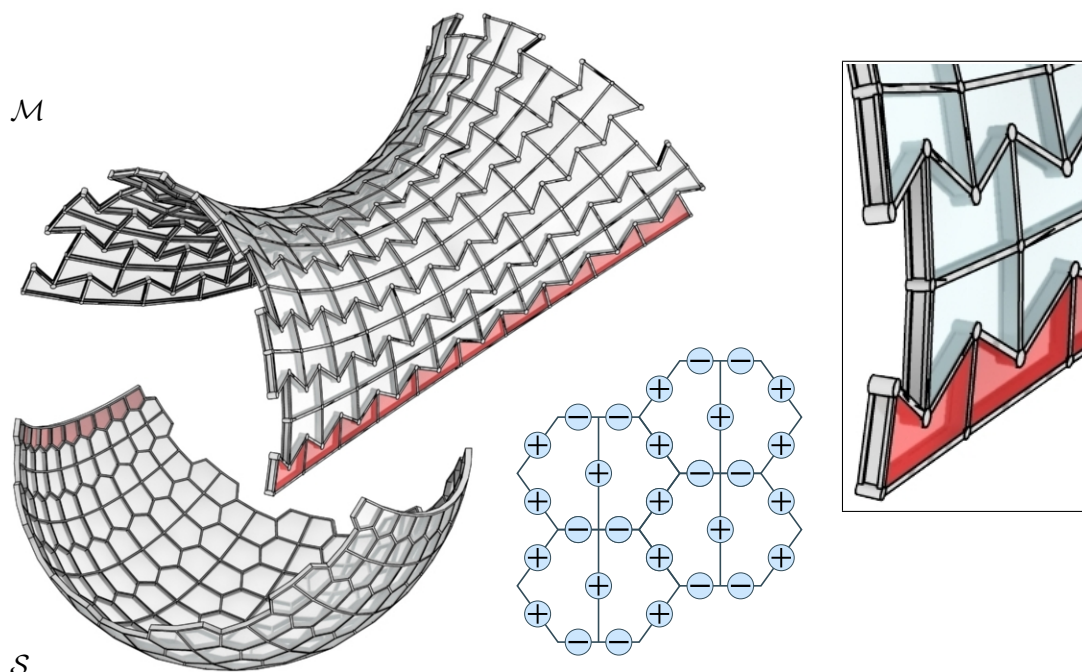


FIGURE 7.11: *Edge offset mesh of negative curvature.* The mesh \mathcal{M} with pentagonal faces has a parallel mesh \mathcal{S} whose edges are tangent to the unit sphere, so \mathcal{M} has the edge offset property. Corresponding edges of \mathcal{M} and \mathcal{S} are parallel, but the correspondence is orientation-reversing for some edges. The general pattern which edges keep their orientation and which do not is indicated by the schematic diagram. The mesh \mathcal{S} is a Koebe polyhedron; \mathcal{M} was found by minimizing the Laplacian energy of Equation (7.3) in the space $\mathcal{P}(\mathcal{S})$, under appropriate sign constraints on the coefficients λ_{ij} of Equation (7.1).

7.4 Optimizing Support Structures

In view of Proposition 7.2 we cannot expect a general mesh with planar faces which is neither circular nor conical to have vertex offsets or face offsets. The class of meshes with edge offsets is even more restricted, as discussed in Sections 7.3.2 and 7.3.3. Therefore the problem of constructing offsets at *approximately constant distance* is important. This section first discusses such approximate offset meshes from a theoretical viewpoint, and then shows how to compute them by minimizing a quadratic function in the space $\mathcal{P}(\mathcal{M})$. We also treat the problem of finding a mesh with planar faces which approximates a given shape in the first place. This completes the processing pipeline from shape to mesh, and further to support structure. The last part of this section deals with more complex optimization problems.

7.4.1 Approximate offsets

The mesh \mathcal{M}' which is parallel to the mesh \mathcal{M} is said to be an *approximate offset* of \mathcal{M} , if $\mathcal{M}' = \mathcal{M} + d\mathcal{S}$, where $\mathcal{S} \in \mathcal{P}(\mathcal{M})$ is a mesh which approximates the unit sphere. We say that the distance of \mathcal{M}' from \mathcal{M} is approximately constant, and that \mathcal{S} is an approximate Gauss image of \mathcal{M} . As there are different kinds of (exact) Gauss image anyway, we will drop the attribute ‘approximate’ and simply speak of a Gauss image. The vertex \mathbf{s}_i of \mathcal{S} corresponding to a vertex \mathbf{m}_i of \mathcal{M} is considered as an approximate normal vector for the vertex \mathbf{m}_i . Then the approximate offset at distance d has vertices $\mathbf{m}_i + d\mathbf{s}_i$, which is directly analogous to the previous notation (see Figure 7.12). We use the symbol $\sigma : \mathcal{M} \rightarrow \mathcal{S}$ for the natural correspondence between the meshes \mathcal{M} and \mathcal{S} (σ is the *Gauss mapping*). The *computation* of an approximate Gauss image for a given mesh with planar faces is discussed in Sections 7.4.2 and 7.4.3 below.

7.4.2 Offset meshes by optimization in $\mathcal{P}(\mathcal{M})$

We approach the problem of computing offsets at approximately constant distance as follows: For a given mesh \mathcal{M} we must find a mesh $\mathcal{S} \in \mathcal{P}(\mathcal{M})$ which approximates the unit sphere (\mathcal{S} is a Gauss image of \mathcal{M}). We can convert this problem into minimization of a quadratic functional: For each face F , we have its normal vector \mathbf{n}_F , and a vertex $\mathbf{m}_{i(F)} \in F$. For each vertex \mathbf{m}_i , we estimate a unit normal vector $\tilde{\mathbf{n}}_i$. The vertex of \mathcal{S} corresponding to the vertex \mathbf{m}_i is denoted by \mathbf{s}_i . We set up the functionals

$$f_{faces} = \sum_{\text{faces } F} (\mathbf{n}_F \cdot (\mathbf{s}_{i(F)} - \mathbf{n}_F))^2, \quad f_{vert} = \sum_{\text{vertices } \mathbf{m}_i} (\mathbf{s}_i - \tilde{\mathbf{n}}_i)^2,$$

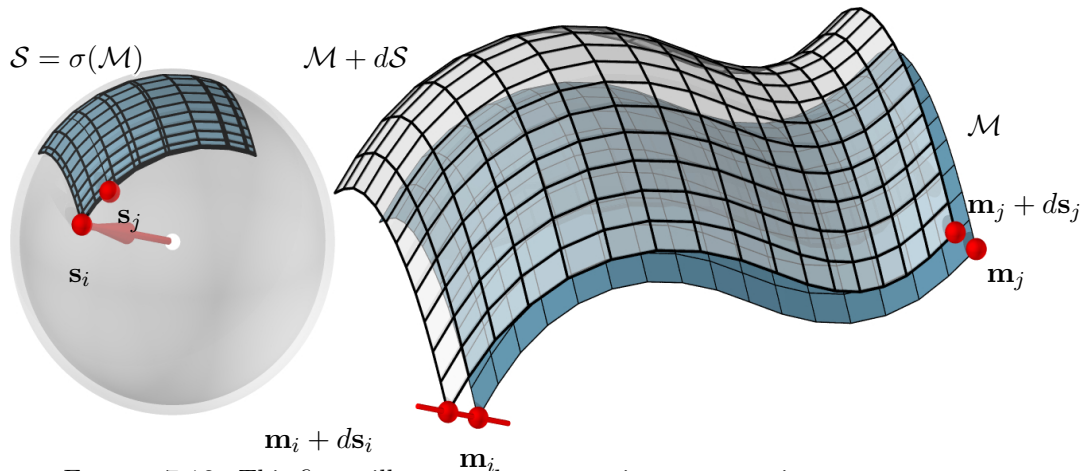


FIGURE 7.12: This figure illustrates how we assign a geometric support structure to a given mesh \mathcal{M} and thus make it buildable with optimized nodes and controlled beam heights. Optimization in the space $\mathcal{P}(\mathcal{M})$ yields a parallel mesh \mathcal{S} approximating the unit sphere S^2 , thus defining offsets $\mathcal{M}' = \mathcal{M} + d\mathcal{S}$ at approximately constant distance d . Here \mathcal{S} has an overfolding due to a change in the sign of curvature in \mathcal{M} , and is contained in the layer between radii 0.98 and 1.04.

and minimize a linear combination of f_{faces} , f_{vert} , and the fairness functional $f_{Laplacian}$ of Equation (7.3) (here each term of f_{faces} is the distance from \mathbf{s}_i to its parallel face and each term of f_{vert} is the difference from \mathbf{s}_i to the vertex normal). The aims $f_{vert} \rightarrow \min$ and $f_{faces} \rightarrow \min$ express the requirement that indeed the mesh \mathcal{S} is an (approximate) Gauss image of \mathcal{M} . Results are shown by Figures 7.12 and 7.13. Also the processing pipeline described in Section 7.4.3 uses this construction (cf. Figure 7.14). We may assign a low weight to the fairness term for \mathcal{S} , as a lack of fairness for \mathcal{S} is hardly noticeable in the support structure (Figure. 7.14d). It may be more important that beam heights are approximately constant.

