GEOMETRIC PROCESSING OF CAD DATA AND MESHES AS INPUT OF INTEGRAL EQUATION SOLVERS

by

Maharavo Randrianarivony

June 25, 2006

Supervisor: Prof. Dr. Guido Brunnett

COMPUTER SCIENCE FACULTY
TECHNISCHE UNIVERSITÄT CHEMNITZ

Acknowledgment

I would like to express my sincere appreciation to my advisor Prof. Guido Brunnett for his guidance during the preparation of this thesis. The frequent discussions with him have tangibly improved my knowledge in CAGD. He has helped me a lot during the conception of most geometric approaches especially about decomposition of CAD surfaces into four-sided subsurfaces and treatment of diffeomorphisms by means of transfinite interpolations using Coons and Gordon patches. Special acknowledgment is granted to him for taking the time to correct my ideas as well as my misspellings in several former drafts of some chapters. The financial assistance that he offered me is also very much appreciated.

I am also grateful to Prof. Reinhold Schneider who has developped his explanations of the required surface representations for his mesh-free numerical methods of integral equations. Thanks are equally due to Dr. Helmut Harbrecht with whom I have had several discussions which greatly helped me to have a deeper insight about the wavelet Galerkin scheme.

Special thanks go to the Computer Graphic Group which has provided me with computing equipments that helped me implement the proposed approaches. Additional thanks are due to the Computer Science Faculty of the Technische Universität Chemnitz for giving me an environment where I could pursue my research.

Last but not the least, I thank my sister Lova who has given me moral supports during the frustrating period of working on this thesis.

Abstract

Among the presently known numerical solvers of integral equations, two main categories of approaches can be traced:

- Mesh-free approaches
- Mesh-based approaches.

We will propose some techniques to process geometric data so that they can be efficiently used in subsequent numerical treatments of integral equations. In order to prepare geometric information so that the above two approaches can be automatically applied, we need the following items.

- Splitting a given surface Γ into several four-sided patches Γ_i.
- Generating a diffeomorphism γ_i from the unit square to Γ_i.
- Generating a mesh $\mathcal{M}$ on a given surface.
- Patching of a given triangulation.

In order to have a splitting $\Gamma = \cup_{i=1}^{N} \Gamma_i$, we need to approximate the surfaces first by polygonal regions. We use afterwards quadrangulation techniques by removing quadrilaterals repeatedly. We will generate the diffeomorphisms by means of transfinite interpolations of Coons and Gordon types.

The generation of a mesh $\mathcal{M}$ from a piecewise Riemannian surface will use some generalized Delaunay techniques in which the mesh size will be determined with the help of the Laplace-Beltrami operator.

We will describe our experiences with the IGES format because of two reasons. First, most of our implementations have been done with it. Next, some of the proposed methodologies assume that the curve and surface representations are similar to those of IGES.

Patching a mesh consists in approximating or interpolating it by a set of practical surfaces such as B-spline patches. That approach proves useful when we want to utilize a mesh-free integral equation solver but the input geometry is represented as a mesh.

Contents

Acknowledgment iii
Abstract v
List of notations xi
List of figures xii

1 INTRODUCTION 1
1.1 Brief reminder about integral equations 1
1.1.1 Wavelet Galerkin method 3
1.1.2 Mesh-based method 4
1.2 Challenges of geometric processing 5
1.3 Main results 6
1.3.1 Chapter 2: Building an adjacency model from an IGES format 7
1.3.2 Chapter 3: Decomposing a surface into four-sided patches . 8
1.3.3 Chapter 4: Constructing diffeomorphic reparametrization . 9
1.3.4 Chapter 5: Mesh generation 10
1.3.5 Chapter 6: Paving of meshes by four-sided patches 12

2 IGES FORMAT AND SURFACE STRUCTURES 13
2.1 Geometric storage and transmission 13
2.1.1 CAD interface usage 13
2.1.2 IGES as a CAD neutral format 14
2.2 IGES file structure 15
2.2.1 Entity classification and data types 15
2.2.2 IGES file organization 16
2.2.3 Shortcomings and proposed remedies 18
2.3 Segment separators 21
2.4 Features of the implementation 21
2.5 Practical experiences and CAD flaws 22
2.6 Surface structures 24
2.6.1 Topological structure 24
2.6.2 Metric structure 26
2.7 Other CAD interfaces 27

3 FOURSIDED SPLITTING **29**
3.1 Introduction 29
3.2 Problem setting and general approach 31
3.3 Polygonal regions 32
3.4 Removing a quadrilateral from a simple polygon 35
3.5 Toward multiply connected polygons 38
3.6 Quadrilating multiply connected polygons 46
3.7 Cut search 49
3.7.1 Cuts having extreme vertices 50
3.7.2 Cuts having nonextreme vertices 50
3.7.3 Summary 53
3.8 Conversion into convex quadrangulation 54
3.9 Cleanup 55
3.9.1 Quality control 56
3.9.2 Cleanup operations 57
3.10 Converting a triangulation into a quadrangulation 58
3.11 Regions having curved boundaries 61
3.12 Structure of the polygonal model 61
3.12.1 Discretization of curved boundaries 62
3.12.2 Edge splitting 62
3.12.3 Even polygonal approximation 63
3.13 Metric aspect of the polygonal model 65
3.13.1 Discretization artifact 65
3.13.2 Edge quality 66
3.13.3 Polygonal approximation 67
3.14 Postprocessing 68
3.14.1 Boundary replacement 68
3.14.2 Boundary interference 69
3.14.3 Treating G^1 vertices 72
3.14.4 Subdivision for diffeomorphisms 74
3.15 Summarizing algorithm 75
3.16 Special splittings 75
3.16.1 All G^1 vertices 76
3.16.2 Simple polygons 76
3.17 Numerical results 81

4 MAPPING REGULARITY **91**
4.1 Introduction 91
4.2 Transfinite interpolation and problem setting 92
4.3 First sufficient condition 95
4.4 Second sufficient condition 99
4.5 Sufficient and necessary condition 100
4.5.1 Subdivision 101
4.5.2 Adaptivity 103

4.6 Numerical results 105
4.7 Mapping search 108
  4.7.1 Overspill phenomenon 109
  4.7.2 Gordon patch 109
4.8 Generation of internal curves 110
  4.8.1 Finding the internal points $\mathbf{x}_{ij}$ 111
  4.8.2 Generating the interpolating curves 113
4.9 Diffeomorphic Gordon and termination guarantee 114
4.10 Practical results 119

5 MESH GENERATION 121
5.1 Introduction 121
5.2 Definitions and problem setting 122
5.3 Motivation for the planar case 124
5.4 Meshing using the first fundamental form 127
  5.4.1 Splitting according to first fundamental form 127
  5.4.2 Flipping according to first fundamental form 128
5.5 Initial coarse mesh 128
5.6 Nonsmooth boundary approximation 129
5.7 Determination of the edge size function 131
5.8 Meshing of a surface with multiple patches 133
5.9 Discretization of the curved boundaries 133
5.10 Theoretical discussion 135
5.11 Numerical results 139

6 $\mathcal{C}^0$-PAVING OF MESHES BY QUADRILATERAL PATCHES 147
6.1 Introduction 148
6.2 Problem formulation 148
6.3 Handle decomposition 149
6.4 Preliminary search for canonical curves 151
  6.4.1 Normalized canonical form 151
  6.4.2 Numerical realization 154
  6.4.3 Bases of the fundamental group 155
  6.4.4 Homotopic transformation 156
6.5 Improvements of the canonical curves 158
  6.5.1 Weighted split graph 159
  6.5.2 Improvement algorithm 159
6.6 Paving into quadrilateral patches 160
  6.6.1 Mesh parametrization 161
  6.6.2 Splitting into four-sided submeshes 163
6.7 Surface fitting with $\mathcal{C}^0$-joint 164
6.8 Numerical results 167

A NUMERICAL TOPOLOGY **173**
A.1 Simplicial complexes 173
A.2 Boundary operator and incidence matrix 174
A.3 Homology group 175

B CURVES ON SURFACES **177**
B.1 Topological surfaces 177
B.2 Classification of surfaces 179
B.3 Fundamental group 180
B.4 Reidemeister moves 180

BIBLIOGRAPHY **i**

List of Notations

$\mathbf{Z}, \mathbf{R}$: the integers, the reals.
$(\mathbf{ab})^+$: half plane on the right of $\overrightarrow{\mathbf{ab}}$.
$(\mathbf{ab})^-$: half plane on the left of $\overrightarrow{\mathbf{ab}}$.
$\mathcal{W}(a)$: wedge of a vertex a inside a polygon.
$\ker(P)$: kernel of the polygon P.
δ_{ij} : Kroenecker symbol.
$\mathrm{Ker}(A)$: kernel of a linear operator A.
$\mathrm{Im}(A)$: image of a linear operator A.
$[\mathbf{u}, \mathbf{v}]$: determinant of the 2D vectors $\mathbf{u}$ and $\mathbf{v}$.
$\mathcal{D}(\mathbf{x})$: Dirichlet energy of the function $\mathbf{x}$.
σ^k : k-dimensional simplex.
∂_k : k-th boundary operator.
$H_k(\mathcal{C})$: k-th homology group of a chain complex $\mathcal{C}$.
$\pi_1(X, x_0)$: (first) fundamental group.
$[\sigma^{n+1} : \sigma^n]$: incidence number of σ^{n+1} and σ^n.
E^k : k-th incidence matrix.
$\beta_k(\mathcal{C})$: k-th Betti number.
B_i^n : Bernstein polynomial.
N_i^n : B-spline basis function.
Δ : Laplace operator.
$\Delta_{\mathbf{S}}$: Laplace-Beltrami operator w.r.t. the surface $\mathbf{S}$.
$[a, b]$: a commutator $aba^{-1}b^{-1}$ if a and b are members of an alphabet.
$L^2(X)$: set of all square integrable functions in X.
$H^1(\Omega)$: Sobolev space of order one in Ω.
$H^{1/2}(\Gamma)$: set of all functions in $L^2(\Gamma)$ which are boundary values of functions in $H^1(\Omega)$ where $\Gamma = \partial\Omega$.
$< \cdot, \cdot >$: scalar product in $\mathbf{R}^n$.
$(\cdot, \cdot)_X$: scalar product in $L^2(X)$.
$[a, b, c]$: triangle with apices a, b, c.
$\square$: end of a proof.

List of Figures

Fig. 1.1: (a)Inadmissible mesh (b) Edge lengths 4
Fig. 1.2: Flow diagram about geometric preparation 6

Fig. 2.1: Sample of an IGES file 20
Fig. 2.2: (a)Segment separators (b)Nonperiodic cylinder 23
Fig. 2.3: CAD imperfection 24
Fig. 2.4: Three types of adjacent patches 25
Fig. 2.5: (a)Nonadjacent surfaces (b)Adaptive search 26
Fig. 2.6: (a)Metric information (b)Covered edge 26

Fig. 3.1: Splitting into four-sided subregions 30
Fig. 3.2: (a) Double edge (b) Internal domain of a polygon. 33
Fig. 3.3: Chopping off a quadrilateral 36
Fig. 3.4: (a) Generate ω (b) $\mathbf{x}_{i+2}$ is reflex in $\tilde{Q}$. 36
Fig. 3.5: (a),(b)Positions of reflex vertex (c)Steiner point 37
Fig. 3.6: A ear and a quadrilateral are adjacent 37
Fig. 3.7: (a)Initial polygon (b)Make simply connected (c)Make even 39
Fig. 3.8: Original and auxiliary polygons. 41
Fig. 3.9: Inadmissible polygons 41
Fig. 3.10: Counter-example for multiply connected polygons 43
Fig. 3.11: (a) $\kappa(\mathbf{x}_{\sigma+2}) \leq \kappa(\mathbf{x}_{\sigma})$ (b) $\kappa(\mathbf{x}_{\sigma}) < \kappa(\mathbf{x}_{\sigma+2}) < \kappa(\mathbf{x}_{\sigma+1})$. 44
Fig. 3.12: Chopping an ear off next to a shifted node. 45
Fig. 3.13: Double nodes: (a) $\mathbf{x}_i$ and $\mathbf{x}_j$ are convex (b) $\mathbf{x}_i$ is reflex. 46
Fig. 3.14: Cuts joining the interior and the exterior boundaries 47
Fig. 3.15: Quadrilating a multiply connected polygon 49
Fig. 3.16: Finding a cut and the possible directions 51
Fig. 3.17: (a),(b)Segment rectangle intersection (c)Domain of interest 51
Fig. 3.18: Forming a hexagon from two quadrilaterals 55
Fig. 3.19: Cleanup: (a) Shifting a node ω(b) Flipping an edge 57
Fig. 3.20: Effect of cleanup 58

Fig. 3.21: Edge-graph of a mesh and sibling in a tree 59
Fig. 3.22: Convert a triangulation into a quadrangulation.................. 60
Fig. 3.23: (a)Triangulation (b),(c)Two possible quadrangulations 60
Fig. 3.24: Initial polygonal approximation 61
Fig. 3.25: New node insertion 63
Fig. 3.26: Converting odd faces into even ones............................. 64
Fig. 3.27: Discretization artifacts.. 65
Fig. 3.28: Edges of bad quality 67
Fig. 3.29: Subregions with curved sides from quadrangulation 69
Fig. 3.30: First boundary interference and requadrangulation 70
Fig. 3.31: Second boundary interference and requadrangulation 70
Fig. 3.32: Illustration of G^1 vertices 73
Fig. 3.33: An edge should emanate from a G^1 vertex 73
Fig. 3.34: Refinement for diffeomorphism 74
Fig. 3.35: Special splittings 76
Fig. 3.36: Splitting: nonconvex quadrilaterals and pentagons 77
Fig. 3.37: Hexagon: one reflex vertex...................................... 78
Fig. 3.38: Hexagon: two reflex vertices 79
Fig. 3.39: Hexagon: three reflex vertices.................................. 80
Fig. 3.40: Intersection of two convex polygons 80
Fig. 3.41: First set of results. 82
Fig. 3.42: Second set of results. 83
Fig. 3.43: Third set of results. 84
Fig. 3.44: Fourth set of results. 85
Fig. 3.45: Fifth set of results. 86
Fig. 3.46: First set of four-sided splittings 88
Fig. 3.47: Second set of four-sided splittings 89

Fig. 4.1: A four sided domain for Coons patch 93
Fig. 4.2: Diffeomorphic Coons patches 94
Fig. 4.3: Undesired overspill phenomena 95
Fig. 4.4: Tagent vectors for Coons patch 97
Fig. 4.5: Multiple subdivisions: (a)uniform (b)adaptive................... 103
Fig. 4.6: Control polygons in Coons patch for $\sigma = 0.216$. 105
Fig. 4.7: Examples of diffeomorphisms 107
Fig. 4.8: A network of curves for Gordon patch 108
Fig. 4.9: Meshes $\mathcal{M}$ and $\mathcal{N}$ for point search 110

Fig. 4.10: Distortion quantification 111
Fig. 4.11: Image of R_{ij} by a Gordon patch and an example 115
Fig. 4.12: Subdivision and termination guarantee 118
Fig. 4.13: Examples of a diffeomorphic Gordon patch 120

Fig. 5.1: (a)Inadmissible mesh (b) Edge lengths 124
Fig. 5.2: Planar case: (a)boundary nodes (b)internal nodes 125
Fig. 5.3: Mesh on a trimmed surface and its 2D-correspondant 126
Fig. 5.4: (a)Splitting (b)Flipping 127
Fig. 5.5: Selected steps in mesh recursive refinement 128
Fig. 5.6: Nonsmoothness of first kind 130
Fig. 5.7: Nonsmoothness of second kind 130
Fig. 5.8: Nonsmoothness of third kind 131
Fig. 5.9: (a)Boundary nodes (b)Surface mesh 134
Fig. 5.10: Two-ear theorem 136
Fig. 5.11: Computing the cotan formula 139
Fig. 5.12: Mesh with 5424 elements and 2710 vertices 140
Fig. 5.13: Mesh with 32310 elements and 16145 vertices 141
Fig. 5.14: Edge length 141
Fig. 5.15: Mesh with 7314 elements and 3651 vertices 142
Fig. 5.16: Results from CAD data in IGES files 144
Fig. 5.17: Results from CAD data in IGES files 145

Fig. 6.1: Triangulation paving 147
Fig. 6.2: A surface with genus 2 and its four canonical curves 150
Fig. 6.3: A genus zero surface and its flattened polygon 150
Fig. 6.4: (a)Homology (b)Node coincidence 152
Fig. 6.5: Approach paths 153
Fig. 6.6: Double nodes and homotopic shift 156
Fig. 6.7: Loop shiftings 156
Fig. 6.8: Common edges in loops 157
Fig. 6.9: Loop intersections 158
Fig. 6.10: Dual graph and undesirable canonical curves 160
Fig. 6.11: parametrization of a pretzel 160
Fig. 6.12: Parametrization of a torus 161
Fig. 6.13: Triangular and quadrilateral decompositions 163
Fig. 6.14: Edge cutting and triangle subdivision 164

Fig. 6.15: The B-spline curves $\mathcal{C}_I$, $\mathcal{C}_J$ $\mathcal{C}_K$, $\mathcal{C}_L$ are interpolated by $\mathcal{S}_r$ 166
Fig. 6.16: Result 1: before Reidemeister moves 167
Fig. 6.17: Result 1: after Reidemeister moves 168
Fig. 6.18: Result 1: after combinatorial optimization 169
Fig. 6.19: Result 2: before Reidemeister moves 169
Fig. 6.20: Result 2: after Reidemeister moves 169
Fig. 6.21: Result 2: magnification 170
Fig. 6.22: Result 2: after combinatorial optimization 170
Fig. 6.23: Example 1 of surface approximation 170
Fig. 6.24: Example 2 of surface approximation 171
Fig. 6.25: Example 3 of surface approximation 171

Fig. B.1: A polygonal disk with code $aba^{-1}b^{-1}cdc^{-1}d^{-1}$ 178
Fig. B.2: Two surfaces of genus unity..................................... 178
Fig. B.3: A topological disk and a topological surface 179
Fig. B.4: A surface of genus zero with its code $\omega = abb^{-1}a^{-1}$.............. 180
Fig. B.5: Equivalent and non-equivalent loops 181
Fig. B.6: Unnecessary crossings and first Reidemeister move 181
Fig. B.7: Second and third Reidemeister moves 182

Chapter 1

INTRODUCTION

In this introductory chapter, we will try to explain the importance of geometric preprocessing in the context of integral equations. Thereto, we will describe the standard forms in which surfaces have to be given to serve as input of integral equation solvers. If the geometry of a particular application is given in a different way, a geometric conversion process has to be performed. We will formulate the challenges in the design of a semi-automatic conversion process and summarize the main results of the thesis.

1.1 Brief reminder about integral equations

In order to fix the idea, we are going to solve the following Laplace problem with Dirichlet boundary condition. For a given function $g \in H^{1/2}(\Gamma)$, search for some $\mathcal{U} \in H^1(\Omega)$ such that

$$\left\{ \begin{array}{rlcl} \Delta\mathcal{U} & = 0 & \text{in} & \Omega \\ \mathcal{U} & = g & \text{on} & \Gamma, \end{array} \right. \tag{1.1}$$

where $\Omega \subset \mathbf{R}^3$ with boundary $\Gamma := \partial\Omega$.

Since we deal with a 3D problem, this can be reduced [61] to the solving of the following integral equation of the second kind on the 2D manifold Γ

$$u(\mathbf{x}) + \int_\Gamma \mathbf{k}(\mathbf{x},\mathbf{t})u(\mathbf{t})d\mathbf{t} = -2g(\mathbf{x}) \qquad \forall\, \mathbf{x} \in \Gamma, \tag{1.2}$$

in which the unknown is $u \in H^{1/2}(\Gamma)$ and the kernel function $\mathbf{k}$ is given by

$$\mathbf{k}(\mathbf{x},\mathbf{t}) := \frac{1}{2\pi}\frac{\nu(\mathbf{t})\cdot(\mathbf{t}-\mathbf{x})}{\|\mathbf{t}-\mathbf{x}\|^3} \qquad \text{for } \mathbf{x},\mathbf{t} \in \Gamma, \tag{1.3}$$

where $\nu(\mathbf{t})$ denotes the outward normal vector at a point $\mathbf{t}$. Let us denote by $\mathcal{K}$ the double layer operator which assigns to a function f the function $\mathcal{K}f$ that is given by

$$(\mathcal{K}f)(\mathbf{x}) := \int_\Gamma \mathbf{k}(\mathbf{x},\mathbf{t})f(\mathbf{t})d\mathbf{t} \qquad \text{for } \mathbf{x} \in \Gamma. \tag{1.4}$$

The original unknown $\mathcal{U}$ of the Laplace problem (1.1) can therefore be deduced from u by applying the double layer operator:

$$\mathcal{U} = \mathcal{K}u. \tag{1.5}$$

By multiplying the equation (1.2) with a function $v \in H^{1/2}(\Gamma)$ and by taking the integral over Γ afterwards, we obtain

$$\int_\Gamma \left\{ u(\mathbf{x}) + \int_\Gamma \mathbf{k}(\mathbf{x},\mathbf{t})u(\mathbf{t})d\mathbf{t} \right\} v(\mathbf{x})d\mathbf{x} = \int_\Gamma -2g(\mathbf{x})v(\mathbf{x})d\mathbf{x}\,. \tag{1.6}$$

With the double layer operator $\mathcal{K}$ in place, we can introduce a second operator $\mathcal{A}$ defined by

$$(\mathcal{A}f)(\mathbf{x}) := f(\mathbf{x}) + (\mathcal{K}f)(\mathbf{x}) \qquad \forall\,\mathbf{x} \in \Gamma. \tag{1.7}$$

We obtain therefore the following variational formulation: search for $u \in H^{1/2}(\Gamma)$ such that

$$(\mathcal{A}u, v)_\Gamma = (f, v)_\Gamma \qquad \forall v \in H^{1/2}(\Gamma)\,. \tag{1.8}$$

In general, the approximation scheme consists in considering a finite dimensional space R_h where we approximate the function u by $u_h \in R_h$ in which we solve the following problem

$$(\mathcal{A}u_h, v_h)_\Gamma = (f, v_h)_\Gamma \qquad \forall v_h \in R_h. \tag{1.9}$$

There are many numerical methods to solve integral equations. They range from Nyström [102, 103], collocation [25], to Galerkin [3] methods. That categorization can be further sorted in subcategories and there are also mixed methods that intend to combine the advantages of different approaches. For a comprehensive in-depth study about convergence results and numerical treatments of integral equations, see [61, 3]. The main problem in the treatment of integral equations is that the resulting linear system is not sparse. Many methods have been proposed to overcome this difficulty. These methods include the panel clustering [62], wavelet compression [109, 63, 64], and fast multipole [59]. Some approach about preconditioning the linear system from the wavelet Galerkin scheme is also proposed in [26].

Regarding the assumptions made about the geometry, we can distinguish between methods that assume that Γ is represented as a polygonal mesh or as a parametric surface composed of smooth polynomial patches. Examples of mesh-free approaches include the works of R. Schneider, A. Kunoth, W. Dahmen, H. Harbrecht [109, 63, 64, 28]. Examples of mesh-based methods include the works of Hackbush [61] and Atkinson [3]. Due to our cooperation with Prof. Schneider in the SFB 393, we were mainly interested in the wavelet Galerkin method.

1.1.1 Wavelet Galerkin method

The wavelet Galerkin method [109, 63, 26] is a special case of (1.9) in which the construction of the finite dimensional space R_h is done by means of multiresolution techniques. That method requires that one generates wavelets on a manifold Γ. On that account, the manifold is given as

$$\Gamma = \bigcup_i \Gamma_i, \tag{1.10}$$

where Γ_i is the image of a mapping

$$\gamma_i : [0,1]^2 \longrightarrow \Gamma_i. \tag{1.11}$$

The splitting is also supposed to be conforming, i.e. the intersection of two different patches Γ_i and Γ_j is either empty or it is the image of a whole edge of the unit square or it is a vertex. Basically, the construction of wavelets on the manifold Γ is performed in three steps:

1. Define wavelets on the unit interval $[0,1]$.
2. Use tensor products method to obtain wavelets on the unit square $[0,1]^2$ from the results on the unit interval.
3. Use parametric lifting with the help of the parametric function γ_i to carry the results from the unit square over to the manifold Γ_i.

After that, one can deduce two families of nested linear spaces

$$\left\{ \begin{array}{l} S_{j_0} \subset S_{j_0+1} \subset \cdots \subset S_j \subset S_{j+1} \subset \cdots \subset L^2(\Gamma) \\ \tilde{S}_{j_0} \subset \tilde{S}_{j_0+1} \subset \cdots \subset \tilde{S}_j \subset \tilde{S}_{j+1} \subset \cdots \subset L^2(\Gamma), \end{array} \right. \tag{1.12}$$

known as primal and dual multiresolutions[63, 28]. Now the wavelet spaces W_j (resp. $\tilde{W}_j$) which are complements of S_j (resp. $\tilde{S}_j$) in S_{j+1} (resp. $\tilde{S}_{j+1}$) [63, 28] are introduced:

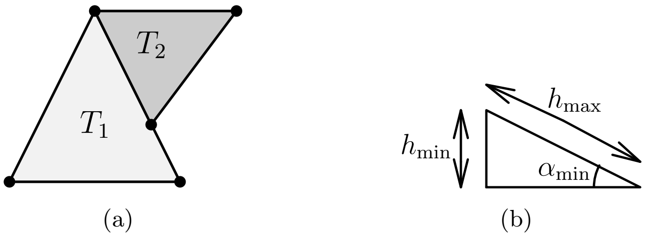
$$S_{j+1} = S_j \oplus W_j \qquad \text{and} \qquad \tilde{S}_{j+1} = \tilde{S}_j \oplus \tilde{W}_j. \tag{1.13}$$

After a multiple applications of relation (1.13), one has the following direct sums for any $J > j_0$

$$S_J = S_{j_0} \oplus W_{j_0} \oplus W_{j_0+1} \oplus \cdots \oplus W_{J-1}. \tag{1.14}$$

The discretization method for solving the integral equation (1.2) consists in using S_J as finite dimensional space in (1.9).

On the other hand, one needs to use numerical quadratures to evaluate the integrals involved in the entries of the stiffness matrix which results from the equation



(a) (b)

Figure 1.1: (a)Inadmissible mesh (b) Edge lengths

(1.9) when applied to the Wavelet Galerkin scheme. If the parametric function γ_i is a diffeomorphism, then the following first fundamental tensor is symmetric definite positive:

$$K_i(\mathbf{s}) = \left[\left(\frac{\partial \gamma_i(\mathbf{s})}{\partial s_j}, \frac{\partial \gamma_i(\mathbf{s})}{\partial s_k} \right) \right]_{j,k=1,2}. \tag{1.15}$$

As a consequence, integrations over the manifold Γ can be transformed on the unit square:

$$\int_\Gamma u(\mathbf{x}) v(\mathbf{x})\, d\sigma(\mathbf{x}) = \sum_{i=1}^{N} \int_{[0,1]^2} u(\gamma_i(\mathbf{s})) v(\gamma_i(\mathbf{s})) \sqrt{\det\left(K_i(\mathbf{s})\right)}\, d\mathbf{s}. \tag{1.16}$$

1.1.2 Mesh-based method

There are some numerical solvers [3, 61] of integral equations which require the input geometric information to be represented as a mesh $\mathcal{M}_h$. In that case, the finite dimensional space R_h is usually chosen to be piecewise polynomials. That is, if we denote by $\mathcal{P}^k$ the set of polynomials of degree at most k, then

$$R_h \subset \{p : p|_T \in \mathcal{P}^k \quad \forall T \in \mathcal{M}_h\}. \tag{1.17}$$

The two standard assumptions that one expects from the triangular elements of the mesh $\mathcal{M}_h$ are the following:

(A1) The intersection of two triangles having nonempty intersection is either a node or a complete edge.

(A2) The smallest angle $\alpha_{\min}(T)$ inside each triangle T is larger than some prescribed threshold $\alpha_0 > 0$.

As an illustration, the triangles in Fig. 1.1(a) does not fulfill condition (A1). Note that if the lengths of the three edges in any triangle $T \in \mathcal{M}_h$ are proportional, then condition (A2) follows. That is, if we have

$$h_{\max}(T) \approx h_{\min}(T), \tag{1.18}$$

where $h_{\max}(T)$ and $h_{\min}(T)$ are the lengths of the longest and shortest edges of T respectively (Fig. 1.1(b)) then (A2) holds. A mesh having triangles satisfying (1.18) is usually nicely shaped.

There are basically two types of approximation methods which use a mesh: h-version and p-version. The degrees of the polynomials inside the triangles are kept constant if one uses the h-version approaches. That is, if we want to have more accurate approximation u_h of the solution u to the integral equation, then we have to refine the mesh $\mathcal{M}_h$. For that case, we need an a-posteriori error estimator $\varepsilon_T(u_h)$ which has the following property:

$$\lambda_1 \sum_{T\in\mathcal{M}_h} \varepsilon_T(u_h) \leq \|u - u_h\| \leq \lambda_2 \sum_{T\in\mathcal{M}_h} \varepsilon_T(u_h), \tag{1.19}$$

where λ_1 and λ_2 are positive constants independent of u and u_h. There are different methods of obtaining a-posteriori error estimators which can be used to identify the parts of the mesh that need to be refined. The special property of an a-posteriori error estimator $\varepsilon_T(u_h)$ is that it can be computed without knowing the function u. If the value of the a-posteriori error estimator $\varepsilon_T(u_h)$ with respect to the triangle T exceeds some prescribed value ε_0, then we subdivide the triangle T into several subtriangles. In the case of p-versions, the polynomial degrees are allowed to be variable. In order to obtain a better accuracy, the mesh $\mathcal{M}_h$ does not need to be refined. Instead, we increase polynomial degrees k of the piecewise polynomials in relation (1.17). A combination hp-version also exists in order to improve numerical performance.

1.2 Challenges of geometric processing

With the development of fast methods to solve integral equations, one became interested to apply these methods to real world problems that usually come with a non-trivial geometry. In this thesis, we consider 3D objects that have been created with a CAD system. We assume that the boundary of such an object is a closed 2D manifold of arbitrary genus. Since real world objects will contain curved regions, we will focus on surfaces that are represented as collections of parametric surface patches. However, we will also consider some aspects of the processing of triangular meshes.