7.4.3 A processing pipeline from shape to beam layout

Section 7.4.2 describes how we can find, for a given mesh \mathcal{M} with planar faces, an approximate offset, which can be used e.g. for the definition of beams of approximately constant heights. This section considers also the problem which in the overall processing pipeline comes earlier: How to find a mesh which has planar faces and approximates a given shape.

We compute a quad-dominant mesh with planar faces for a given surface by optimizing a mesh which follows a network of *conjugate curves* (of which Figures 7.14, a–c show some examples). Obviously the angle of intersection of such curves is important for the quality of the mesh, but for negatively curved surfaces it is not guaranteed that a network of conjugate curves has only transverse intersections. The special case of *principal curvature lines* has an intersection angle of 90° throughout the network. However, we

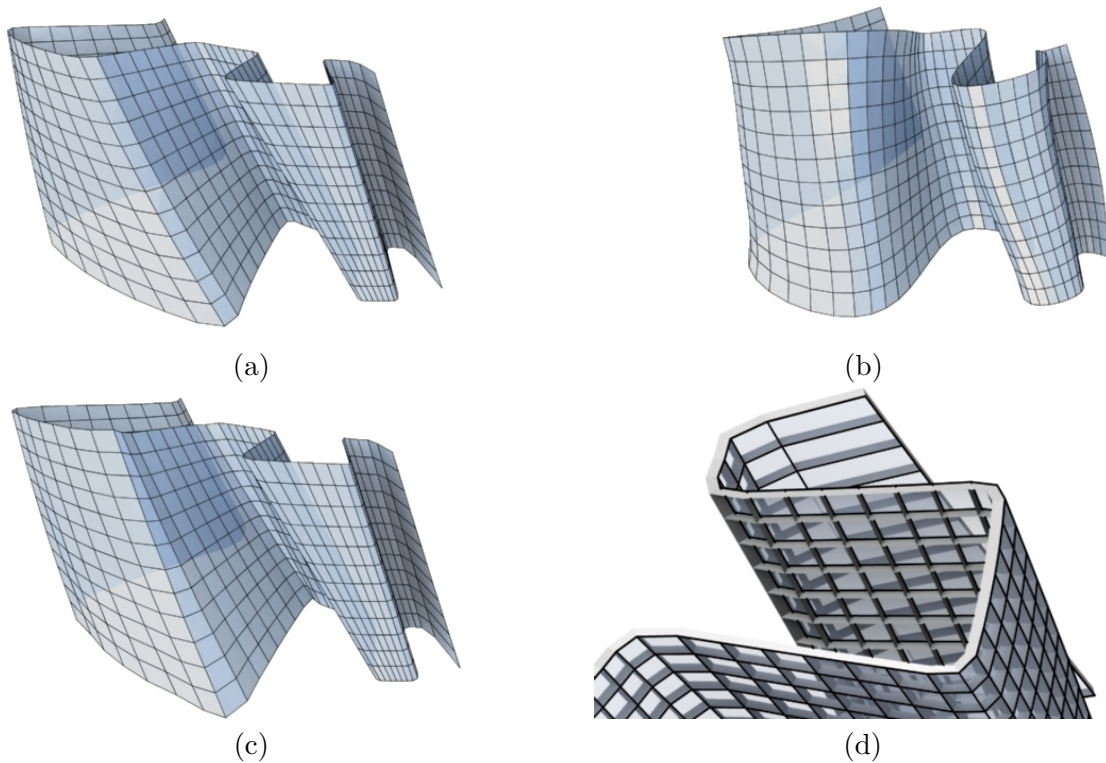


FIGURE 7.13: This figure illustrates that finding a support structure according to the method of Section 7.4.2 can be done with little change in the original mesh. (a) a mesh \mathcal{M} created by subdivision. (b) In order to create a geometric support structure with nice offset properties, we first make the mesh conical by perturbing vertices. As the intersection angles of mesh polylines in \mathcal{M} are far from 90 degrees, the optimized mesh deviates much from \mathcal{M} . (c) If \mathcal{M} is optimized to become planar, not necessarily conical, the deviations from the original mesh are small. (d) We apply our procedure for creating a support structure with approximately constant beam heights.

would like to have more degrees of freedom at our disposal when choosing the curve network.

Figure 7.14 illustrates our approach to this problem. We apply an affine transformation α to the given surface, compute principal curvature lines for the transformed surface, and transform them back with α^{-1} . Conjugacy of curves is invariant with respect to affine transformations, and the intersection angle of the resulting curves is close to 90° if α is close to the identity transformation. So we are able to create a number of suitable curve networks simply by choosing different affine mappings (see Figures 7.14, a–c).

We then lay out a quad-dominant mesh along a curve network which fits the design intent best, e.g. has few singularities, or has the singularities in the right places (we choose Figure 7.14c). This mesh is optimized such that its faces become planar (cf. Chapter 6). Having computed the mesh \mathcal{M} , we next need to find a support structure with beams of approximately constant height. This is done by finding a mesh $\mathcal{S} \in \mathcal{P}(\mathcal{M})$ which approximates S^2 , using the minimization procedure described in Section 7.4.2. A result is shown by Figure 7.14d. The geometric support structure which is bounded by the meshes \mathcal{M} and $\mathcal{M}' = \mathcal{M} + d\mathcal{S}$ is illustrated by Figure 7.14e.

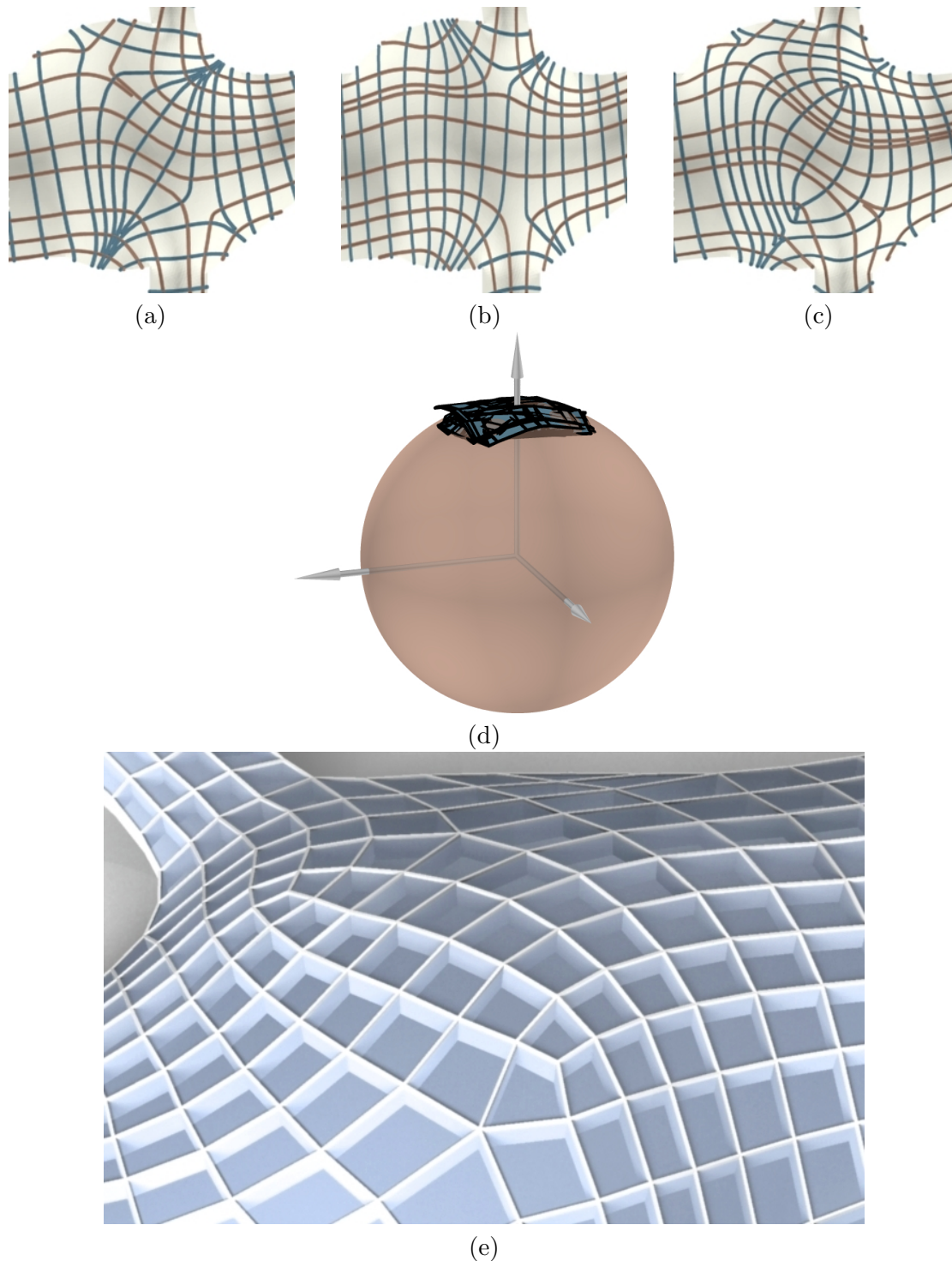


FIGURE 7.14: *Meshing and construction of a support structure with optimized nodes for a given architectural freeform design. (a)–(c). A study of different conjugate curve networks is performed. We lay out a quad mesh \mathcal{M} along the network in (c) and use the method in Chapter 6 to optimize \mathcal{M} such that its faces become planar. \mathcal{M} is shown in subfigure (e). (d) We recompute a Gauss image \mathcal{S} of \mathcal{M} which approximates the unit sphere S^2 . (e) \mathcal{S} leads to a support structure with optimized nodes and approximately equal beam heights for \mathcal{M} .*

7.4.4 Other ways of optimization

Combined optimization of mesh and Gauss image.