In order to be able to treat objects that have been created by different CAD systems [67], we use the standard exchange format IGES as input description to our geometry processing. All examples of 3D objects in this thesis have been created with Pro-Engineer. We will consider three tasks to preprocess geometric data for subsequent use in integral equations. The main task of this thesis can be formulated as follows. Given an IGES file that describes a 2D manifold Γ of arbitrary genus as a collection of possibly trimmed surface patches. Then, the problem is to design and realize a program that outputs an exact representation

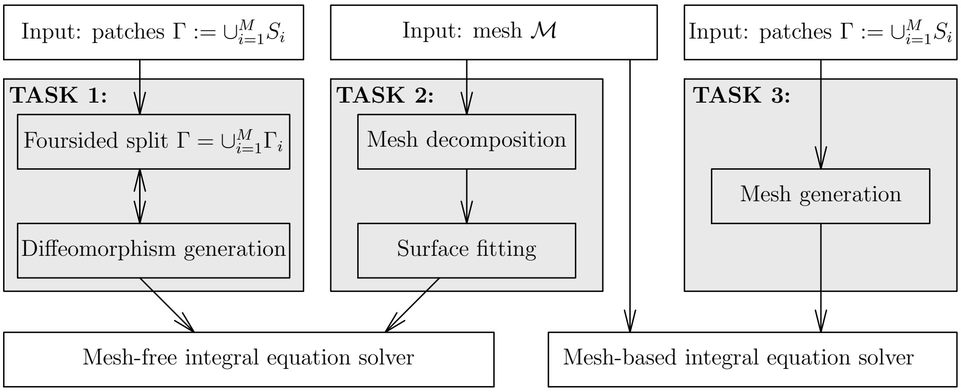

Input: patches $\Gamma := \cup_{i=1}^{M} S_i$
Input: mesh $\mathcal{M}$
Input: patches $\Gamma := \cup_{i=1}^{M} S_i$
TASK 1:
Foursided split $\Gamma = \cup_{i=1}^{M} \Gamma_i$
Diffeomorphism generation
TASK 2:
Mesh decomposition
Surface fitting
TASK 3:
Mesh generation
Mesh-free integral equation solver
Mesh-based integral equation solver

Figure 1.2: Flow diagram about geometric preparation

R of Γ as a collection of four-sided patches $R = \{\Gamma_i\}_{i=1}^{n}$. Note that the decomposition R has to be conforming. That is to say, the intersection of two different four-sided patches Γ_i and Γ_j which is not empty must either a point or a complete curved edge. Further, the parametrization $\gamma_i : [0,1]^2 \longrightarrow \Gamma_i$ of each patch has to be a C^1-diffeomorphism. The number of patches n should stay reasonably small. Angles and curve lengths below a user supplied threshold should be avoided. The user interaction required to perform the conversion should be as low as possible.

Beside the main task described above, we will also consider two other tasks related to the processing of geometry as input for integral equation solvers.

If a mesh-based solver is available [3, 61] but the surface is given in piecewise parametric form, one needs to create a triangular mesh from the parametric surface. In Chapter 5, an algorithm that creates a high quality discretization is described. It uses a generalized Delaunay triangulation technique which invokes the first fundamental form.

If a mesh free solver is available but the surface is given as triangular mesh, one may consider to approximate the mesh by a collection of parametric surfaces. In Chapter 6, we present a method for this task that consists of a mesh decomposition and a subsequent surface fitting step.

Fig. 1.2 illustrates the three different tasks of geometry processing as a flow chart where the input is either a set of trimmed parametric surfaces or a mesh.

1.3 Main results

In this section, we will summarize our main results. There are three chapters devoted to the solution of task 1 in this thesis.

1.3.1 Chapter 2: Building an adjacency model from an IGES format

Since we defined IGES to serve as our exchange format, it was necessary to realize routines that read IGES files. The implementation of the data extraction from an IGES file is a very time-consuming and tedious task. First, we need to assemble routines which can find information about the components of the stored geometry. Special functions have to be implemented in order to locate the positions of separators, IGES sections and IGES records which can be used to identify the values pertaining to IGES entities. Besides, there are geometric entities which need large memory storage. For instance, a B-spline surface having a large number of de-Boor points needs some treatment to reduce storage requirements. In order to be able to withdraw those entities from IGES files, we need to have the information about their size. This knowledge has to be there before data extraction, so that efficient memory allocation or deallocation become possible. On the other hand, we have to implement a large number of extraction routines for the different IGES entities. Since the IGES description varies from one IGES entity to another, we need a set of routines for the loading of the record related to each IGES entity. The IGES format contains geometric and nongeometric entities. Since we want to preprocess geometric data for integral equations, we are only interested in geometric entities. From all IGES entities, we have implemented the 12 most important geometric ones for our applications. For each entity, a conversion from the IGES records to the internal data structure has to be implemented. There are entities which are combinations of other smaller entities. For example, a composite curve can consist of different types of curves. As a consequence, we must have efficient data structures to organize the components of the stored geometry. Finally, we have to implement an evaluation routine for each data structure to provide access to the needed information in the geometric algorithms.

The main problem in the context of geometry processing is that the IGES file does not contain information about the topological structure of the surface components in general. Therefore, it was necessary to convert the IGES description into an internal model that contains structural and metric information. For this, we have developed a method which can be used to test coincidence between two curves. We use an adaptive algorithm to efficiently characterize if a point belongs to a given curve. The topological model is enhanced with metric information. That is, we store not only the adjacency between surfaces but also the parameter values of the intersections. That information could be useful to improve the adjacency test and to perform an edge refinement process in polygonal approximations. Furthermore, we describe CAD imperfections which often make the implementations of geometric methods difficult. The presence of CAD flaws might cause inaccurate or incomplete adjacency information.

1.3.2 Chapter 3: Decomposing a surface into four-sided patches

We will consider the decomposition of a closed surface S given as a set of surfaces $\{S_i\}_{i\in\Lambda}$ into a collection of four-sided subsurfaces according to the constraints stated in section 1.2:

$$S = \bigcup_{i=1}^{m} F_i\,. \tag{1.20}$$

We suppose that each S_i is the image by a function ψ_i of a multiply connected 2D-region $\mathcal{D}_i$ having possibly curved boundaries.

Our main approach to achieve (1.20) consists in splitting the 2D regions $\mathcal{D}_i$ into four-sided regions $Q_{k,i}$:

$$\mathcal{D}_i = \bigcup_{k} Q_{k,i}. \tag{1.21}$$

Thereto for each $\mathcal{D}_i$, we create an even polygonal approximation $P^{(i)}$ which we decompose into a set of quadrilaterals $q_{k,i}$. The four-sided domains $Q_{k,i}$ are obtained from $q_{k,i}$ by replacing the straight boundary edges of $q_{k,i}$ by the corresponding curve portion of $\mathcal{D}_i$. That process could generate boundary interferences which need to be detected and repaired.

So as to have a decomposition which is conforming everywhere, we proceed as follows. We approximate the curved boundaries of $\{S_i\}$ by straight line segments separated by nodes $\{X_k\} \subset \mathbf{R}^3$. Then, we make the local splitting (1.21) in such a way that it is conforming inside $\mathcal{D}_i$ and that it uses only the preimages $\psi_i^{-1}(X_k)$ of the nodes $\{X_k\}$ as boundary vertices. That is, we do not use any additional boundary nodes.

The method that we propose for the local split (1.21) tessellates a polygon with n boundary vertices into $\mathcal{O}(n)$ convex quadrilaterals. Therefore, if the number of its boundary vertices n_i for all polygons $P^{(i)}$ is smaller than n, then the total number of quadrilaterals is $\mathcal{O}(M \cdot n)$. For all examples that we considered, we found that the total number of quadrilaterals is quite small. However, we do not examine how close our local approach comes in average to the globally optimal solution.

In order to realize the above approach, we should solve the problem about quadrilating multiply connected polygons. We show that for a simply connected polygon, either one can chop off a quadrilateral which is not necessarily convex by inserting a cut or one can introduce an internal Steiner point to remove a convex quadrilateral. In order to generalize this result about simply connected polygons to multiply connected ones, the notion of double-edged polygons is introduced. Such a polygon may contain different nodes having the same coordinates but its interior is connected. We prove that the above result about the decomposition of a simply connected polygon holds true for double-edged ones. In its proof, the generalization of the two-ear theorem for double-edged polygons is used. In order to generate a double-edged polygon from a multiply connected one, a cut per internal boundary is inserted. Each cut is afterwards replaced by double edges which are traversed in opposite direction.

By applying the above theoretical results, a quadrangulation technique is obtained that repeatedly removes quadrilaterals. Since the process of removing quadrilaterals by inserting cuts might generate a quadrangulation $\mathcal{Q}$ having nonconvex quadrilaterals, the resulting quadrangulation is converted into another one which contains only convex quadrilaterals as follows. First, we merge every pair of nonconvex quadrilaterals whose union is a quadrilateral. The second step consists in forming the hexagon which is formed by the union of any nonconvex quadrilateral Q and a neighboring quadrilateral P. After generating a convex quadrangulation $\mathcal{Q}_{\mathrm{loc}}$ of the hexagon, the union $Q \cup P$ of $\mathcal{Q}$ is replaced by $\mathcal{Q}_{\mathrm{loc}}$.

On the other hand, we do not want any four-sided domain which has G^1-vertices (i.e. smooth corners). Therefore, a repairing process is used in order to ensure that some internal edges emanate from such vertices.

1.3.3 Chapter 4: Constructing diffeomorphic reparametrization

Here we address the following subproblem that arises after a closed surface has been split into four-sided patches. We consider the generation and characterization of diffeomorphisms originating from transfinite interpolations. Consider four parametric curves α, β, γ, δ which are defined in $[0,1]$. We would like to use the corresponding Coons map with respect to the blending functions F_0 and F_1

$$\mathbf{x}(u,v) = -\begin{bmatrix} -1 \\ F_0(u) \\ F_1(u) \end{bmatrix}^T \begin{bmatrix} \mathbf{0} & \alpha(u) & \gamma(u) \\ \delta(v) & \alpha(0) & \delta(1) \\ \beta(v) & \beta(0) & \beta(1) \end{bmatrix} \begin{bmatrix} -1 \\ F_0(v) \\ F_1(v) \end{bmatrix}. \tag{1.22}$$

We want to find some conditions to identify if $\mathbf{x}$ is a diffeomorphism. Apart from theoretical exactness, we are mainly interested in finding criteria which are easy to check in practice. We will suppose that the curves are given in Bézier form:

$$\alpha(t) = \sum_{i=0}^{n} \alpha_i B_i^n(t), \qquad \beta(t) = \sum_{i=0}^{n} \beta_i B_i^n(t), \tag{1.23}$$

$$\gamma(t) = \sum_{i=0}^{n} \gamma_i B_i^n(t), \qquad \delta(t) = \sum_{i=0}^{n} \delta_i B_i^n(t). \tag{1.24}$$

Further, our blending functions are also polynomials:

$$F_1(t) = 1 - F_0(t) = \sum_{i=0}^{n} \phi_i B_i^n(t). \tag{1.25}$$

We will present three main theoretical results about sufficient or necessary conditions. The first result is related to the tangent vectors which can be expressed with the control points α_i, β_i, γ_i, δ_i of the bounding curves. The second approach computes

$$J_{pq} := \sum_{\substack{i+k=p\\ j+l=q}} C(i,j,k,l) \frac{\binom{n}{i}\binom{n}{k}}{\binom{2n}{i+k}} \frac{\binom{n}{j}\binom{n}{l}}{\binom{2n}{j+l}} > 0 \tag{1.26}$$

for all $p, q = 0, \cdots, 2n$ where $C(i,j,k,l)$ involves some determinant computation related to the control points. That first condition is very easy to check but it fails to give answers when the bounding curves become complicated. The second approach is more reliable than the first one but it is more computationally expensive because the above formula involves many computations.

In order to reduce the computational costs while retaining functionality, we use subdivision techniques in the third approach. In fact, we use polar forms to derive necessary and sufficient conditions. More precisely, the blossom function corresponding to the Jacobian which is a polynomial for our chosen blending functions provides an adaptive algorithm that allows us to locate positions where we should apply subdivision.

From our three theoretical results, three practical methods follow. We will show their performance in practical cases from which we observe that the method based on subdivision is reliable and faster than the first two.

Coons patches may suffer from an overspill of the isolines. In order to get rid of the overspill phenomenon, we use Gordon patches,

$$\mathbf{x}(u,v) := \sum_{i=0}^{M} \mathbf{g}_i(v)\varphi_i(u) + \sum_{j=0}^{N} \mathbf{f}_j(u)\psi_j(v) - \sum_{i=0}^{M}\sum_{j=0}^{N} \mathbf{x}_{ij}\varphi_i(u)\psi_j(v). \tag{1.27}$$

We will describe methods of finding the internal points $\mathbf{x}_{ij}$ and curves $\mathbf{f}_j$, $\mathbf{g}_i$ to be interpolated by the Gordon patch. Further, the theoretical results for the Coons patch can be carried over to the Gordon patch if the blending functions φ_i , ψ_j are properly chosen. If we do not obtain a diffeomorphic map yet after using a Gordon patch, then we subdivide the four-sided domain into a few four-sided subdomains.

1.3.4 Chapter 5: Mesh generation

We are interested in generating a surface mesh which approximates a set of parametric surfaces $\{\mathbf{S}_r\}_{r\in\Lambda}$. As we have discussed in section 1.1.2, we want to have the properties (A1) and (A2). Additionally, we want to have nicely shaped triangles as specified in (1.18). The ideal case would be to have a mesh such that all triangles are equilateral. Since that ideal is generally impossible to obtain because the discretizations of the boundary curves are not necessarily uniform, we try to have triangles which are as equilateral as possible. In other words, the edge lengths should change very smoothly. If we impose that the edge size is a harmonic function, then that objective is obtained. We have chosen the Delaunay technique for two reasons. First, it has a well-known nice property which tries to have a triangulation which maximizes the smallest angles. Additionally, the Delaunay triangulation can be used to control the size of the edges by splitting all edges having lengths larger than the ideal edge sizes.

Each surface $\mathbf{S}_r$ is supposed to be the image of a parametric function of a 2D domain $\mathbf{D}_r$. The generation of a mesh on $\mathbf{S}_r$ consists in finding a 2D-mesh in $\mathbf{D}_r \subset \mathbf{R}^2$ which is then mapped to $\mathbf{S}_r$. For every $\mathbf{S}_r$, we start from an initial coarse mesh on $\mathbf{D}_r$ which is refined recursively according to Delaunay techniques. In other words, we split an edge $[\mathbf{a},\mathbf{b}]$ of a 2D mesh if the distance with respect to the inner metric between $\mathbf{a}$ and $\mathbf{b}$ is larger than some ideal edge length ρ:

$$d(\mathbf{a},\mathbf{b}) := \sqrt{\overrightarrow{\mathbf{ab}}^T T \overrightarrow{\mathbf{ab}}} > \rho \qquad \text{with} \qquad T := 0.5(I_{\mathbf{a}} + I_{\mathbf{b}}), \tag{1.28}$$

where $I_{\mathbf{a}}$ and $I_{\mathbf{b}}$ are the first fundamental forms at $\mathbf{a}$ and $\mathbf{b}$.

Consider two triangles $[\mathbf{a},\mathbf{b},\mathbf{c}]$ and $[\mathbf{a},\mathbf{c},\mathbf{d}]$ of the 2D mesh. We flip the edge $[\mathbf{a},\mathbf{c}]$ into $[\mathbf{d},\mathbf{b}]$ if the generalized Delaunay angle criterion with respect to the inner metric is fulfilled:

$$\|\overrightarrow{\mathbf{bc}} \times \overrightarrow{\mathbf{ba}}\|(\overrightarrow{\mathbf{da}}^T T \overrightarrow{\mathbf{dc}}) < \|\overrightarrow{\mathbf{da}} \times \overrightarrow{\mathbf{dc}}\|(\overrightarrow{\mathbf{cb}}^T T \,\overrightarrow{\mathbf{ba}}). \tag{1.29}$$

As for the generation of the initial coarse mesh, we use triangle chopping method. That is to say, we chop off triangles repeatedly from a polygon by inserting internal cuts. Therefore, the resulting coarse mesh has only nodes that lie on the boundary. We will use the 2-ear theorem to show existence of such an initial coarse triangulation.

Since the ideal edge length ρ in (1.28) is not known a-priori, we would like to determine it by using the Laplace-Beltrami operator

$$\Delta_{\mathbf{S}}\rho = -\frac{1}{\sqrt{g}}\frac{\partial}{\partial u_j}\left(\sqrt{g}g_{ij}\frac{\partial F}{\partial u_i}\right). \tag{1.30}$$

In order that the variation of the mesh size ρ is smooth, we will show the numerical method of solving

$$\left\{ \begin{array}{rcll} -\Delta_{\mathbf{S}}\rho & = & 0 & \text{in } \ \mathbf{S} \\ \rho & = & \rho_{\text{bound}} & \text{on } \ \partial\mathbf{S}\,. \end{array} \right. \tag{1.31}$$

Since we have several surfaces $\mathbf{S}_r$ $(r = 1, \cdots, n)$, we apply the above technique to each $\mathbf{S}_r$. In order not to have hanging nodes in the interfaces of the surface $\mathbf{S}_r$, the polygonal approximation of the boundaries of $\{\mathbf{S}_r\}_r$ has to be done before the actual mesh generation.

1.3.5 Chapter 6: Paving of meshes by four-sided patches

We will consider the approximation of a given surface mesh $\mathcal{M}$ by a set of B-spline surfaces $\cup_{r=1}^{M}\mathcal{S}_r$. This method is very useful when we have a mesh-free numerical solver of integral equations but the input 2-manifold that we have at our disposal is given as a mesh. Our method consists in flattening the whole surface $\mathcal{M}$ so that it becomes a single planar polygon. In other words, we need a parameterization from a planar polygon $\mathbf{P}$ to $\mathcal{M}$. After decomposing the mesh $\mathcal{M}$ into several four-sided submeshes $\mathcal{M}_r$, every submesh $\mathcal{M}_r$ is approximated by a B-spline S_r given by:

$$\mathbf{X}_r(u,v) = \sum_{i,j=0}^{n} \mathbf{d}_{ij}^{r} N_i^k(u) N_j^k(v). \qquad (1.32)$$

An important objective that we want is that the collection of surfaces $\{\mathcal{S}_r\}$ is globally continuous. That is, two neighboring B-spline patches must have $\mathcal{C}^0$-joint. Our method of achieving that goal consists in approximating the piecewise linear curves $\mathcal{D}_T$ which bound the submeshes by B-spline curves

$$\mathcal{C}_T(t) = \sum_{i=0}^{n} \delta_i^T N_i^k(t). \qquad (1.33)$$

We use chord-length parameterization to have a mapping from an interval to $\mathcal{D}_T$. We impose later that the curves $\mathcal{C}_T$ are interpolated by the incident B-splines. In other words, the boundary de-Boor points of $\mathbf{X}_r$ are taken from those of the bounding B-spline curves. If any parameterization does not satisfy the Schoenberg-Whitney condition, then we have to perform a resampling.

Chapter 2

IGES FORMAT AND SURFACE STRUCTURES

Abstract: The purpose of this chapter is to address practical aspects about geometric storage and exchange. First, we would like to give some description of the CAD interface IGES because our approaches and our implementations are mainly based on this format. Therefore, some understanding of the format will help understand our future descriptions. It is impossible to describe the IGES format thouroughly in the scope of this chapter. Therefore, we are only going to describe the file organizations and some selected entities which we have supported in our implementations. We will also report on our realizations of loading and parsing of an IGES file along with the accompanying data structures. Furthermore, we describe CAD flaws which often make the implementations of geometric methods difficult. In the second part, we propose a method for finding the structures of a given surface in which we distinguish topological and metric information. Readers who are only interested in abstract geometric methods can skip this chapter.

2.1 Geometric storage and transmission

2.1.1 CAD interface usage

Originally, technical drawings have been processed manually. Since the introduction of computer assisted modelling of 2D and 3D components, the digitization of the produced parts has become a necessity. Nowadays, a CAD interface is used for multiple purposes which can be sorted into three categories [118] that we want to describe briefly:

(1) Interface between CAD systems,

(2) Automatic controls of CAM systems,

(3) Subsequent numerical computations.

First, a CAD interface is used to store and transmit technical drawings. In that process, a CAD translator serves as digitizer of finished drawing from a CAD system. Conversely, it is used to parse CAD interfaces so that they can be loaded into CAD systems.

A CAD interface supports equally mechanical constructions or CAM (Computer Aided Manufacturing). In other words, it plays an important role in NC(Numerically Controlled) programming of 2-axis productions [68]. The stored geometric models are transformed into NC programming languages.

Finally, one needs CAD interfaces for providing geometries which are needed in numerical modelings. These methods include FEM, BEM or other solvers of problems originating from engineering such as aerodynamics, structural mechanics, electrical computations. Although specialists in numerical computations have very efficient approaches to solve engineering problems, they often use manually generated geometries. This fact restricts the performance of numerical methods to the level of theories because it is very tedious to create practical geometries manually.

2.1.2 IGES as a CAD neutral format

The storage and transmission of technical drawings have demonstrated themselves to be necessary. Furthermore, they should be feasible not only within an individual organization but also to large communities. This requirement has led to a way to transfer geometric components in neutral formats. That is, a format which is not specific to any individual organization and which is understood and acceptable by all CAD systems. The advantage of a neutral format over a direct transmission is that the number of translations reduces considerably. Conceived in 1979 by a combination of people including ICAM and NBS representatives [57], IGES has become the first CAD interface. During its phase of conception, it was termed "Interim" Graphics Exchange Specification. The first release of IGES as an approved ANSI standard took place in 1981 [127]. In the version IGES 1.0, it was labeled *Initial Graphics Exchange Specification.* That first version mainl contained supports for drafting of technical drawings and it included only 34 entities. More entities have been introduced in the version IGES 2.0 which supported FEM (Finite Element Method) modelings and electronic printed wiring boards. In the version 3.0, more applications about architectures and constructions [126] were incorporated.

2.2 IGES file structure

In this section, we would like to give a description of an IGES file format. While reading this short description, we recommend that the readers compare the explanation with Fig. 2.1 which shows a simple example of an IGES file.

2.2.1 Entity classification and data types

We distinguish two main entity categories in an IGES file: geometric and nongeometric entities. The first group contains information which is required to describe shapes such as curves, surfaces, solids and relationships between them. The second group is needed for other graphical or computational purposes that include color properties in RGB, CMY, or HLS format, or physical units such as measures of mass, time, temperature, luminous intensity which are needed in physical simulations or numerical applications. Descriptive properties such as text fonts are also classified in the non-geometric entities. IGES standard supports several data types which include strings, integers, and real numbers. Real numbers are usually represented in the form `rDs`, a floating point number r followed by the character 'D' followed by a signed integer s as `9.807693D-1` or `1.5D2`. This represents a real number r multiplied by ten to the power of s. A string has the format `lHf` in which the integer `l` determines the length of the string to be described. The real string is shown after the letter `H` as `11HProEngineer`. Such a representation could prove very helpful if we use such formatted programming language as FORTRAN.

Entity	ID number	IGES-code
Line	110	`LINE`
Circular arc	100	`ARC`
Polynomial/rational B-spline curve	126	`B_SPLINE`
Composite curve	102	`CCURVE`
Surface of revolution	120	`SREV`
Tabulated cylinder	122	`TCYL`
Polynomial/rational B-spline surface	128	`SPLSURF`
Trimmed parametric surface	144	`TRM_SRF`
Transformation matrix	124	`XFORM`

Table 2.1: Implemented curve, surface and transformation entities

In the IGES format, a *parameter* is a value which can be integer, string, or floating point and which describes some information in the global section or parameter data. Those parameters are separated by a symbol known as *parameter delimiter* which can be specified or defined in the global section and which has comma (,) as default value. Every entity which is found in the directory entry is detailed by one *record* which is a sequence of parameters. The symbol separating two records is the *record delimiter* whose default value is a semicolon (;).

2.2.2 IGES file organization

The number of lines of an IGES file depends on the geometric information to be stored. The lines are generally partitioned into five main sections:

(1) Start section,

(2) Global section,

(3) Directory entry,

(4) Parameter data,

(5) Terminate section.

The structure of an IGES file is always organized in 80 columns whose corresponding specific roles are split into three parts. First, columns 1 till 72 contain the most valuable parameter and record values pertaining to the digitized geometry. Those columns include information which varies according to the section to be described. Second, column 73 contains one letter which specifies the current section. Thus, the five IGES-sections that we have described above are identified respectively by the letters '**S**', '**G**', '**D**', '**P**', and '**T**'. Finally, columns 74 till 80 contain integer data which are right-justified and which indicate the line numbers of every section. In simple words, an IGES file has the structure displayed in Table 2.2.

Now, we would like to summarize the purpose if each section of an IGES file. The start section which should have at least one line is a human-readable section in which you can write anything like the directory location of the file. This section usually contains the comments of the sender to the receiver.

In the global section, we find general information such as how to read the current file, where, when and by whom was the file generated. It can also specify the parameter delimiter as well as the record delimiter.

The actual description of the entities takes place in the directory entry and parameter data sections. The directory entry gives in general an overview of the different components of the stored geometry and it points to records in the Parameter Data section which contains the complete information about all parameters. For instance, in the directory entry we can see that we have to deal with NURBS surfaces while the information about the control points, knot sequence and weights cannot be found in the Directory Entry. Every entity has its own identification number which can range from 0 to as many as 514. In Table 2.1,

1 ... 72	73	74 ... 80
START SECTION	S S S	1 2 ...
GLOBAL SECTION	G G G	1 2 ...
DIRECTORY ENTRY	D D D	1 2 ...
PARAMETER DATA	P P P	1 2 ...
TERMINATE SECTION	T	1

Table 2.2: IGES file structure

Field	Columns	Section
1	1-8	Start
2	9-16	Global
3	17-24	Directory entry
4	25-32	Parameter data

Table 2.3: Four fields in terminate section

we summarize the entity numbers of a few important geometries. It is in the Parameter Data section that we can find the actual values of all parameters. The content of the parameter data section varies from one entity to another but it has in general the following structure

```
entity number,parameter1,parameter2,...,parameterN;
```

The terminate section consists only of a single line which does not use columns 33-72 and which describes the lengths of the former four sections as described in Table 2.3.

2.2.3 Shortcomings and proposed remedies

In the following discussion, we summarize the two negative properties that CAD specialists usually experience with the IGES format.

The first main problem with which many users complain [57, 118, 126] about IGES is its volume. IGES files which store real-world geometries such as complex mechanical parts are extremely large. It could take several hours to load and interpret them. This problem about voluminous file is often attributed to the structure of the IGES files which repeat in some cases the same information. For instance, a curve is sometimes described several times. Parameters that are described in the Directory Entry section are in some events still retrieved in the Parameter Data section. Some remedies have already been proposed to overcome such difficulties. One solution that is described in the IGES 5.3 [119] is the introduction of compressed IGES. This avoids the redundancy in the Directory Entry and the Parameter Data sections. Those two sections are merged into a single one which is termed *Data section*. A new additional one line section, the *Flag section*, is introduced in order to indicate that we deal with the compressed format. The compressed format has another way of storage which reduces the space requirement and which facilitates the access to the entities. There is an interface software [119] for converting from the initial IGES format to the compressed one and conversely. Another proposed remedy [126] to the volume of the IGES file is the IIF format (Internal IGES file) which is used in several IGES translators to serve as an interface between the IGES file and the CAD/CAM system. The IIF formats reorganize an IGES file into several Internal IGES files which have the advantage of providing a more convenient way of accessing geometric entities.