Section 7.4.2 discussed the problem of computing a geometric support structure – or an approximate Gauss image \mathcal{S} – for a given mesh \mathcal{M} , and in that section we described how to solve that problem by quadratic optimization in the space $\mathcal{P}(\mathcal{M})$. Here we go one step further and optimize both \mathcal{M} and \mathcal{S} at the same time.

The purpose of this computation is to design a mesh which has offset or curvature properties useful for architectural design. We know that any such mesh has to approximately follow a network of principal curves. We start with a mesh with this property and set up an optimization problem as follows. We consider the functionals $f_{close,1} := \sum_i \text{dist}(\mathbf{m}_i, \Phi)^2$ and $f_{close,2} := \sum_i \|\mathbf{s}_i\|^2$, which express proximity of the vertices of \mathcal{M}, \mathcal{S} to their respective reference surfaces Φ, S^2 ; further, the functional

$$f_{par} := \sum_{\text{edges } \mathbf{m}_i \mathbf{m}_j} \left\| \frac{\mathbf{m}_i - \mathbf{m}_j}{\|\mathbf{m}_i - \mathbf{m}_j\|} \times (\mathbf{s}_i - \mathbf{s}_j) \right\|^2,$$

which expresses parallelity of meshes \mathcal{M} and \mathcal{S} ; the fairness functionals $f_{Laplacian}(\mathcal{M})$, $f_{Laplacian}(\mathcal{S})$ according to Equation (7.3); and the functional f_{det} which expresses planarity of the mesh \mathcal{M} . We then use a penalty method to minimize a linear combination of the functionals above. Other functionals may be included. An example which includes a functional aiming at constant mean curvature is presented in Section 7.5 (Figure 7.19).

Methods of optimization and numerics.

The solution of the nonlinear optimization problems which arise when minimizing the functionals above, under the side condition of planarity of faces and parallelity of meshes, usually is difficult. We have employed a penalty method analogous to the mesh optimization procedure described in Chapter 6. In order to minimize a functional F under k different constraints $G_1 = 0, \dots, G_k = 0$, we consider the auxiliary minimization problem $\lambda F + \sum \mu_j G_j^2 \rightarrow \min$: For stable convergence, λ has a higher value at the beginning, and tends to zero as optimization progresses. The unconstrained minimization problem $\lambda F + \sum \mu_j G_j^2 \rightarrow \min$ can effectively be solved by the Gauss-Newton method with LM regularization [74]. In our case, F is a linear combination of $f_{close,1}$, $f_{close,2}$, $f_{Laplacian}(\mathcal{M})$, $f_{Laplacian}(\mathcal{S})$, while the constraints are given by f_{det} and f_{par} . This method usually requires user interaction when balancing the weights of the individual functionals.

Translating the Gauss image.

We want to mention a simple fact which nevertheless has an interesting application: If \mathcal{S} is a Gauss image for the mesh \mathcal{M} , then formally any translate $\mathcal{S}' = \mathcal{S} + \mathbf{x}$ is a valid

Gauss image, simply because it was never specified how well \mathcal{S}' must approximate the unit sphere, and meshes \mathcal{S}' and \mathcal{M} are parallel. It is a different question if we can still call the vertices of \mathcal{S}' *normal vectors* of the polyhedral surface \mathcal{M} without violating geometric intuition, but if \mathbf{x} is small, we surely can (see Figure. 7.15).

Figure 7.1 shows an architectural design based on a mesh \mathcal{M} and *two different* geometric support structures: One is defined by a Gauss image mesh \mathcal{S} whose edges are tangent to S^2 . It is used to create the supporting beams (then of constant height). The other one is based on the Gauss image $\mathcal{S}' = \mathcal{S} + \mathbf{x}$, and is physically realized as shading elements. The vector \mathbf{x} has been found by subjecting it to optimization: We select parallel light (for particular sun positions) and compute \mathbf{x} such that the total area of shadow cast by the shading elements is maximal, under the constraint $\|\mathbf{x}\| \leq 0.99$.

7.5 Curvatures in Meshes with Planar Faces

It has been observed many times that properties of smooth or discrete surfaces which are interesting from the mathematical viewpoint often lead to very aesthetic figures [139]. This is even more important in our current work which focuses on architecture and design. Therefore the aim of this section on curvatures is not only to create geometric functionals useful for smoothing and other optimization tasks, but to lay the foundations for interactive design and form finding tools. We want to demonstrate that the earlier developed theory, namely offset properties and support structures based on parallelism, is compatible with the second one, namely the definition of curvatures based on mesh parallelism. Not only they are compatible, but surfaces of constant or vanishing mean curvature, as well as other special surfaces, serve as basis of architectural designs with functional properties.

Preparation: Mixed areas and Steiner’s formula.

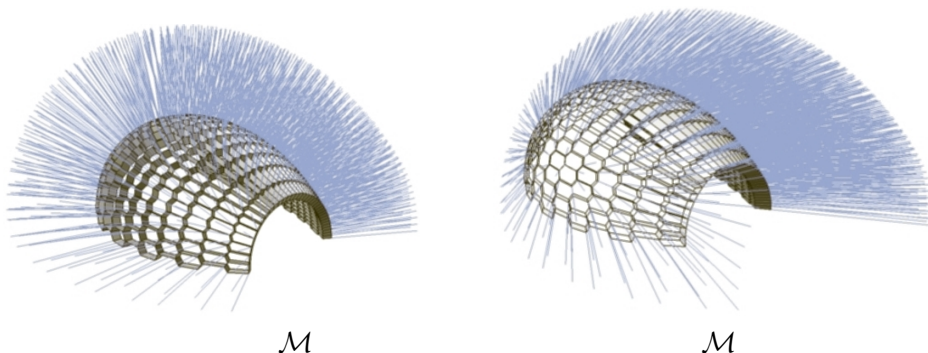


FIGURE 7.15: These two systems of ‘approximate normal vectors’ of a mesh \mathcal{M} are defined by a Gauss image \mathcal{S} , whose edges are tangent to S^2 (left image), and the Gauss image $\mathcal{S} + \mathbf{x}$ (right image). They correspond to the two different support structures employed in Figure. 7.1.

Assume that $P = (\mathbf{p}_0, \dots, \mathbf{p}_{k-1})$ and $Q = (\mathbf{q}_0, \dots, \mathbf{q}_{k-1})$ are planar polygons whose corresponding edges are parallel ('parallel polygons'). Then also the polygon $P + dQ = (\mathbf{p}_0 + d\mathbf{q}_0, \dots)$ is parallel to P and Q . We are interested in the oriented area of $P + dQ$, with the orientation defined by a normal vector \mathbf{n} of the plane which contains P . Its computation is based on the formula $\frac{1}{2} \det(\mathbf{b} - \mathbf{a}, \mathbf{c} - \mathbf{a}, \mathbf{n})$ for the area of a triangle with vertices $\mathbf{a}, \mathbf{b}, \mathbf{c}$, lying in a plane with unit normal vector \mathbf{n} . It follows that $\text{area}(P + dQ) = \frac{1}{2} \sum_{i=0}^{k-1} \det(\mathbf{p}_i + d\mathbf{q}_i, \mathbf{p}_{i+1} + d\mathbf{q}_{i+1}, \mathbf{n})$, with indices modulo k . This implies that

$$\begin{aligned} \text{area}(P + dQ) &= \text{area}(P) + 2d \text{area}(P, Q) + d^2 \text{area}(Q), \text{ with} \\ \text{area}(P, Q) &= \frac{1}{4} \sum_{i=0}^{k-1} [\det(\mathbf{p}_i, \mathbf{q}_{i+1}, \mathbf{n}) + \det(\mathbf{q}_i, \mathbf{p}_{i+1}, \mathbf{n})]. \end{aligned} \quad (7.4)$$

The computation so far is well known – $\text{area}(P, Q)$ denotes the *mixed area* of polygons P and Q in the terminology of convex geometry [132].

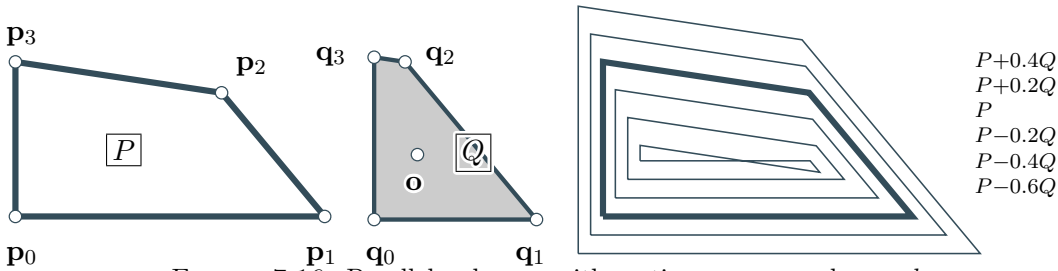


FIGURE 7.16: Parallel polygons with vertices \mathbf{p}_i , \mathbf{q}_i , and $\mathbf{p}_i + d\mathbf{q}_i$.

Curvatures of faces.

Curvatures in polyhedral surfaces can be defined in different ways. One may be guided by the idea that a discrete surface approximates a smooth one, and define a curvature by way of numerical differentiation. Another method is to observe relations between curvatures and geometric properties in smooth surfaces, and to postulate an analogous relation for the discrete case, like in the definition of the mean curvature vector by Polthier [108] as the gradient of the area functional for triangle meshes. In our setting we consider the variation of surface area when passing from a surface Φ to an offset surface Φ^d : Each point $\mathbf{x} \in \Phi$ is moved to $\mathbf{x} + d\mathbf{n}(\mathbf{x})$, where “ \mathbf{n} ” is the field of unit normal vectors. Then the surface area changes according to

$$\text{area}(\Phi^d) = \int_{\Phi} (1 - 2dH(\mathbf{x}) + d^2K(\mathbf{x})) dx, \quad (7.5)$$

with H as mean and K as Gaussian curvature (Steiner’s formula). In the discrete case the change in area exhibits a quite similar behavior:

Proposition 7.5. *The surface area of the (approximate) offset $\mathcal{M}^d = \mathcal{M} + d\mathcal{S}$ of the mesh \mathcal{M} w.r.t. to the Gauss image mesh $\sigma(\mathcal{M}) = \mathcal{S}$ obeys the law*

$$\text{area}(\mathcal{M}^d) = \sum_{F: \text{face of } \mathcal{M}} (1 - 2dH_F + d^2K_F) \text{area}(F), \text{ with} \quad (7.6)$$

$$H_F = -\frac{\text{area}(F, \sigma(F))}{\text{area}(F)}, \quad K_F = \frac{\text{area}(\sigma(F))}{\text{area}(F)}. \quad (7.7)$$

Here each face F of \mathcal{M} is oriented such that $\text{area}(F) > 0$.

Proof. It is sufficient to show (7.6) and (7.7) for a single face F . This follows directly from (7.4) by comparing coefficients. \square

When we compare Equations (7.5) and (7.6), we see that it is natural to *define mean curvature and Gaussian curvature of the face F by the quantities H_F and K_F given by (7.7)*. This definition of curvatures does not refer to the mesh \mathcal{M} alone, but implicitly assumes that the Gauss image mesh \mathcal{S} is given. The values of H_F and K_F behave exactly like they should also in other respects. One is that the Gaussian curvature is the quotient of areas between Gauss image and original surface, as in the smooth case. Another one is that for most faces F , meaningful principal curvatures $\kappa_{1,F}$, $\kappa_{2,F}$ can be defined, such that

$$H_F = (\kappa_{1,F} + \kappa_{2,F})/2, \quad K_F = \kappa_{1,F}\kappa_{2,F}.$$

The curvatures $\kappa_{1,F}$, $\kappa_{2,F}$ are the roots of the polynomial $f(x) = x^2 - 2H_Fx + K_F$; they exist if and only if

$$H_F^2 - K_F \geq 0.$$

We do not give details, but it is not difficult to show that this inequality is true whenever the face F , or its Gauss image $\sigma(F)$ is strictly convex. This follows from Minkowski's first inequality "area(P, Q)² - area(Q) area(P) ≥ 0 ", which applies when both P, Q are convex [132].

Meshes of constant mean curvature.

Discrete surfaces of constant mean curvature H_F (*cmc meshes*) or discrete *minimal* surfaces, which have $H_F = 0$, are interesting not only from the purely mathematical viewpoint, but also from the viewpoint of aesthetics. The condition that a mesh \mathcal{M} has constant curvatures with respect to a Gauss image mesh \mathcal{S} is not as rigid as one might expect, and it is possible to construct a great variety of such meshes (both quadrilateral and hexagonal), even meshes with the edge offset property.

First we discuss minimal meshes. The condition of minimality means that for all faces F , we have $H_F = 0$, and consequently the parallel polygons F and $\sigma(F)$ have vanishing

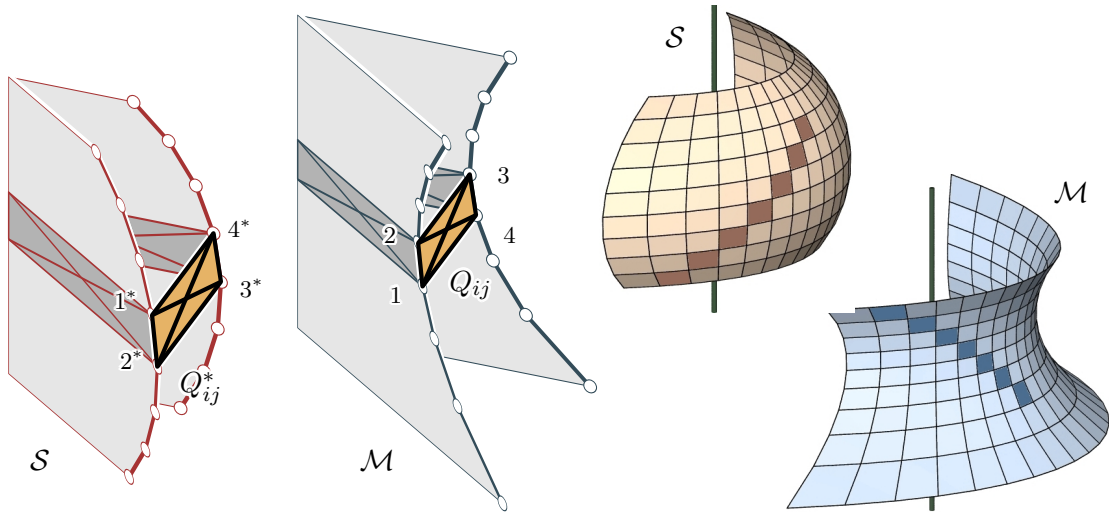


FIGURE 7.17: Construction of simple minimal quad meshes via parallelism of diagonals. The meshes \mathcal{M}, \mathcal{S} carry ‘horizontal’ polylines in horizontal planes and ‘meridian polylines’ in planes through a fixed axis. \mathcal{S}, \mathcal{M} are parallel – note corresponding faces Q_{ij} and Q_{ij}^* – but the correspondence is orientation-reversing. The mean curvature of the face Q_{ij} in the mesh \mathcal{M} with respect to the Gauss image \mathcal{S} vanishes if and only if diagonals in Q_{ij} are parallel to diagonals in Q_{ij}^* ($13 \parallel 2^*4^*$ and $24 \parallel 1^*3^*$).

mixed area. It is easy to show that in the case of parallel *quadrilaterals* $Q = 1234$ and $Q^* = 1^*2^*3^*4^*$, we have

$$\text{area}(Q, Q^*) = 0 \iff 13 \text{ parallel } 2^*4^*, 24 \text{ parallel } 1^*3^* \quad (7.8)$$

(see Figure 7.17). It is worth noting that this parallelism of diagonals in corresponding quads also occurs in the Christoffel duality constructions of [17] and [22], which has gone unnoticed so far. For this reason we would like to call parallel meshes *Christoffel transforms* of each other, if corresponding faces have vanishing mixed area.

We should note that not every mesh \mathcal{S} has the property that there exists a mesh \mathcal{M} which is minimal with respect to \mathcal{S} as Gauss image. In the following we therefore restrict ourselves to special cases.

Example: minimal and cmc quad meshes of simple geometry. For quad meshes \mathcal{M} and \mathcal{S} of ‘generalized rotational symmetry’ as described by Figure 7.17, we can construct a minimal mesh \mathcal{M} for given \mathcal{S} by starting with one vertex, say the one denoted by “1”, and computing the faces of \mathcal{M} step by step. They are uniquely determined by the requirements of parallelism of edges and parallelism of diagonals. The construction of a cmc mesh \mathcal{M} from \mathcal{S} is quite analogous to the minimal surface case; instead of the condition $H_F = 0$ we have now $H_F = \text{const}$ (we omit the details) Surfaces of revolution where the vertices of the Gauss image mesh \mathcal{S} lie in the unit sphere (and therefore \mathcal{M} is a circular mesh), have been considered by [64].