The second shortcomming is related to selective implementations of the whole IGES format. On account of the large number of entities in the IGES standard, users often select some entities which are adapted to their own concerns. Thus, they implement only their applications with those selected entities. Repeated applications of such a process can cause loss of information. Further, IGES can represent the same geometric information in a lot of ways which could accentuate the problem of data loss.

```
 PTC IGES file: H:\IGES_files\open_cylinder.igs                         S      1
1H,,1H;,7HPRT0001,31HH:\IGES_files\open_cylinder.igs,                   G      1
49HPro/ENGINEER by Parametric Technology Corporation,7H2001150,32,38,7, G      2
38,15,7HPRT0001,1.,1,4HINCH,32768,0.5,13H021014.152641,0.00865991,      G      3
86.6025,8Hmaharavo,7HUnknown,10,0,13H021014.152641;                     G      4
     124       1       1       1       0       0       0       001000000D      1
     124       0       0       1       0                   XFORM       1D      2
     100       2       1       1       0       0       1       001010000D      3
     100       0       0       1       0                     ARC       1D      4
     124       3       1       1       0       0       0       001000000D      5
     124       0       0       1       0                   XFORM       2D      6
     100       4       1       1       0       0       5       001010000D      7
     100       0       0       1       0                     ARC       2D      8
     110       5       1       1       0       0       0       001010000D      9
     110       0       0       1       0                    LINE       1D     10
     124       6       1       1       0       0       0       001000000D     11
     124       0       0       1       0                   XFORM       3D     12
     100       7       1       1       0       0      11       001010000D     13
     100       0       0       1       0                     ARC       3D     14
     124       8       1       1       0       0       0       001000000D     15
     124       0       0       1       0                   XFORM       4D     16
     100       9       1       1       0       0      15       001010000D     17
     100       0       0       1       0                     ARC       4D     18
     110      10       1       1       0       0       0       001010000D     19
     110       0       0       1       0                    LINE       2D     20
     110      11       1       1       0       0       0       001010000D     21
     110       0       0       1       0                    LINE       3D     22
     110      12       1       1       0       0       0       001010000D     23
     110       0       0       1       0                    LINE       4D     24
     120      13       1       1       0       0       0       001010000D     25
     120       0       0       1       0                    SREV       1D     26
     110      14       1       1       0       0       0       001010000D     27
     110       0       0       1       0                    LINE       5D     28
     102      15       1       1       0       0       0       001010000D     29
     102       0       0       1       0                  CCURVE       1D     30
     110      16       1       1       0       0       0       001010500D     31
     110       0       0       2       0                    LINE       6D     32
     110      18       1       1       0       0       0       001010500D     33
     110       0       0       2       0                    LINE       7D     34
     110      20       1       1       0       0       0       001010500D     35
     110       0       0       2       0                    LINE       8D     36
     110      22       1       1       0       0       0       001010500D     37
     110       0       0       2       0                    LINE       9D     38
     102      24       1       1       0       0       0       001010500D     39
     102       0       0       1       0                  CCURVE       2D     40
     142      25       1       1       0       0       0       001010500D     41
     142       0       0       1       0                  UV_BND       1D     42
     144      26       1       1       0       0       0       000000000D     43
     144       0       0       1       0                 TRM_SRF       1D     44
     110      27       1       1       0       0       0       001010000D     45
     110       0       0       1       0                    LINE      10D     46
     110      28       1       1       0       0       0       001010000D     47
     110       0       0       1       0                    LINE      11D     48
     120      29       1       1       0       0       0       001010000D     49
     120       0       0       1       0                    SREV       2D     50
     110      30       1       1       0       0       0       001010000D     51
     110       0       0       1       0                    LINE      12D     52
     102      31       1       1       0       0       0       001010000D     53
     102       0       0       1       0                  CCURVE       3D     54
     110      32       1       1       0       0       0       001010500D     55
     110       0       0       2       0                    LINE      13D     56
     110      34       1       1       0       0       0       001010500D     57
     110       0       0       2       0                    LINE      14D     58
     110      36       1       1       0       0       0       001010500D     59
     110       0       0       2       0                    LINE      15D     60
     110      38       1       1       0       0       0       001010500D     61
     110       0       0       1       0                    LINE      16D     62
     102      39       1       1       0       0       0       001010500D     63
     102       0       0       1       0                  CCURVE       4D     64
     142      40       1       1       0       0       0       001010500D     65
     142       0       0       1       0                  UV_BND       2D     66
     144      41       1       1       0       0       0       000000000D     67
     144       0       0       1       0                 TRM_SRF       2D     68
     406      42       1       1       0       0       0       001000000D     69
     406       0       0       1      15                    PROP       1D     70
     402      43       1       1       0       0       0       000000300D     71
     402       0       0       2       7                   LAYER       1D     72
     406      45       1       1       0       0       0       001000000D     73
     406       0       0       1      15                    PROP       2D     74
     402      46       1       1       0       0       0       000000300D     75
     402       0       0       2       7                   LAYER       2D     76
124,-1D0,0D0,0D0,7.5D1,0D0,-1D0,0D0,1.5D2,0D0,0D0,1D0,0D0;             1P      1
100,0D0,0D0,0D0,2.5D1,0D0,-2.5D1,0D0;                                  3P      2
124,1D0,0D0,0D0,7.5D1,0D0,-1D0,0D0,1.5D2,0D0,0D0,-1D0,-5D1;            5P      3
100,0D0,0D0,0D0,2.5D1,0D0,-2.5D1,0D0;                                  7P      4
110,5D1,1.5D2,0D0,5D1,1.5D2,-5D1;                                      9P      5
124,1D0,0D0,0D0,7.5D1,0D0,1D0,0D0,1.5D2,0D0,0D0,1D0,0D0;              11P      6
100,0D0,0D0,0D0,2.5D1,0D0,-2.5D1,0D0;                                 13P      7
124,-1D0,0D0,0D0,7.5D1,0D0,1D0,0D0,1.5D2,0D0,0D0,-1D0,-5D1;           15P      8
100,0D0,0D0,0D0,2.5D1,0D0,-2.5D1,0D0;                                 17P      9
110,1D2,1.5D2,0D0,1D2,1.5D2,-5D1;                                     19P     10
110,7.5D1,1.5D2,0D0,7.5D1,1.5D2,1D0;                                  21P     11
110,1D2,1.5D2,-5.1D1,1D2,1.5D2,1D0;                                   23P     12
120,21,23,3.078760800518D0,6.346017160251D0;                          25P     13
110,5D1,1.5D2,-5D1,5D1,1.5D2,0D0;                                     27P     14
102,4,3,19,7,27;                                                      29P     15
110,9.807692307692D-1,3.141592653590D0,0D0,9.807692307692D-1,         31P     16
6.283185307180D0,0D0;                                                 31P     17
110,9.807692307692D-1,6.283185307180D0,0D0,1.923076923077D-2,         33P     18
6.283185307180D0,0D0;                                                 33P     19
110,1.923076923077D-2,6.283185307180D0,0D0,1.923076923077D-2,         35P     20
3.141592653590D0,0D0;                                                 35P     21
110,1.923076923077D-2,3.141592653590D0,0D0,9.807692307692D-1,         37P     22
3.141592653590D0,0D0;                                                 37P     23
102,4,31,33,35,37;                                                    39P     24
142,0,25,39,29,1;                                                     41P     25
144,25,1,0,41;                                                        43P     26
110,7.5D1,1.5D2,0D0,7.5D1,1.5D2,1D0;                                  45P     27
110,1D2,1.5D2,-5.1D1,1D2,1.5D2,1D0;                                   47P     28
120,45,47,-6.283185307180D-2,3.204424506662D0;                        49P     29
110,1D2,1.5D2,-5D1,1D2,1.5D2,0D0;                                     51P     30
102,4,13,9,17,51;                                                     53P     31
110,9.807692307692D-1,0D0,0D0,9.807692307692D-1,                      55P     32
3.141592653590D0,0D0;                                                 55P     33
110,9.807692307692D-1,3.141592653590D0,0D0,1.923076923077D-2,         57P     34
3.141592653590D0,0D0;                                                 57P     35
110,1.923076923077D-2,3.141592653590D0,0D0,1.923076923077D-2,         59P     36
0D0,0D0;                                                              59P     37
110,1.923076923077D-2,0D0,0D0,9.807692307692D-1,0D0,0D0;              61P     38
102,4,55,57,59,61;                                                    63P     39
142,0,49,63,53,1;                                                     65P     40
144,49,1,0,65;                                                        67P     41
406,1,17H02___PRT_ALL_AXES;                                           69P     42
402,18,3,7,9,13,17,19,21,23,25,29,39,43,45,47,49,53,63,67,0,1,        71P     43
69;                                                                   71P     44
406,1,18H06___PRT_ALL_SURFS;                                          73P     45
402,18,3,7,9,13,17,19,21,23,25,29,39,43,45,47,49,53,63,67,0,1,        75P     46
73;                                                                   75P     47
S      1G      4D     76P     47                                        T      1
```

Figure 2.1: Sample of an IGES file

2.3 Segment separators

In this section, we would like to discuss about the way a planar multiply connected region $\mathcal{D}$ is often organized with IGES. Such a region $\mathcal{D}$ is delineated by an exterior boundary $\mathbf{B}_{\text{out}}$ and N internal boundaries $\mathbf{B}_{\text{in}}^{j}$. The exterior boundary $\mathbf{B}_{\text{out}}$ is traversed counter-clockwise while the interior ones clockwise. All curves defining those boundaries are represented as composite curves. A point which resides between two curved segments will be referred to as *segment separators* as illustrated in Fig. 2.2(a). Later in our algorithms, we will have to insert more nodes on $\mathbf{B}_{\text{out}}$ or $\mathbf{B}_{\text{in}}^{j}$.

Now we would like to describe a nice property of the segment separators in IGES representation. Consider two adjacent surfaces S_i and S_j which are the images of the planar multiply connected regions $\mathcal{D}_i$ and $\mathcal{D}_j$ by the functions ψ_i and ψ_j. Suppose that S_i and S_j share a curve $\mathcal{C}$ which is bounded by two points A, $B \in \mathbf{R}^3$. The IGES format represents $\mathcal{D}_i$ and $\mathcal{D}_j$ such that there are segment separators a_i, $b_i \in \mathcal{D}_i$ and a_j, $b_j \in \mathcal{D}_j$ with

$$A = \psi_i(a_i) = \psi_j(a_j), \qquad B = \psi_i(b_i) = \psi_j(b_j). \tag{2.1}$$

In the splitting into four-sided regions, segment separators which have smooth joints will have to be treated as reflex vertices. That is, we have to insert cuts emanating from these points so that we do not have any four-sided regions with smooth corners.

2.4 Features of the implementation

This section will be occupied by the descriptions of the difficulties that we encountered during the implementations with IGES format. Although this section has no direct theoretical interests, we would like to report on the practical treatments of digitized geometries without which no theoretical methodology is applicable. The problem about using an IGES file for storing the input geometric components is related to the loading, parsing and evaluating processes. During the treatments of the selected entities in Table 2.1, the following tasks have to be implemented.

(a) Automatic extraction of the size of the entities which require large storage: we need the different integer values which indicate the volume required in the data structures. That is a necessary step in order that accurate memory allocations and deallocations can be feasible. This step has to happen before the real entity extraction can take place.

(b) A large number of simple routines for transforming the IGES parameters to acceptable inputs in the code: those routines include location of the parameter delimiters and record delimiters so that appropriate records could be withdrawn from IGES files. This is necessary in order to access a specified record in the

Parameter Data section which is pointed by an entity in the Directory Entry section.

(c) Extraction of relevant entities from the IGES file: one has to write a single routine for each entity because the IGES description of parameter sequences varies from one entity to another in the Parameter Data section.

(d) Loading of the parameters of the entities from the IGES file to the C-data structures: we need convertion of the IGES data formats into formats which are understood by the C programming language. We have generated a data structure for every entity. The following is for example the data structure that is used to store a NURBS surface. The other entities can be structured in a similar way.

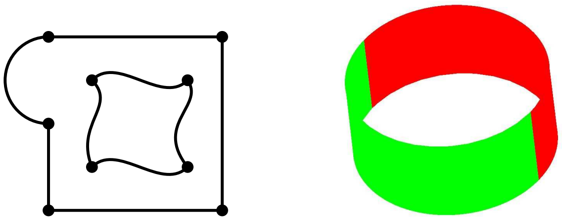
```
typedef struct nurbs_surface{
      int nu;           //order in u-variable
      int nv;           //order in v-variable
      int ku;           //smoothness in u-variable
      int kv;           //smoothness in v-variable
      point **d;        //3D coordinates of control points
      double **w;       //weights
      double *tau_u;    //knot sequence along u-direction
      double *tau_v;    //knot sequence along v-direction
      double u0;        //[u0,u1]=interval of def along u-dir
      double u1;        //[u0,u1]=interval of def along u-dir
      double v0;        //[v0,v1]=interval of def along v-dir
      double v1;        //[v0,v1]=interval of def along v-dir
      int prop1;        //open or closed in u-direction
      int prop2;        //open or closed in v-direction
      int prop3;        //rational or polynomial
      int prop4;        //periodic or not in u-direction
      int prop5;        //periodic or not in v-direction
}nurbs_surface;
```

(f) Evaluation functions: one step that cannot be neglected is the entity evaluations because one has to evaluate the loaded entities repeatedly in our subsequent geometric algorithms. We have to implement an evaluation routine for each and every given data structure. For instance, we need a de-Boor algorithm [39] to evaluate a B-spline surface.

2.5 Practical experiences and CAD flaws

We would like to describe in this section some important information that we gained from the experience about implementing with the IGES format.

(a) There are situations where the input CAD data have imperfections [125].



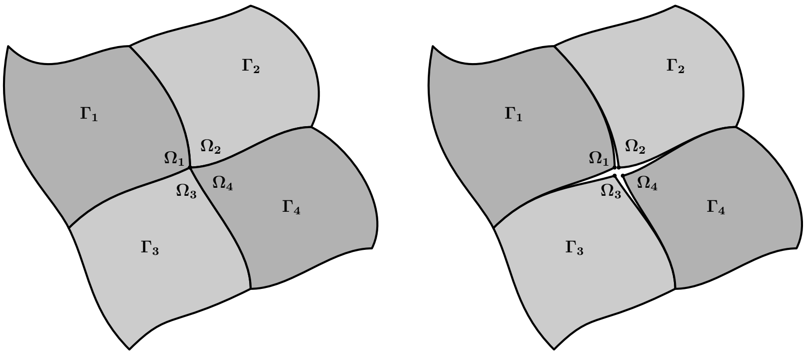
(a) (b)

Figure 2.2: (a)The large points represent segment separators, (b)A cylinder represented as two half-cylinders

That is, the digitized geometry does not represent exactly the theoretical expectation of the users. For instance, there are sometimes two surfaces which are supposed to be theoretically adjacent but there is in practice a gap between them. That is also the case for nodes or corner points which should be shared by some neighboring surfaces but which do not coincide in practice as in Fig. 2.3. Those CAD flaws could cause numerical instabilities because adjacency information could become wrong or incomplete. One can read [86] for an in-depth studies of CAD imperfections along with methodologies of repairing them. There are commercial softwares which can detect such imperfections and which correct them afterwards. In our implementations, the input CAD data have mostly been generated by Pro-Engineer which usually produces good CAD geometries. Therefore, our approach demonstrates itself to be very efficient for the 3D surfaces that we created ourselves. There are however some geometries which we receive from other sources and which present non-negligible CAD inaccuracies. Without the dispositions of CAD repair softwares, those data have to be treated individually by prescribing some threshold ε and by making some assumption that all curves or points within ε-accuracy coincide. Such an approach makes our current implementation only semi-automatic because such a threshold could vary from an IGES file to another. Therefore, some user-interaction is sometimes unavoidable.

(b) For describing periodic geometric components such as surfaces of revolution, it is theoretically conceivable to represent them as one single patch. By investigating a large number of geometric data, we have noticed that IGES represents such surfaces as non-periodic components in which they are already split. For instance, a cylinder is represented as two half-cylinders as in Fig. 2.2(b). Therefore, we will assume that no two edges of a single surface will coincide throughout this thesis. This assumption will be implicitly useful during the topological description of composite surfaces.

(c) According to the practical experiences that we have gained from implementing with IGES format, both trimmed surfaces as well as theoretically untrimmed ones are often represented as trimmed entities. That is, they are both assigned the IGES entity number 144 with the code `TRM_SRF`. If the theoretical expectation



(a) (b)

Figure 2.3: CAD imperfection: the nodes $\mathbf{\Omega}_1$, $\mathbf{\Omega}_2$, $\mathbf{\Omega}_3$, $\mathbf{\Omega}_4$ are supposedly the same: (a) theoretical expectation (b) practical stored geometry.

defines an untrimmed surface $\mathcal{S}$ on the parameter domain $[a, b] \times [c, d]$ then the IGES descriptor extends this region as $[a - \varepsilon, b + \varepsilon] \times [c - \mu, d + \mu]$ and $\mathcal{S}$ is considered as a trimmed surface such that the trimming curve is the rectangle consisting of the following corners:

$$A := (a, c) \qquad B := (b, c) \qquad C := (b, d) \qquad D := (a, d). \tag{2.2}$$

2.6 Surface structures

2.6.1 Topological structure

In this section, we would like to describe how to obtain the structure which represents the adjacencies of the surfaces $\mathcal{S}_i$ belonging to a given CAD model. Our objective consists in finding some practical way of identifying if two surfaces are adjacent.

In order that two surfaces S_i and S_j are adjacent, three different cases can be distinguished as in Fig. 2.4. The first one applies when two boundary curves $\mathcal{C}_i$ and $\mathcal{C}_j$ belonging respectively to S_i and S_j exactly coincide. The second case occurs when the first endpoint of $\mathcal{C}_i$ coincides with an endpoint of $\mathcal{C}_j$ and the second endpoint of $\mathcal{C}_i$ resides strictly within $\mathcal{C}_j$. The third case is that the curve $\mathcal{C}_i$ resides strictly inside $\mathcal{C}_j$. We are interested in finding a practical way of detecting those three cases.