Example: Hexagonal meshes of rotational symmetry which have vanishing or constant mean curvature. The previous example concerning quad meshes extends to hex meshes

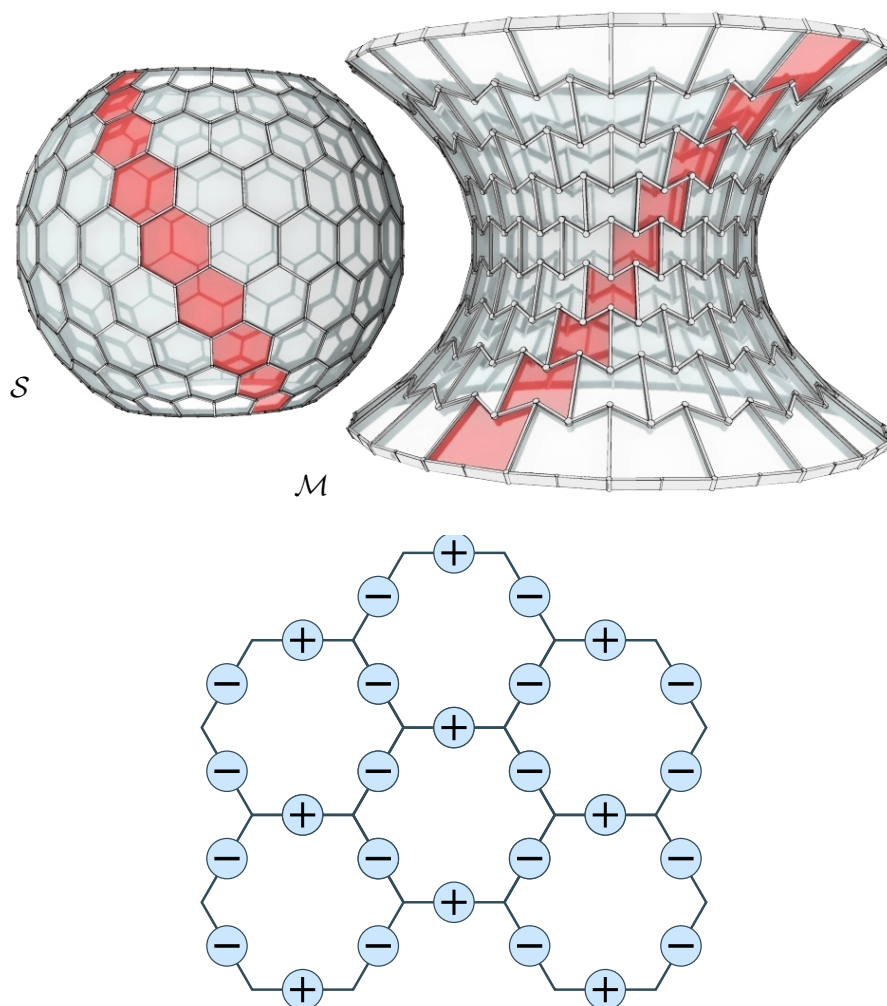


FIGURE 7.18: A *discrete minimal EO mesh*. The mesh \mathcal{M} is constructed from the Koebe polyhedron \mathcal{S} by the conditions that \mathcal{M} and \mathcal{S} are parallel meshes, and that the mixed areas of all corresponding faces vanish. Both \mathcal{M} and \mathcal{S} have rotational symmetry. The correspondence between \mathcal{M} and \mathcal{S} is orientation-reversing for some edges; the sign pattern is schematically illustrated at right.

as well, if we restrict ourselves to meshes with rotational symmetry (see Figures 7.10, 7.18, and 7.19). We do not provide details here, because they are not difficult and would take up too much space. We only mention that by splitting symmetric hexagons into quads we can treat this case in a way very similar to the previous example.

7.6 Discussion

Limitations.

With highly nonlinear optimization problems, there is in general no guarantee that optimization achieves success and is not stuck in a local minimum. Therefore it is very important to know beforehand which meshes can be optimized towards the goal under consideration. E.g. if a quad-dominant mesh is to become planar by moving vertices

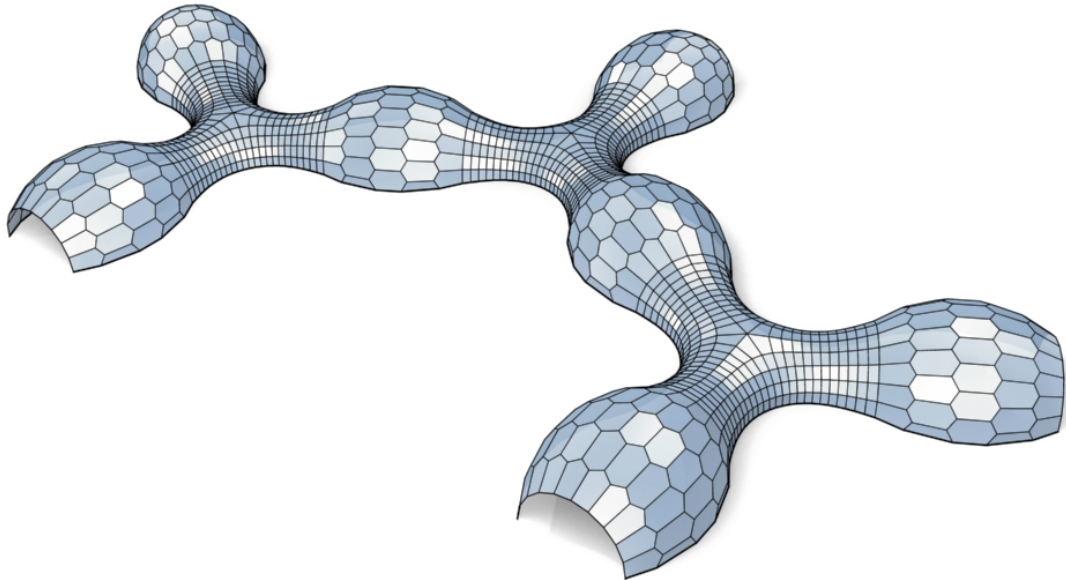


FIGURE 7.19: This design with convex faces is composed from pieces of discrete cmc surfaces obtained in different ways. The junction piece (quad mesh) was originally computed as a trinoid (cf. [25, 130]) by the applet at [129]. After combined optimization of mesh and Gauss image in order to achieve planarity of faces and $H_F = \text{const.}$, we arrive at a planar quad mesh which has $H_F \in [0.966, 1.048]$. The bulging pieces are hexagonal EO meshes whose mean curvature with respect to a previously chosen Koebe polyhedron as Gauss image is exactly constant.

as little as possible, then this mesh must originally have been aligned with a conjugate network of curves. We did not experience problems when the original mesh was chosen appropriately. However, this issue is very important for applications in practice. We also emphasized on easier optimization tasks like optimization in $\mathcal{P}(\mathcal{M})$ for computing a support structure, which exhibit quite tame behaviour.

The complexity of the modeling task shown by Figure 7.14 is rather high. It is hard to satisfy all design requirements if the underlying reference surface is not very smooth. This problem becomes even more severe if boundary conditions have to be met. As a consequence it would be difficult to find a geometrically optimal support structure for data sets like the Stanford bunny, for instance. Fortunately, architectural designs tend to be smoother.

Implementation and run times.

The most computationally expensive tasks in our work are nonlinear optimization procedures, for which we employed a Gauss-Newton method, and computing a basis of $\mathcal{P}(\mathcal{M})$ which is done by SVD. SVD runs well even if it needs a long run time, because we estimate the dimension of $\mathcal{P}(\mathcal{M})$ beforehand. The run times of code on a 2 GHz PC with 1 GB RAM are as follows: Computation of principal curves in Figure. 7.14a–c and meshing costs 25 seconds each. The resulting mesh \mathcal{M} has 649 vertices and 568 faces. Planarization costs 3 seconds, and a basis of $\mathcal{P}(\mathcal{M})$ is computed with SVD in 4.4 minutes; the Gauss image of Figure. 7.14d takes 0.68 seconds to compute. As to Figure.

7.19, simultaneous optimization of the mesh and its Gauss image towards $H = \text{const.}$ needs 3 minutes.

Conclusion.

We have presented mesh parallelism as a basic method for the solution of problems which occur in the design and construction of architectural freeform structures. It allows us to find an optimized beam layout with torsion free nodes, even after the design phase when we are just given a mesh with planar faces. Moreover, it is a key tool for modeling meshes with special offset properties. We introduced the new class of edge offset meshes which yield the cleanest possible nodes, if built with beams of constant height. Our results apply to EO quad meshes as well as to the more flexible pentagonal and hexagonal meshes. As a contribution to aesthetic design and a component for geometric optimization algorithms, we introduced a novel discrete curvature theory which is based on parallel meshes. In our examples we have pointed to invariance under certain transformations, which turned out to be of great practical value (blending of meshes, Laguerre transformations).

Claim: the presented material in this chapter has been published in [121].

Chapter 8

Conclusion and Future Research

8.1 Principal Contributions

In this thesis, we focus on the optimal shape reconstruction in curve and surface fitting and registration, and geometry modeling for architecture. The contributions of the present work are as follows:

- A thorough and systematic study of least squares orthogonal distance fitting of parametric curves and surfaces is presented. Effective and efficient methods based on local geometric properties have been proposed and analyzed in the numerical optimization framework. The proposed and improved methods including SDM, TDM, CDM and GTDM have super-linear convergence and have been successfully applied in 2D/3D, high-dimensional curve and surface fitting and constrained 3D shape reconstruction. The detailed discussion and comparison between PDM and other type methods from the point of view of geometry and optimization explain the poor behavior of PDM, although, which is widely used. Our proposed methods and easy implementation will benefit many industrial applications;
- We prove that the piecewise CVT function is C^2 in a 2D/3D convex region and accelerate the CVT computation by applying quasi-Newton methods that are much more efficient than Lloyd's method both in convergence rate and computation time.
- The introduced conical meshes and edge-offset meshes enrich the research of discrete differential geometry and demonstrate their superiority over other types of meshes, such as triangular meshes for architecture design and other applications where planarity and exact offset properties are demanded. The proposed PQ perturbation algorithms with subdivision are powerful tools for computing PQ meshes and developable meshes;

- The concept of mesh parallelism is an elegant approach to modeling offsets, multi-layer structures and geometrically optimized nodes. It provides a unified computational framework for circular meshes, conical meshes and edge-offset meshes. A discrete theory of curvature for meshes with planar faces analogous to the small theory is developed based on mesh parallelism and is very useful and powerful to generate discrete minimal surfaces, CMC surfaces in polyhedral meshes (quadrilateral, pentagonal and hexagonal meshes);

8.2 Future Research

There are many interesting problems which are worth studying. Some of these problems have been addressed in preceding chapters.

- We have explored the relationship between the optimization technique and geometrical error in least squares orthogonal distance fitting of parametric curves and surfaces, a natural extension of this work is to study the geometry behind methods for implicit curve/surface fitting. It is interesting to see how the local differential geometrical properties of implicit curve/surface are involved in the optimization methods, and find some geometric variation to simplify optimization methods and speed up the optimization;
- The generation of conical meshes and edge-offset meshes rely on the initial input meshes. The quality of the initial meshes affects the final result. For instance, the size and shape of planar faces, the shape of curve network affect the aesthetics, thus it would be very helpful to design modeling tools to construct and modify the initial meshes interactively, and derive good criterion to characterize aesthetical properties;
- Polyhedral meshes including quadrilateral meshes, hexagonal meshes have good potential in future geometric design and are deserved to have more attention. Some possible research topics include polyhedral mesh approximation, the shape control of polyhedral faces, discrete differential geometry and finite element analysis on polyhedral meshes, static equilibrium of polyhedral structures for architecture construction.

Bibliography

- [1] S. J. Ahn. *Least Squares Orthogonal Distance Fitting of Curves and Surfaces in Space*, volume 3151 of *Lecture Notes in Computer Science*. Springer, 2004.
- [2] E. Akleman, V. Srinivasan, and E. Mandal. Remeshing schemes for semi-regular tilings. In *Shape Modeling and Applications, Proceedings*, pages 44–50. IEEE, 2005. ISBN 076952379X.
- [3] P. Alliez, D. Cohen-Steiner, O. Devillers, and B. Levy and M. Desbrun. Anisotropic polygonal remeshing. *ACM Trans. Graphics (Siggraph 2003)*, 22(3):485–493, 2003.
- [4] P. Alliez, É. C. de Verdière, O. Desvillers, and M. Isenburg. Centroidal voronoi diagrams for isotropic surface remeshing. *Graphical Models*, 67(3):204–231, 2003.
- [5] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. *ACM Transactions on Graphics (Proceeding of SIGGRAPH 2005)*, 24(3):617–625, 2005.
- [6] L. Ambrosio and C. Montegazza. Curvature and distance function from a manifold. *Journal of Geometric Analysis*, 8(5):723–748, 1998.
- [7] Y. Asami. A note on the derivation of the first and second derivatives of objective functions in geographical optimization problems. *Journal of Faculty of Engineering, The University of Tokyo(B)*, XLI(1):1–13, 1991.
- [8] A. Atieg and G. A. Watson. A class of methods for fitting a curve or surface to data by minimizing the sum of squares of orthogonal distances. *Journal of Computational and Applied Mathematics*, 158:227–296, 2003.
- [9] G. Aumann. Degree elevation and developable Bézier surfaces. *Comp. Aided Geom. Design*, 21:661–670, 2004.
- [10] F. Aurenhammer. Voronoi diagrams – survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.
- [11] P. Benkő, G. Kos, T. Varady, L. Andor, and R. Martin. Constrained fitting in reverse engineering. *Computer Aided Geometric Design*, 19:173–205, 2002.

- [12] M. Bercovier and M. Jacob. Minimization, constraints and composite Bézier curves. *Computer Aided Geometric Design*, 11(5):533–563, 1994.
- [13] P. J. Besl and N. D. McKay. A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:239–256, 1992.
- [14] A. Bjorck. *Numerical Methods for Least Squares Problems*. Mathematics Society for Industrial and Applied Mathematics, Philadelphia, 1996.
- [15] A. Blake and M. Isard. *Active Contours*. Springer, 1998.
- [16] W. Blaschke. *Vorlesungen über Differentialgeometrie*, volume 3. Springer, 1929.
- [17] A. Bobenko and U. Pinkall. Discrete isothermic surfaces. *J. Reine Angew. Math.*, 475:187–208, 1996.
- [18] A. Bobenko and B. Springborn. Variational principles for circle patterns and Koebe’s theorem. *Trans. Amer. Math. Soc.*, 356:659–689, 2004.
- [19] A. Bobenko and Y. Suris. Discrete differential geometry. Consistency as integrability. monograph pre-published at <http://arxiv.org/abs/math.DG/0504358>, 2005.
- [20] A. Bobenko and Y. Suris. On organizing principles of discrete differential geometry, geometry of spheres. to appear. <http://arxiv.org/abs/math.DG/0608291>, 2006.
- [21] A. Bobenko, D. Matthes, and Y. Suris. Discrete and smooth orthogonal systems: C^∞ -approximation. *Int. Math. Res. Not.*, (45):2415–2459, 2003. ISSN 1073-7928.
- [22] A. Bobenko, T. Hoffmann, and B. A. Springborn. Minimal surfaces from circle patterns: geometry from combinatorics. *Annals of Mathematics*, 164:231–264, 2006.
- [23] J-D Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, 1998. ISBN 0521565294.
- [24] C. F. Borges and T. Pastva. Total least squares fitting of Bézier and B-spline curves to ordered data. *Computer Aided Geometric Design*, 19(4):275–289, 2002.
- [25] K. Große-Brauckmann and K. Polthier. Constant mean curvature surfaces derived from Delaunay’s and Wente’s examples. In *Visualization and mathematics (Berlin-Dahlem, 1995)*, pages 119–134. Springer, 1997.
- [26] S. Brell-Cokcan and H. Pottmann. Tragstruktur für Freiformflächen in Bauwerken. Patent No. A1049/2006, 2006.
- [27] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, 1995.

- [28] T. Cecil. *Lie Sphere Geometry*. Springer, 1992.
- [29] E. Cerda, S. Chaieb, F. Melo, and L. Mahadevan. Conical dislocations in crumpling. *Nature*, 401:46–49, 1999.
- [30] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.
- [31] K.-S. D. Cheng, W. Wang, H. Qin, K-Y K. Wong, H-P Yang, and Y. Liu. Design and analysis of optimization methods for subdivision surface fitting. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):878–890, 2007.
- [32] C. H. Chu and C. Sequin. Developable Bézier patches: properties and design. *Computer-Aided Design*, 34:511–528, 2002.
- [33] R. Cipolla and P. Giblin. *Visual Motion of Curves and Surfaces*. Cambridge University Press, 2000.
- [34] U. Clarenz, M. Rumpf, and A. Telea. Robust feature detection and local classification for surfaces based on moment analysis. *IEEE Trans. Visual. Comp. Graphics*, 10:516–524, 2004.
- [35] D. Cohen-Steiner and J.-M. Morvan. Restricted Delaunay triangulations and normal cycle. In *Proc. 19th annual symposium on Computational geometry*, pages 312–321. ACM, 2003. ISBN 1-58113-663-3.
- [36] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Trans. Graphics*, 23(3):905–914, 2004.
- [37] J. Cortés, S. Martínez, and F. Bullo. Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM: Control, Optimisation and Calculus of Variations*, 11(4):691–719, 2005.
- [38] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *Computer Graphics*, 30:303–312, 1996.
- [39] B. Cutler and E. Whiting. Constrained planar remeshing for architecture. In *Proc. Graphics Interface*. 2007.
- [40] M. Desbrun, E. Grinspun, and P. Schröder. *Discrete Differential Geometry*. SIGGRAPH Course Notes, 2005.
- [41] M. Djebali, M. Melkemi, and N. Sapidis. Range-image segmentation and model reconstruction based on a fit-and-merge strategy. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*, pages 127–138, 2002.