In order to facilitate the presentation, we are only going to describe the first case. The other two cases can be treated in a similar manner. We will suppose that the curves $\mathcal{C}_i$ and $\mathcal{C}_j$ are defined parametrically from the intervals $[a_i, b_i]$ and $[a_j, b_j]$

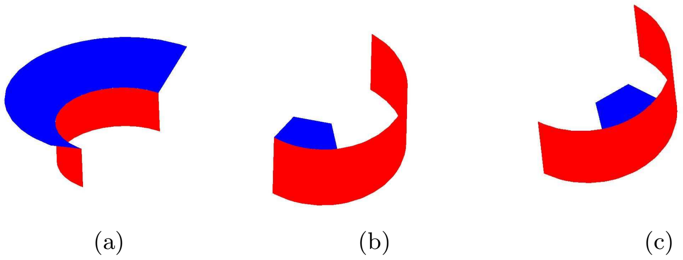


Figure 2.4: Three types of adjacent patches

respectively. If $\mathcal{C}_i$ and $\mathcal{C}_j$ share two endpoints, the two surfaces S_i and S_j are already adjacent for most mechanical objects. But it is worth mentioning that it is possible that only the two endpoints coincide while the surfaces S_i and S_j are not at all adjacent. Such a situation can be observed in Fig. 2.5(a) where one half-cylinder and a toroidal portion only meet at two corners. In order to exclude such a situation, we perform a further test. That is, we check if the two curves share an internal vertex, i.e if there exists some point $\mathbf{X} \in \mathbf{R}^3$ with

$$\mathbf{X} = \mathcal{C}_i(t_i) = \mathcal{C}_j(t_j) \qquad t_i \in]a_i, b_i[\quad \text{and} \quad t_j \in]a_j, b_j[. \tag{2.3}$$

Let us describe briefly how the condition (2.3) can be checked in practice. We choose t_i as the midpoint of the first interval $[a_i, b_i]$ and we define as $\mathbf{X}$ the image $\mathcal{C}_i(t_i)$. Therefore, the test (2.3) amounts to determining the existence of t_j. The easiest way of finding that is to select some uniform samples $\{s_k\}$ in the interval $]a_j, b_j[$ and to verify if some images $C_j(s_k)$ approximate $\mathbf{X}$ within a prescribed accuracy. But such a method is not quite efficient because it could require a lot of function evaluations. Our preferred method is an adaptive one in which we search for a sequence of decreasing subintervals $I_k = [\sigma_k, \tau_k]$

$$I_0 \supset I_1 \supset I_2 \supset \cdots \tag{2.4}$$

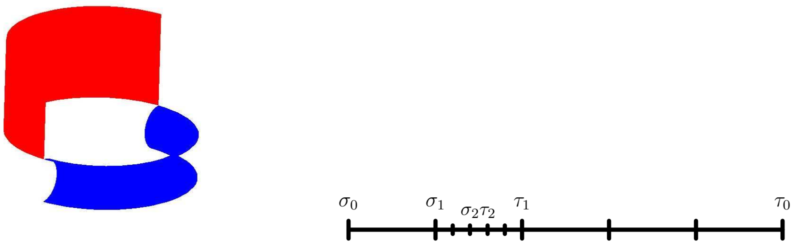
whose lengths $\mu(I_k)$ are geometrically decreasing:

$$\mu(I_{k+1}) < \lambda\mu(I_k) \qquad \text{with} \quad \lambda \in]0, 1[. \tag{2.5}$$

The approach consists first in initializing I_0 to be the interval $[a_j, b_j]$ itself and in prescribing a splitting parameter $M \geq 2$. If $I_k = [\sigma_k, \tau_k]$ is known, we split it into M subintervals and we select as $I_{k+1} = [\sigma_{k+1}, \tau_{k+1}]$ the subinterval which is the closest one to $\mathbf{X}$. We repeat this iteration until some prescribed accuracy ε is obtained:

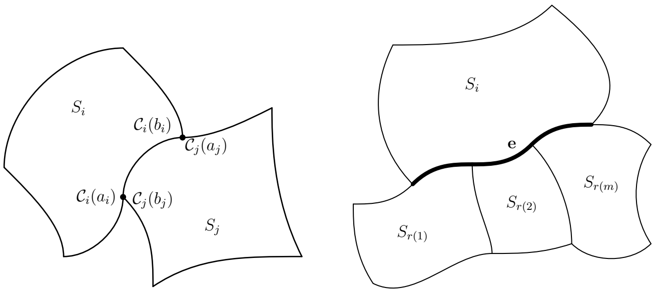
$$\|\mathcal{C}_j(m_k) - \mathbf{X}\| < \varepsilon \quad \text{where} \quad m_k := 0.5(\sigma_k + \tau_k). \tag{2.6}$$

In general, we do not need to choose an accuracy ε which is too fine because the objective is to detect adjacency and not to have an accurate solution of criterion



(a) (b)

Figure 2.5: (a)Two surfaces share two points but they are not adjacent (b)Adaptively decreasing intervals



(a) (b)

Figure 2.6: (a)Metric information (b)The edge **e** of S_i is completely covered

(2.3). In our implementations, an accuracy ε of order 0.01 demonstrates itself to be enough.

For the second case, we can proceed as in the first case to test for the curve membership. A further test of existence of an internal common point is theoretically conceivable but in our practical tests we have observed that the two endpoints suffice. The third case can be treated also in a similar manner with the exception that you have to test for curve membership for the two endpoints of $\mathcal{C}_i$.

2.6.2 Metric structure

Apart from topological information, it is also beneficial to store metric information. While topological structures specify the interrelation between surfaces with the help of *integer* entities, metric ones indicate point coordinates or parameter values at which certain points are passed by curves. For two adjacent patches S_i and S_j we store the time intervals $[a_i, b_i]$ and $[a_j, b_j]$ which determine the superposing curve portions of the patches as in Fig. 2.6(a). That is, if $\mathcal{C}_i$ and $\mathcal{C}_j$ are the curves which delineate the boundary parts of S_i and S_j then $\mathcal{C}_i([a_i, b_i])$ and $\mathcal{C}_j([a_j, b_j])$ coincide. The advantage of having such metric information is that it is easy to verify whether an edge e of a given patch S_i is *completely covered* by other patches $S_{r(1)}$, ..., $S_{r(m)}$ as in Fig. 2.6(b). As a consequence, when we want to perform an adjacency test, we do not need to verify the incidence of such a completely covered edge e upon a patch S_j with j different from $r(1)$,...,$r(m)$. Exploiting the metric structure has a considerable acceleration result in the determination of adjacencies.

2.7 Other CAD interfaces

We have implemented our geometric algorithms with the IGES format but we believe that they can be also used with other CAD interfaces with little modifications. We would like to briefly describe below some well-known CAD interfaces other than IGES.

VDA-FS: The file format VDA-FS or **V**erband **d**er **A**utomobilindustrie-**F**lächen **S**chnittstelle has [33] its origin from the German car manufacturers [105] association VDA. The VDA-FS format is well known in its capability to manipulate free form 3D surfaces which are very frequently involved in car designs. A VDA-FS file can be recognized by its suffix ".vda". In contrast to IGES which has a lot of entities, VDA-FS has only five entities [32].

SET: The file format SET (**S**tandard d'**É**change et de **T**ransfert) started its development in 1983 and it became a French standard for CAD data exchange in 1985. This format has originated from the aircraft industry Airbus.

STEP: The International Organization for Standardization (ISO) has introduced the STEP (**ST**andard of the **E**xchange of **P**roduct model data) format [81] in order to release a CAD exchange model which should be accepted as standard worldwide. While IGES, VDA-FS, SET were mainly used as US, German and French standards respectively, the STEP standard was [116] designed for international use. That facilitates transfer between international partners and customers [17]. Note that STEP is nowadays the most modern CAD interface.

DXF: The format DXF, which stands for **D**rawing e**X**change **F**ormat [4], was originally used to store 3D models created by the CAD-system autoCAD [14, 77]. DXF format represents information in tagged data mode. That is, an integer precedes every data piece. DXF files are available in both ASCII and binary formats. There are nowadays interesting tools for loading a DXF file and use the data from it in OpenGL applications.

Save files: SAT (**S**tandard **A**CIS **T**ext) and SAB (**S**tandard **A**CIS **B**inary) are file formats that are used for storing and transferring data in ACIS [24] which is a very flexible C++ and SCHEME library produced by Spatial Technologies for geometry manipulation purposes. These two formats are also known as save files. Although SAT files are introduced by ACIS, they can also be loaded by some other CAD-systems.

Chapter 3

FOURSIDED SPLITTING

Abstract: The purpose of this chapter is the decomposition of a given trimmed surface into several four-sided subregions. Before treating the general cases, we will first focus on regions with polygonal boundaries. We aim at splitting a given multiply connected polygon into a set of convex quadrilaterals. Then, we will generalize our methods for regions having curved boundaries. Our approach consists in repeatedly removing quadrilaterals from a given polygon. We will first derive theoretical results pertaining to quadrangulation of simple polygons from the usual 2-ear theorem. Unfortunately, the 2-ear theorem holds only for simply connected polygons while we are interested in quadrangulation of polygons with holes. As a consequence, we will introduce a special type of polygon which serves as intermediate step to derive results for multiply connected polygons. For a surface having curved boundaries, we approximate it first by a coarse polygonal region. After quadrilating the resulting polygon, we replace the boundary edges by the initial curves. Furthermore, we will show methods of avoiding boundary interferences. Numerical results are reported to illustrate the approaches. Our benchmarks include CAD objects which come directly from IGES files.

3.1 Introduction

We address the problem of design with trimmed surfaces which are fundamental entities in CAGD. The majority of CAD objects, even among the simplest ones such as closed cylinders, are partly or completely composed of trimmed surfaces. Therefore, completely ignoring treatment of trimmed surfaces is an unacceptable restriction if one wants to deal with real-world CAD data.

Since our splitting method is based upon quadrangulation, we will first focus on quadrilating polygonal regions in the next sections. Decompositions of curved regions will be treated after having a complete insight about multiply connected polygons. There are generally two categories of numerical methods for quadri-

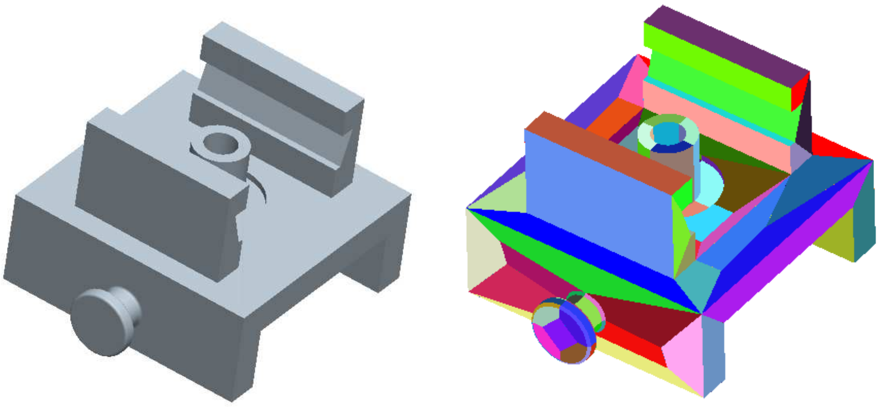

(a) (b)

Figure 3.1: Splitting into four-sided subregions

lating a polygon. The indirect method (see [99, 100] and the references there) consists in generating an initial triangulation which is then converted into a quadrangulation. On the opposite, the second approach generates a quadrangulation directly from the initial polygon [88, 80, 78]. The methods that are mainly discussed in this chapter fall in the second category. In simple words, the fundamental algorithms of our approach consist in removing a quadrilateral from the given polygon by using the constructive proofs of our theorems. We repeat the same process to the remaining polygon recursively until the whole polygon is quadrilated. The process of removing a quadrilateral amounts to introducing some cuts within the polygon. We will try to insert few cuts in order to eventually obtain few quadrilaterals.

In the next sections, we will describe the problem setting more accurately and we will introduce some important terminology. Our main contributions are found in sections 3.4, 3.5 and 3.6 where we prove several theoretical results pertaining to removal of boundary convex or nonconvex quadrilaterals by introducing only internal Steiner points (additional nodes not belonging to the initial polygon). As initial preparations, we will discuss about chopping procedure for simply connected polygons in order to gain insight about the type of theoretical results that are developed in this document. Unfortunately, many results which are correct for simply connected polygons cannot be directly generalized to multiply connected polygons. Even the 2-ear theorem will fail for multiply connected polygons without additional assumption as we will show in the next discussion. As a consequence, we will introduce in section 3.5 the notion of double-edged polygons in order to carry the results over to multiply connected polygons. Those auxiliary polygons may contain double nodes and double edges which are traversed in opposite directions.

In most theoretical results that can be found in the literature, quadrangulations may contain initially *nonconvex* quadrilaterals. Since a quadrilateral decomposition having only *convex* members is very desired in practical cases, we will describe in section 3.8 a method of converting a nonconvex quadrangulation into a convex one.

For trimmed surfaces having curved boundaries, one takes coarse approximations of the curves which bound the trimmed surface. Afterward, we quadrilate the resulting multiply connected polygon. Finally, the quadrilateral cells are transformed into four-sided regions by replacing the straight boundary edges by the corresponding curved boundary portion of the original trimmed surface. Methods will be given in order to deal with boundary interference and local requadrangulations. At the end of this chapter, we present some numerical results of quadrilating simply and multiply connected polygons of various complexity. We will also give some benchmarks which are based on CAD data stored in IGES files [119].

3.2 Problem setting and general approach

Let us consider a closed surface $S \subset \mathbf{R}^3$ given as a collection of M trimmed parametric surfaces $S_1, \cdots, S_M$ defined on the 2D-domains $\mathcal{D}_1, \cdots, \mathcal{D}_M$ which are multiply connected regions in $\mathbf{R}^2$. The external and internal (when relevant) boundary curves of each domain $\mathcal{D}_i$ are supposed to be composite curves. We suppose further that the parametric functions defining S_i

$$\psi_i : \mathcal{D}_i \longrightarrow S_i \tag{3.1}$$

are diffeomorphisms.

Furthermore, we do not allow the existence of cusps at the boundaries of any $\mathcal{D}_i$. That is, if we suppose that the closed curve representing a boundary (exterior or interior) of $\mathcal{D}_i$ is given by the parametric curve κ, then we must have the following.

(B1) For all τ, we have $\dot{\kappa}(\tau) \neq \mathbf{0}$.

(B2) For all $\kappa(\tau)$ belonging to the boundary of $\mathcal{D}_i$,

$$\lim_{t \to \tau^-} \dot{\kappa}(t) \neq -\lambda \lim_{t \to \tau^+} \dot{\kappa}(t) \qquad \forall\, \lambda > 0. \tag{3.2}$$

Additionally, the curve κ is supposed to be bijective piecewise polynomial which can only have corners (discontinuity of tangents) at the segment separators (see former chapter about IGES).

The objective of this chapter is to (Fig. 3.1) tessellate S into m four-sided domains F_i

$$S = \bigcup_{i=1}^{m} F_i\,. \tag{3.3}$$

Furthermore, we will aim at having a splitting which is conforming. That is to say, every two different subregions F_i and F_j share a complete edge or they share a single corner or they are disjoint. Another objective is to keep the number m of surfaces F_i small. However, we do not intend to compute the globally optimal tessellation that minimizes m because the computational cost would be extremely high. We will try to have quadrilaterals which are well-shaped. Therefore, we will introduce techniques for measuring the quality of quadrilaterals in section 3.9. Since we want to construct diffeomorphisms, we show in section 3.14.4 the relationship between this chapter and the next one.