- [42] M. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.
- [43] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Comp. Aided Geom. Design*, 22:392–423, 2005.
- [44] Q. Du and M. Emelianenko. Acceleration schemes for computing centroidal voronoi tessellations. *Numerical Linear Algebra with Applications*, 13:173–192, 2006.
- [45] Q. Du and M. Gunzburger. Grid generation and optimization based on centroidal voronoi tessellations. *Applied Mathematics and Computation*, 133(2-3):591–607, 2002.
- [46] Q. Du and D. Wang. Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. *International Journal for Numerical Methods in Engineering*, 56(9):1355–1373, 2003.
- [47] Q. Du and D. Wang. Anisotropic centroidal voronoi tessellations and their applications.
- [48] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: applications and algorithms. *SIAM Review*, 41:637–676, 1999.
- [49] Q. Du, M. Gunzburger, and L. Ju. Constrained centroidal voronoi tessellations for surfaces. *SIAM J. SCI. COMPUT.*, 24(5):1488–1506, 2003.
- [50] Q. Du, M. Emelianenko, and L. Ju. Convergence of the lloyd algorithm for computing centroidal voronoi tessellations. *SIAM J. NUMBER. ANAL.*, 44:102–119, 2006.
- [51] M. J. B. Durrande. Questions résolues, démonstration du théorème de géométrie énoncé à la page 384 du V^e volume de ce recueil. *Ann. de math. pures appl.*, 6: 49–54, 1815.
- [52] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, New York, 4th edition, 1997.
- [53] R. B. Fisher. Applying knowledge to reverse engineering problems. *Computer-Aided Design*, 36:501–510, 2004.
- [54] D. R. Forsey and R. H. Bartels. Surface fitting with hierarchical splines. *ACM Transactions on Graphics*, 14:134–161, 1995.
- [55] W. Frey. Modeling buckled developable surfaces by triangulation. *Computer-Aided Design*, 36(4):299–313, 2004.

- [56] J. Glymph, D. Shelden, C. Ceccatoand, J. Mussel, and H. Schober. A parametric strategy for freeform glass structures using quadrilateral planar facets. In *Acadia 2002*, pages 303–321. ACM, 2002.
- [57] G. Golub and V. Pereyra. Separable nonlinear least squares: the variable projection method and its applications. *Inverse Problems*, 19:1–26, 2003.
- [58] A. A. Goshtasby. Grouping and parameterizing irregularly spaced points for curve fitting. *ACM Transactions on Graphics*, 19(3):185–203, 2000.
- [59] G. Greiner, A. Kolb, and A. Riepl. Scattered data interpolation using data dependant optimization techniques. *Graphical Models*, 64(1):1–18, 2002.
- [60] J. Haber, F. Zeilfelder, O. Davydov, and H. P.Seidel. Smooth approximation and rendering of large scattered data sets. In *Proceedings of Visualization'01*, pages 341–348, 2001.
- [61] H.P. Helfrich and D. Zwick. A trust region algorithm for parametric curve and surface fitting. *Journal of Computational and Applied Mathematics*, 73(1-2):119–134, 1996.
- [62] K. Hildebrandt, K. Polthier, and M. Wardetzky. On the convergence of metric and geometric properties of polyhedral surfaces. Technical Report 05-24, Zuse Institute Berlin, 2005.
- [63] M. Hofer, B. Odehnal, H. Pottmann, T. Steiner, and J. Wallner. 3D shape recognition and reconstruction based on line element geometry. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1532–1538, 2005.
- [64] T. Hoffmann. Discrete rotational CMC surfaces and the ellipticbilliard. In H. C. Hege and K. Polthier, editors, *Mathematical Visualization*, pages 117–124. Springer, 1998.
- [65] H. Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH'96*, pages 99–108, 1996.
- [66] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of SIGGRAPH'94*, pages 295–302, 1994.
- [67] J. Hoschek. Intrinsic parameterization for approximation. *Computer Aided Geometric Design*, 5(1):27–31, 1988.

- [68] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. AK Peters, 1993.
- [69] S. M. Hu and J. Wallner. second order algorithm for orthogonal projection onto curves and surfaces. *Computer Aided Geometric Design*, 22(3):251–260, 2005.
- [70] M. Iri, K. Murota, and T. Ohya. A fast voronoi-diagram algorithm with applications to geographical optimization problems. In *Proceedings of the 11th IFIP Conference*, pages 273–288, 1984.
- [71] H. Jin, T. Duchamp, H. Hoppe, J. A. McDonald, K. Pulli, and W. Stuetzle. Surface reconstruction from misregistered data. In *Proceedings of SPIE – Volume 2573, Vision Geometry IV*, pages 324–328, 1995.
- [72] D. Julius, V. Kraevoy, and A. Sheffer. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum (Proc. Eurographics 2005)*, 24(3):581–590, 2005.
- [73] A. Karniel, Y. Belsky, and Y. Reich. Decomposing the problem of constrained surface fitting in reverse engineering. *Computer-Aided Design*, 37:399–417, 2005.
- [74] C. T. Kelley. *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, 1999.
- [75] A. Kilian. *Design exploration through bidirectional modeling of constraints*. PhD thesis, Massachusetts Inst. Technology, 2006.
- [76] S-J Kim and M-Y Yang. Triangular mesh offset for generalized cutter. *Computer-Aided Design*, 37(10):999–1014, 2005.
- [77] R. Kunze, F-E Wolter, and T. Rausch. Geodesic voronoi diagrams on parametric surfaces. In *Computer Graphics International, 1997. Proceedings*, pages 230–237, 1997.
- [78] S. Lavalley and R. Szeliski. Recovering the position and orientation of free-form objects from image contours using 3d distance maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:378–390, 1995.
- [79] E. T. Y. Lee. Choosing nodes in parametric curve interpolation. *Computer-Aided Design*, 21(6):363–370, 1989.
- [80] G. Leibon and D. Letscher. Delaunay triangulations and voronoi diagrams for riemannian manifolds. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 341–349, 2000.
- [81] A. J. Lexell. *Acta Sc. Imp. Petr.*, 6:89–100, 1781.

- [82] Y. Liu, H-P Yang, and W. Wang. Reconstructing B-spline curves from point clouds – a tangential flow approach using least squares minimization. In *International Conference on Shape Modeling and Applications 2005*, pages 4–12, 2005.
- [83] Y. Liu, H. Pottman, J. Wallner, Y. Yang, and W. Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006)*, 25(3):681–689, 2006.
- [84] Y. Liu, H. Pottmann, and W. Wang. Constrained 3D shape reconstruction using a combination of surface fitting and registration. *Computer Aided Design*, 38(6): 572–583, 2006.
- [85] Yang Liu and Wenping Wang. A revisit to least squares orthogonal distance fitting of parametric curves and surfaces. In *Geometric Modeling and Processing - GMP 2008*, pages 384–397. Springer, 2008.
- [86] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.
- [87] D. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 1984.
- [88] W. Y. Ma and J. P. Kruth. Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces. *Computer-Aided Design*, 27(9):663–675, 1995.
- [89] K. Madsen, H. B. Nielsen, and O. Tingleff. Optimization with constraints. Lecture Notes, 2004.
- [90] I. Maekawa and K.H. Ko. Surface construction by fitting unorganized curves. *Graphical Models*, 64(5):316–332, 2002.
- [91] T. Maekawa. An overview of offset curves and surfaces. *Computer-Aided Design*, 31:251–267, 1999.
- [92] M. Marinov and L. Kobbelt. Direct anisotropic quad-dominant remeshing. In *Proc. Pacific Graphics*, pages 207–216, 2004.
- [93] R. R. Martin, J. de Pont, and T. J. Sharrock. Cyclide surfaces in computer aided design. In J. A. Gregory, editor, *The mathematics of surfaces*, pages 253–268. Clarendon Press, Oxford, 1986.
- [94] H. Meinhardt. *The Algorithmic Beauty of Sea Shells*. Springer, 1995.
- [95] J. Mitani and H. Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Trans. Graphics*, 23(3):259–263, 2004.

- [96] J. Montagnat and H. Delingette. A hybrid framework for surface registration and deformable models. In *Computer Vision and Pattern Recognition, CVPR'97*, pages 1041–1046, 1997.
- [97] D. J. Newman. The hexagon theorem. *IEEE Transactions on Information Theory*, 28:137–139, 1982.
- [98] Y. Nishikawa, H. Jinnai, T. Koga, T. Hashimoto, and S. T. Hyde. Measurements of interfacial curvatures of bicontinuous structure from three-dimensional digital images. 1. a parallel surface method. *Langmuir*, 14:1241–1249, 1998.
- [99] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.
- [100] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley, 1st edition, 1992.
- [101] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 165–176, 2006.
- [102] P. Oswald and P. Schröder. Composite primal/ dual $\sqrt{3}$ -subdivision schemes. *Comp. Aided Geom. Design*, 20:135–164, 2003.
- [103] N. Paragios and M. Rousson. Shape analysis towards model-based segmentation. In S. Osher and N. Paragios, editors, *Geometric Level Set Methods in Imaging, Vision and Graphics*, pages 231–250. Springer, 2003.
- [104] T. Pavlidis. Curve fitting with conic splines. *ACM Transactions on Graphics*, 2(1):1–31, 1983.
- [105] G. Peyré and L. Cohen. Surface segmentation using geodesic centroidal tessellation. In *2nd International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2004)*, pages 995–1002, 2004.
- [106] L. Piegl and W. Tiller. *The NURBS Book*. Springer, New York, 2nd edition, 1996.
- [107] M. Plass and M. Stone. Curve-fitting with piecewise parametric cubics. *Computer Graphics*, 17(3):229–239, 1983.
- [108] K. Polthier. *Polyhedral surfaces of constant mean curvature*. Habilitationsschrift TU Berlin, 2002.
- [109] K. Polthier. Unstable periodic discrete minimal surfaces. In *Nonlinear Partial Differential Equations*, pages 127–143. Springer, 2002.

- [110] I. R. Porteous. *Geometric Differentiation for the Intelligence of Curves and Surfaces*. Cambridge Univ. Press, 1994.
- [111] H. Pottmann and M. Hofer. Geometry of the squared distance function to curves and surfaces. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 223–244. Springer, 2003.
- [112] H. Pottmann and S. Leopoldseder. A concept for parametric surface fitting which avoids the parametrization problem. *Computer Aided Geometric Design*, 20:343–362, 2003.
- [113] H. Pottmann and Y. Liu. Discrete surfaces in isotropic geometry. In M. Sabin and J. Winkler, editors, *Mathematics of Surfaces XII*, volume 4647 of *LNCS*, pages 431–363, 2007.
- [114] H. Pottmann and J. Wallner. The focal geometry of circular and conical meshes. *Advances in Computational Mathematics*, 2008.
- [115] H. Pottmann and J. Wallner. *Computational Line Geometry*. Springer, 2001.
- [116] H. Pottmann, S. Leopoldseder, and M. Hofer. Approximation with active B-spline curves and surfaces. In *Proceedings of Pacific Graphics 2002*, pages 8–25. IEEE Computer Society Press, 2002.
- [117] H. Pottmann, S. Leopoldseder, and M. Hofer. Registration without icp. *Computer Vision and Image Understanding*, 95:54–71, 2004.
- [118] H. Pottmann, Q.-X. Huang, and Y.-L. Yang and S. Kölbl. Integral invariants for robust geometry processing. Geometry Preprint 146, TU Wien, 2005.
- [119] H. Pottmann, A. Asperl, M. Hofer, and A. Kilian. *Architectural Geometry*. Bentley Institute Press, 2007.
- [120] H. Pottmann, S. Brell-Cokcan, and J. Wallner. Discrete surfaces for architectural design. In P. Chenin, T. Lyche, and L. L. Schumaker, editors, *Curves and Surface Design: Avignon 2006*, pages 213–234, 2007.
- [121] H. Pottmann, Y. Liu, J. Wallner, A. Bobenko, and W. Wang. Geometry of multi-layer freeform structures for architecture. *ACM Transactions on Graphics (Proc. SIGGRAPH 2007)*, 26(3):#65,1–11, 2007.
- [122] V. Pratt. Techniques for conic splines. In *Proceedings of SIGGRAPH’85*, pages 151–160, 1985.
- [123] N. Ray, W.-C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Trans. Graphics*, 25(4):1460–1485, 2006.

- [124] A. Ruhe and P. Wedin. Algorithms for separable nonlinear least squares problems. *SIAM Review*, 22(3):318–337, 1980.
- [125] B. Sarkar and C-H Menq. Parameter optimization in approximating curves and surfaces to measurement data. *Computer Aided Geometric Design*, 8(4):267–290, 1991.
- [126] R. Sauer. *Differenzengeometrie*. Springer, 1970.
- [127] E. Saux and M. Daniel. An improved hoschek intrinsic parametrization. *Computer Aided Geometric Design*, 20(8-9):513–521, 2003.
- [128] W. K. Schief. On a maximum principle for minimal surfaces and their integrable discrete counterparts. *J. Geom. Physics*, 56:1484–1495, 2006.
- [129] N. Schmitt. Noid. Java Applet, <http://www-sfb288.math.tu-berlin.de/~nick/Noid/NoidApplet.html>, 2003.
- [130] N. Schmitt. Constant mean curvature trinoids. preprint, <http://www.arxiv.org/math.DG/0403036>, 2004.
- [131] R. B. Schnabel and E. Eskow. A revised modified cholesky factorization algorithm. *SIAM J. on Optimization*, 9(4):1135–1148, 1999.
- [132] R. Schneider. *Convex bodies: the Brunn-Minkowski theory*. Cambridge University Press, 1993.
- [133] H. Schober. Freeform glass structures. In *Glass Processing Days 2003*, pages 46–50. Glass Processing Days, Tampere (Fin.), 2003. ISBN 952-91-5910-2.
- [134] O. Schramm. Circle patterns with the combinatorics of the square grid. *Duke Math. J.*, 86:347–389, 1997.
- [135] S. Sechelmann. Koebe polyhedron editor. Java Applet, <http://www.math.tu-berlin.de/geometrie/ps/software.shtml>, 2006.
- [136] C. Sequin. CAD tools for aesthetic engineering. *CAD & Appl.*, 1:301–309, 2004.
- [137] SG. The smart geometry group. <http://www.smartgeometry.com>.
- [138] T. Speer, M. Kuppe, and J. Hoschek. Global reparameterization for curve approximation. *Computer Aided Geometric Design*, 15(9):869–877, 1998.
- [139] J. Sullivan. The aesthetic value of optimal geometry. In M. Emmer, editor, *In The Visual Mind II*, pages 547–563. MIT Press, 2005.
- [140] G. Taubin. Dual mesh resampling. *Graphical Models*, 64(2):94–113, 2002.

- [141] D. Terzopoulos and K. Fleisher. Deformable models. *Visual Computer*, 4:306–331, 1988.
- [142] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Designing quadrangulations with discrete harmonic forms. In *Symp. Geometry Processing*, pages 201–210. Eurographics, 2006.
- [143] D. Tubic, P. Hebert, and D. Laurendeau. A volumetric approach for interactive 3D modeling. *Computer Vision and Image Understanding*, 92:56–77, 2003.
- [144] S. Valette and J-M Chassery. Approximated centroidal voronoi diagrams for uniform polygonal mesh coarsening. *Computer Graphics Forum (Eurographics 2004 proceedings)*, 23(3):381–389, 2004.
- [145] S. Valette, J-M Chassery, and R. Prost. Generic remeshing of 3d triangular meshes with metric-dependent discrete voronoi diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 2007.
- [146] T. Várady and R. Martin. *Handbook of Computer Aided Geometric Design*, chapter Reverse Engineering, pages 651–681. Elsevier, 2002.
- [147] B. Vemuri and Y. Chen. Joint image registration and segmentation. In *Geometric Level Set Methods in Imaging, Vision and Graphics*, pages 251–269. Springer, 2003.
- [148] J. Wallner. Gliding spline motions and applications. *Computer Aided Geometric Design*, 21(1):3–21, 2004.
- [149] D. J. Walton and R. Xu. Turning point preserving planar interpolation. *ACM Transactions on Graphics*, 10(3):297 – 311, 1991.
- [150] C. Wang and K. Tang. Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *Vis. Computer*, 20:521–539, 2004.
- [151] W. Wang, H. Pottmann, and Y. Liu. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics*, 25:214–238, 2006.
- [152] W. Wang, J. Wallner, and Y. Liu. An angle criterion for conical mesh vertices. *J. Geometry Graphics*, 11:199–208, 2007.
- [153] X. Corina Wang and C. Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 129 – 138, 2002.

-
- [154] V. Weiss, L. Andor, G. Renner, and T. Várady. Advanced surface fitting techniques. *Computer Aided Geometric Design*, 19(1):19–42, 2002.
- [155] A. Willis, X. Orriols, and D. Cooper. Accurately estimating sherd 3d surface geometry with application to pot reconstruction. In *CVPR Workshop*, 2003.
- [156] W. Wunderlich. Zur Differenzengeometrie der Flächen konstanter negativer Krümmung. *Sitz. Öst. Ak. Wiss.*, 160:41–77, 1951.
- [157] H. Yamauchi, S. Gumhold, R. Zayer, and H-PSeidel. Mesh segmentation driven by Gaussian curvature. *Visual Computer*, 21:659–668, 2005.
- [158] H. P. Yang, W. Wang, and J. G. Sun. Control point adjustment for B-spline curve approximation. *Computer-Aided Design*, 36(7):639–652, 2004.
- [159] M. Zacharias. *Encykl. d. math. Wiss*, volume volume III AB 9, chapter Elementargeometrie und elementare nicht-euklidische Geometrie in synthetischer Behandlung. 1914.
- [160] G. Ziegler. *Lectures on Polytopes*. Springer, 1995.