Our general approach to achieve (3.3) consists in splitting the 2D regions $\mathcal{D}_i$ into four-sided regions $Q_{k,i}$:

$$\mathcal{D}_i = \bigcup_k Q_{k,i}. \tag{3.4}$$

Thereto for each $\mathcal{D}_i$, we create a polygonal approximation $P^{(i)}$ which we tessellate into a list of quadrilaterals $q_{k,i}$. The four-sided regions $Q_{k,i}$ are obtained from $q_{k,i}$ by replacing the straight boundary edges of $q_{k,i}$ by the corresponding curve portion of $\mathcal{D}_i$.

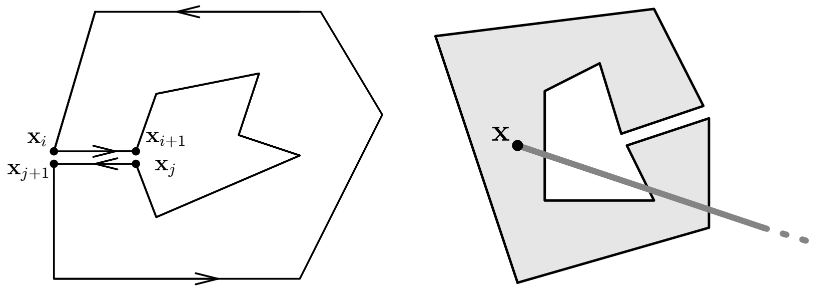
In order to obtain a decomposition which is conforming everywhere, we proceed as follows. We approximate the curved boundaries of $\{S_i\}$ by straight line segments separated by nodes $\{X_k\} \subset \mathbf{R}^3$. Then, we make the local splitting (3.4) in such a way that it is conforming inside $\mathcal{D}_i$ and that it uses only the preimages $\psi_i^{-1}(X_k)$ of the nodes $\{X_k\}$ as boundary vertices. That is, we do not use any additional boundary nodes.

The method that we propose for the local split (3.4) tessellates a polygon with n boundary vertices into $\mathcal{O}(n)$ convex quadrilaterals. As a consequence, if the number of its boundary vertices n_i for all polygons $P^{(i)}$ is smaller than n, then the total number of quadrilaterals is $\mathcal{O}(M \cdot n)$. For all examples that we considered, we found that the total number of quadrilaterals is quite small. However, we do not investigate how close our local approach comes in average to the globally optimal solution. In order to facilitate the presentation, let us use the following set of indices throughout this chapter

$$\Lambda := \{1, ..., M\}. \tag{3.5}$$

3.3 Polygonal regions

Before considering regions having curved boundaries, let us investigate polygonal regions. For each i, decomposing S_i amounts therefore to quadrilating the polygon $\mathcal{D}_i$. In the next discussions, we will skip the subscript i which identify the trimmed surface index. Thus, we consider a polygon P which has an even number of boundary vertices $\{\mathbf{x}_j\}$ and which might have internal boundaries. The



(a) (b)

Figure 3.2: (a) Double edge (b) Internal domain of a polygon.

assumption that the number of boundary vertices is even can be made without loss of generality because the internal angle at a vertex in a polygon is allowed to be π. In section 3.12.3, we describe a method of making the number of boundary vertices of each polygonal approximation even. Our objective is to find a list of convex quadrilaterals Q_k such that

$$P = \bigcup_k Q_k \tag{3.6}$$

with the following properties:

(P1) For any k, the vertices of Q_k are taken from the set of boundary vertices $\{\mathbf{x}_j\}$ or they are generated strictly in the interior domain of P (only interior Steiner points).

(P2) The intersection of two quadrilaterals is either empty or a complete edge.

Since the number of boundary nodes is even, the positive solvability of that problem is discussed in several references (see for example [99, 38, 88]) related to quadrangulation.

Let us now introduce a few definitions which are necessary for the understanding of the subsequent discussions.

Definition 1 For two points $\mathbf{x}$ and $\mathbf{y}$ in the plane we will denote by $[\mathbf{x}, \mathbf{y}]$ and $]\mathbf{x}, \mathbf{y}[$ the closed and open line segments defined by

$$[\mathbf{x}, \mathbf{y}] := \{\lambda \mathbf{x} + (1-\lambda)\mathbf{x}, \qquad \lambda \in [0,1] \subset \mathbf{R}\}, \tag{3.7}$$

$$]\mathbf{x}, \mathbf{y}[:= \{\lambda \mathbf{x} + (1-\lambda)\mathbf{x}, \qquad \lambda \in]0,1[\subset \mathbf{R}\}. \tag{3.8}$$

Definition 2 The line which passes through two given points a $\in \mathbf{R}^2$ and b $\in \mathbf{R}^2$ splits the plane into two half planes. We will define the positive (resp. negative) half plane $(\mathrm{ab})^+$ (resp. $(\mathrm{ab})^-$) as the half-plane on the right (resp. left) of the line. Consider a polygon P and a vertex a. The wedge of a is defined to be the region

$$\mathcal{W}(a) := (pa)^- \cap (sa)^+ , \tag{3.9}$$

where p and s are vertices which denote respectively the predecessor and the successor of a by following the polygon P counterclockwise. A vertex c of P is visible from another vertex d of P if the open line segment $]c, d[$ is located strictly inside the polygon P (we say also that c sees d). We define the kernel $\ker(P)$ of the polygon P to be the set of points from which all the vertices of P are visible.

Definition 3 Consider a polygonal domain P. A vertex $\mathbf{v}$ of P is called a reflex vertex if the internal angle at $\mathbf{v}$ is greater than π.

Definition 4 Consider a polygon P having vertices $\mathbf{x}_i$. A cut is a line segment $[\mathbf{x}_t, \mathbf{x}_s]$ having endpoints from the vertices of P such that the open segment $]\mathbf{x}_t, \mathbf{x}_s[$ is located completely inside P. An ear E of the polygon P is a triangle such that its apices are three consecutive vertices of P and that one edge of E is a cut. Chopping off a subpolygon P' from P means introducing a cut e that splits P into P' and the remaining polygon $\tilde{P}$, i.e.

$$P = P' \cup \tilde{P} \qquad P' \cap \tilde{P} = e. \tag{3.10}$$

Definition 5 For a simply connected polygon P, the kernel is the set of points from which all vertices of P are visible.

Definition 6 Consider a periodic list of points $\{\mathbf{x}_s\}$ in the plane. We say that we have a double edge if there exist $i \neq j$ such that:

$$\left\{ \begin{array}{lcl} \text{coordinates}(\mathbf{x}_i) & = & \text{coordinates}(\mathbf{x}_{j+1}) \\ \text{coordinates}(\mathbf{x}_{i+1}) & = & \text{coordinates}(\mathbf{x}_j). \end{array} \right. \tag{3.11}$$

As a consequence, the edge is traversed in two opposite directions as illustrated in Fig. 3.2(a). Note that in the figures throughout this chapter, nodes which are very close to one another such as those of Fig. 3.2(a) are supposed to have the same coordinates. The nodes are only separated for the graphical illustration.

Definition 7 For any polygon P, we will denote by $\mathcal{I}(P)$ its internal domain which is defined by

$$\begin{array}{rcl} \mathcal{I}(P) & := & \{\mathbf{x} \in \mathbf{R}^2 \text{ not on any edge of } P \text{ : the half-line } [\mathbf{x}, \vec{u}) \\ & & \text{intersects } P \text{ an odd number of times for any } \vec{u}\}. \end{array} \tag{3.12}$$

As an illustration, we show in Fig. 3.2(b) the internal domain by the shaded area.

Since our theoretical development is partially based on the 2-ear theorem [84], let us first recall its statement:

Theorem 1 (Meister, 1975) Every simple (simply connected, without intersecting edges) polygon having at least four vertices has two nonoverlapping ears.

Both the statement and the proof of the 2-ear theorem are very simple. It is yet a very powerful tool for both theoretical and programming purposes. This theorem is very useful in practice when we want to decompose a given polygon into a set of triangles. A very fast algorithm can be easily devised to chop off recursively a triangle from the polygon in order to form a triangulation. In this chapter, this theorem will mainly serve as a tool to decrement the number of vertices in proofs using induction.

3.4 Removing a quadrilateral from a simple polygon

With the above definitions in mind, we are now ready to state our first fundamental result about quadrilateral removal. The following statement can be used to decompose a simple polygon into quadrilaterals which are not necessarily convex.

Theorem 2 Consider a simple polygon P having at least four vertices, one of the following two statements must hold true:

(Op1) One can remove a quadrilateral which is not necessarily convex by inserting a single cut as in Fig. 3.3(a).

(Op2) There exists a point ω located strictly within P such that one can remove a *convex* quadrilateral by inserting two line segments emanating from ω to two vertices of P (Fig. 3.3(b)).

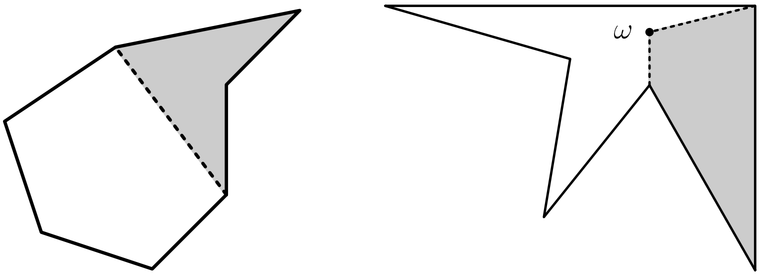
Proof

We will proceed by induction with respect to the number n of the vertices $\mathbf{x}_i$ of the polygon. For $n = 4$, the theorem is evident. As hypothesis of induction, we suppose that the claim holds for every polygon having n vertices.

Consider now a polygon P having $n+1$ vertices and let us show that we may chop a quadrilateral Q from P. First, let us apply the 2-ear theorem in order to chop off a triangle $E = [\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}]$ from P and let us denote by $\tilde{P}$ the remaining polygon which must have n vertices. That is,

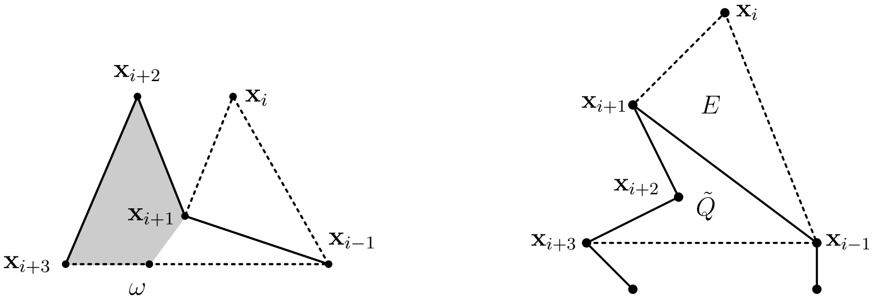
$$\tilde{P} := [\mathbf{x}_0, \cdots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \cdots, \mathbf{x}_n]. \tag{3.13}$$

After applying the hypothesis of induction to the new polygon $\tilde{P}$, we obtain a quadrilateral $\tilde{Q}$. Let us now consider three cases according to the incidence of $\tilde{Q}$ upon the ear E and to the existence of Steiner point in $\tilde{Q}$.



(a) (b)

Figure 3.3: (a) Chop off a quadrilateral with one cut (b) Remove a quadrilateral with two cuts and one internal node.



(a) (b)

Figure 3.4: (a) Generate ω (b) $\mathbf{x}_{i+2}$ is reflex in $\tilde{Q}$.

Case 1: If the quadrilateral $\tilde{Q}$ is not incident upon the edge $[\mathbf{x}_{i-1}, \mathbf{x}_{i+1}]$ inside the polygon $\tilde{P}$, then we simply need to define $Q := \tilde{Q}$.

Case 2: Suppose now that $\tilde{Q}$ has $[\mathbf{x}_{i-1}, \mathbf{x}_{i+1}]$ as edge and $\tilde{Q}$ has no internal Steiner node. We will investigate a few subcases.

Case 2.a: Assume that $\tilde{Q} = [\mathbf{x}_{i-2}, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}]$ as in Fig. 3.6(a).

Inside the quadrilateral $\tilde{Q}$, $\mathbf{x}_{i-1}$ is visible from $\mathbf{x}_{i+2}$ or $\mathbf{x}_{i+1}$ is visible from $\mathbf{x}_{i-2}$ (indeed, you can apply a simple 2-ear theorem inside $\tilde{Q}$). In the former situation, define $Q := [\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}, \mathbf{x}_{i-1}]$. In the latter situation, define $Q := [\mathbf{x}_{i+1}, \mathbf{x}_{i-2}, \mathbf{x}_{i-1}, \mathbf{x}_i]$. That is, we have just applied operation (Op1) to P in case 2.a.

Case 2.b: Assume that $\tilde{Q} = [\mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}, \mathbf{x}_{i+3}]$.

If the quadrilateral $\tilde{Q}$ is convex, then we simply need to apply (Op1) by defining

$$Q := [\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}, \mathbf{x}_{i-1}]. \tag{3.14}$$

In the situation that $\tilde{Q}$ is nonconvex, we investigate four events according to the position of its reflex vertex.

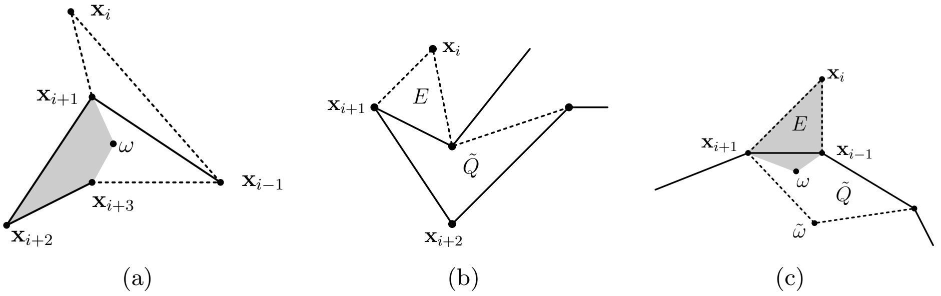


Figure 3.5: (a)$\mathbf{x}_{i+3}$ is reflex in $\tilde{Q}$ (b)$\mathbf{x}_{i-1}$ is reflex in $\tilde{Q}$ (c)Introducing a Steiner point ω in $\tilde{Q} \cap \mathrm{Wedge}(\mathbf{x}_i)$.

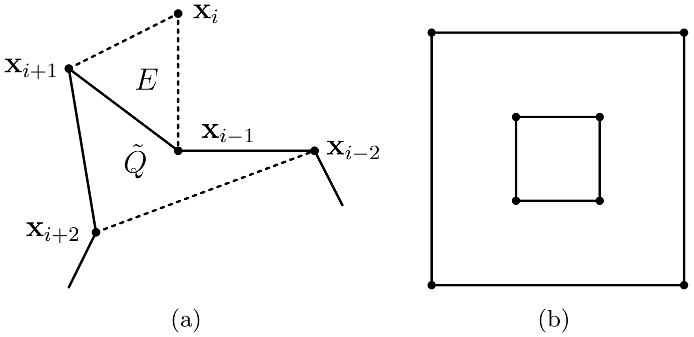


Figure 3.6: (a) The ear $[\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}]$ and the removed quadrilateral are adjacent (b) A doubly connected polygon from which one may not chop any triangle off.

- If the vertex $\mathbf{x}_{i+2}$ (resp. $\mathbf{x}_{i-1}$) is its reflex vertex as depicted in Fig. 3.4(b) (resp. Fig. 3.5(b)), we define $Q := [\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_{i+2}]$ (resp. $Q := [\mathbf{x}_{i+2}, \mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}]$).
- In the case that $\mathbf{x}_{i+1}$ is the reflex vertex of $\tilde{Q}$, take any node ω on the open segment $]\mathbf{x}_{i+3}, \mathbf{x}_{i-1}[$ in the wedge of $\mathbf{x}_{i+2}$ as illustrated in Fig. 3.4(a) and apply (Op2) by defining
$$Q := [\mathbf{x}_{i+2}, \mathbf{x}_{i+3}, \omega, \mathbf{x}_{i+1}]. \tag{3.15}$$
  Note that the quadrilateral Q in relation (3.15) must be convex.
- If the vertex $\mathbf{x}_{i+3}$ is the reflex vertex, we proceed as in (3.15) but the internal Steiner point ω is chosen strictly within the triangle (Fig. 3.5(a)) $[\mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \mathbf{x}_{i+3}]$ such that it is in the wedge of $\mathbf{x}_{i+2}$.

Case 2.c: In the case that $\tilde{Q} = [\mathbf{x}_{i-3}, \mathbf{x}_{i-2}, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}]$, proceed as in case 2.b but in opposite direction (clockwise).

Case 3: In the event that the segment $[\mathbf{x}_{i-1}, \mathbf{x}_{i+1}]$ is an edge of $\tilde{Q}$ which has a Steiner point $\tilde{\omega}$ and which is convex (Fig. 3.5(c)), take any point ω that is strictly inside $\tilde{Q}$ and that is in the wedge of $\mathbf{x}_i$. On account of the convexity of $\tilde{Q}$, both $\mathbf{x}_{i-1}$ and $\mathbf{x}_{i+1}$ are visible from the node ω. Therefore we may define

$$Q := [\mathbf{x}_i, \mathbf{x}_{i+1}, \omega, \mathbf{x}_{i-1}] \tag{3.16}$$

as a quadrilateral which can be removed from P.

□

Remark 1 One can observe that the number of vertices of the remaining polygon is $(n-2)$ after applying operation (Op1) to a polygon having n vertices. On the opposite, applying operation (Op2) as illustrated in Fig. 3.3(b) does not reduce the number of vertices. At first sight, it may therefore seem that recursively applying the above theorem does not split a polygon into a set of quadrilaterals (convex or not). In other words, one may wonder if the following algorithm of quadrilateral decomposition is guaranteed to terminate when applied to a polygon having an even number of vertices.

```
step 0 : Initialize $P_0 := P$ and $k = 0$.
step 1 : If operation (Op1) can be applied to the polygon $P_k$, chop
         off a quadrilateral. Else apply operation (Op2).
         In both cases, let $Q$ be the resulting quadrilateral.
step 2 : Define $P_{k+1} := P_k \setminus Q$.
step 3 : If the number of vertices of $P_{k+1} = 4$, terminate.
         Otherwise, set $k := k + 1$ and go to step 1.
```

If we inspect the above proof more closely, we notice that whenever we apply operation (Op2), then a neighboring quadrilateral can be chopped off in the next iteration by applying operation (Op1). As illustrated in Fig. 3.4(a), after removing the shaded quadrilateral, one may chop the quadrilateral $[\mathbf{x}_{i+1}, \mathbf{x}_{i+2}, \mathbf{x}_{i+3}, \omega]$ off by applying operation (Op1). That is to say, after at most two iterations of the above algorithm, the number of vertices must decrement twice. As a consequence, the number of resulting quadrilaterals is $\mathcal{O}(n)$ for a polygon having n vertices.

3.5 Toward multiply connected polygons

In this section we will present statements that will be used later for the case of multiply connected polygons. Before going on, we note that the above proof cannot be directly generalized for multiply connected polygons because of the following observation.

Remark 2 (Counter-example for multiply connected polygons) The 2-ear theorem does not hold true for general multiply connected polygons. As a counter

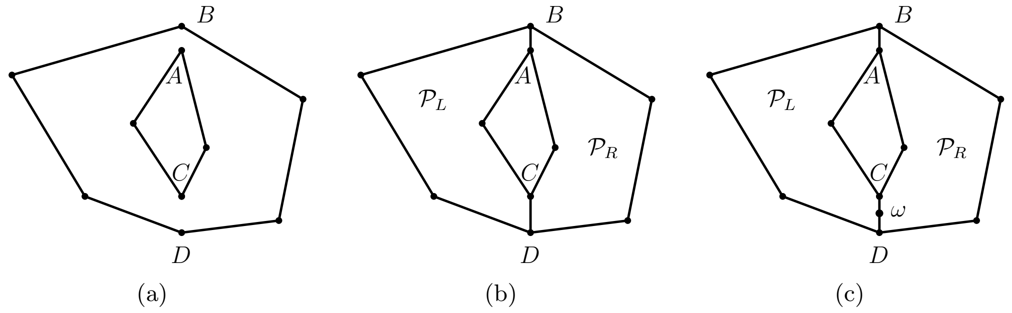


Figure 3.7: (a)Initial polygon (b)Make simply connected (c)Make even

example, we simply need to consider a doubly connected polygon whose exterior and interior boundaries are squares as illustrated in Fig. 3.6(b). As one can clearly observe, it is impossible to chop off a triangle by inserting a single cut.

We have proved a method of quadrilating any simply connected polygon in the previous section. An obvious idea for quadrilating a multiply connected polygon P is to split it first into simply connected ones by inserting two cuts for each internal curve. Then, the previous quadrangulation approach can be applied to each simply connected polygon. This method has severe drawbacks.

First, it is not efficient because two cuts are needed for each interior curve while our approach requires only one cut per interior curve.

Second, it can happen that the simple polygons have an odd number of boundary vertices. As an illustration, if we insert two cuts $[A, B]$ and $[C, D]$ in the polygon of Fig. 3.7(b), then the polygon $\mathcal{P}$ is split into two odd polygons P_L and P_R. To avoid this situation, one has to either include an additional condition about the allowed number of vertices in a subpolygon which further complicates the cut search or one has to introduce an additional node (Fig. 3.7(c)). The polygons that we use in our approach (we call them double-edged polygons) have the property that the number of vertices always remains even as long as the initial polygon has an even number of vertices. Therefore, we can apply directly the quadrangulation technique after converting a multiply connected polygon into a double-edged one.

A further advantage of double-edged polygons is in the implementation aspect. One does not need to have two computer programs to quadrilate a simply connected polygon and a double-edged polygon. That is due to the fact that the input of those kinds of polygon is given as a single sequence of 2D-vertices in practice.

Definition 8 A polygon P with vertices $\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{N-1}$ is called double-edged if it fulfills the following criteria:

(C1) The boundary of P is the set of line segments $[\mathbf{x}_{N-1}, \mathbf{x}_0]$, $[\mathbf{x}_i, \mathbf{x}_{i+1}]$ for $i = 0, \cdots, N-2$.

(C2) P may contain double edges but no consecutive double edges, i.e. the following situation does not occur:

$$\begin{cases} \text{coordinates}(\mathbf{x}_i) & = & \text{coordinates}(\mathbf{x}_{j+2}) \\ \text{coordinates}(\mathbf{x}_{i+1}) & = & \text{coordinates}(\mathbf{x}_{j+1}) \\ \text{coordinates}(\mathbf{x}_{i+2}) & = & \text{coordinates}(\mathbf{x}_j). \end{cases} \tag{3.17}$$

(C3) Any three pairwise different vertices of P are affinely independent.

(C4) All internal angles are different from zero (Fig. 3.9(b)).

(C5) The internal domain $\mathcal{I}(P)$ is connected.

Remark 3 In Fig. 3.9(a), we have a polygon which violates the criterion (C5) because the internal domain has two connected components. Since criterion (C3) is too restrictive to treat interesting practical situations, we are going to relax it later on.

Theorem 3 From every double-edged polygon P having more than four vertices, one may chop off two triangles.

Proof

We proceed by induction with respect to the number m of double edges. First of all, let us consider the case where $m = 1$ and suppose that the edges $[\mathbf{x}_\sigma, \mathbf{x}_{\sigma+1}]$ and $[\mathbf{x}_\rho, \mathbf{x}_{\rho+1}]$ coincide as displayed in Fig. 3.8. Define

$$\vec{u} := \overrightarrow{\mathbf{x}_\sigma \mathbf{x}}_{\sigma+1} / \|\overrightarrow{\mathbf{x}_\sigma \mathbf{x}}_{\sigma+1}\| \tag{3.18}$$

and denote by $\vec{n}$ the unit vector normal to the segment $[\mathbf{x}_\sigma, \mathbf{x}_{\sigma+1}]$ such that $\det(\vec{u}, \vec{n}) = 1$. Denote by $[\kappa(\mathbf{x}), \lambda(\mathbf{x})]$ the coordinates of a point $\mathbf{x}$ in the reference $[\mathbf{0}, \vec{u}, \vec{n}]$. Let us introduce $\mu_0 := \alpha \cdot \lambda_0$ where

$$\lambda_0 := \min\{\lambda(\mathbf{x}_r) > \lambda(\mathbf{x}_\sigma) \ : \ \mathbf{x}_r \text{ vertex of } P\}, \tag{3.19}$$

and α is a positive number whose value will be established later. Now we may use an auxiliary polygon $\tilde{P}$ having the following vertices

$$\begin{cases} \tilde{\mathbf{x}}_\sigma & := & \mathbf{x}_\sigma + \mu_0 \vec{n} \\ \tilde{\mathbf{x}}_{\sigma+1} & := & \mathbf{x}_{\sigma+1} + \mu_0 \vec{n} \\ \tilde{\mathbf{x}}_r & := & \mathbf{x}_r \quad \text{if} \quad r \notin \{\sigma, \sigma+1\}. \end{cases} \tag{3.20}$$

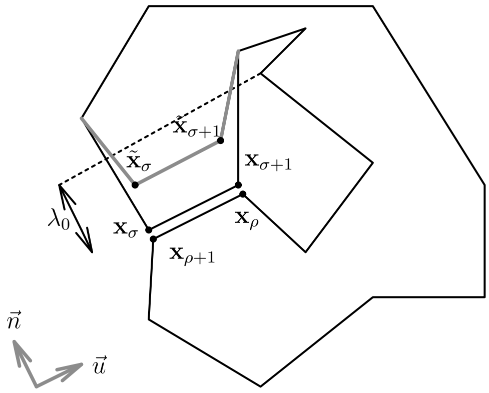
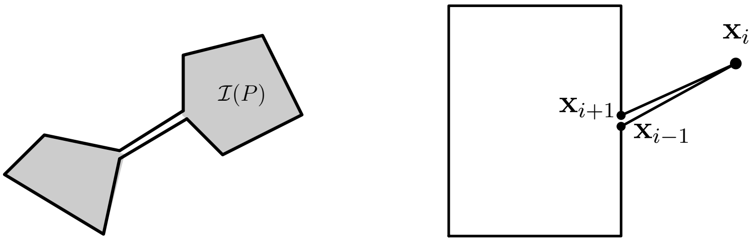


Figure 3.8: Original and auxiliary polygons.



(a)

(b)

Figure 3.9: (a) The interior domain is disconnected (b) Zero angle at $\mathbf{x}_i$ is not allowed.

Since the auxiliary polygon $\tilde{P}$ does not contain any double edge, we may apply the usual 2-ear theorem in order to obtain two triangles $\tilde{E}_1$ and $\tilde{E}_2$ from $\tilde{P}$. For every triangle $\tilde{E}_i$ $(i = 1, 2)$, we distinguish three cases according to the incidence of $\tilde{E}_i$ upon the nodes $\tilde{\mathbf{x}}_\sigma$ and $\tilde{\mathbf{x}}_{\sigma+1}$.

Case 1: If $\tilde{E}_i$ does not have any of the nodes $\tilde{\mathbf{x}}_\sigma$ and $\tilde{\mathbf{x}}_{\sigma+1}$ as apex, we simply define $E_i := \tilde{E}_i$.

Case 2: Let us now assume that the segment $[\tilde{\mathbf{x}}_\sigma, \tilde{\mathbf{x}}_{\sigma+1}]$ is an edge of $\tilde{E}_i$. Without loss of generality we may suppose that the third node of $\tilde{E}_i$ is $\mathbf{x}_{\sigma+2}$ i.e. $\tilde{E}_i = [\tilde{\mathbf{x}}_\sigma, \tilde{\mathbf{x}}_{\sigma+1}, \mathbf{x}_{\sigma+2}]$ (the case where $\tilde{E}_i = [\mathbf{x}_\sigma, \mathbf{x}_{\sigma+1}, \mathbf{x}_{\sigma-1}]$ can be treated exactly in the same manner). Let us define $E_i := [\mathbf{x}_\sigma, \mathbf{x}_{\sigma+1}, \mathbf{x}_{\sigma+2}]$.

Since no three vertices are located on one straight line as specified by criterion (C3), we may introduce $\delta := \min\{\delta_1, \delta_2\}$ where

$$\begin{aligned} \delta_1 &:= \min\{\operatorname{dist}([\mathbf{x}_\sigma, \mathbf{x}_{\sigma+2}], \mathbf{x}_r), \; r \neq \sigma \qquad r \neq \sigma+2 \qquad r \neq \rho+1\} > 0 \\ \delta_2 &:= \min\{\operatorname{dist}([\mathbf{x}_{\sigma+1}, \mathbf{x}_{\sigma-1}], \mathbf{x}_r), \; r \neq \sigma-1 \qquad r \neq \sigma+1 \qquad r \neq \rho\} > 0. \end{aligned} \tag{3.21}$$

Since $\operatorname{dist}([\mathbf{x}_\sigma, \mathbf{x}_{\sigma+2}], \tilde{\mathbf{x}}_\sigma)$ tends to zero as $\alpha \to 0$, there must exist α_0 sufficiently small such that

$$\text{if} \quad \alpha < \alpha_0 \quad \text{then} \quad \operatorname{dist}([\mathbf{x}_\sigma, \mathbf{x}_{\sigma+2}], \tilde{\mathbf{x}}_\sigma) < \delta. \tag{3.22}$$

Let us introduce the regions

$$S := \left\{\mathbf{x} \in \mathbf{R}^2 : \lambda(\mathbf{x}_\sigma) < \lambda(\mathbf{x}) < \mu_0\right\}, \tag{3.23}$$

$$R := \left\{\mathbf{x} \in \mathbf{R}^2 : \lambda(\mathbf{x}) > \mu_0\right\}. \tag{3.24}$$

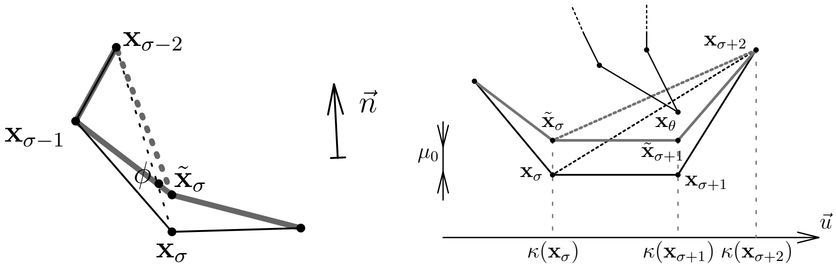
According to the above definition of $\vec{u}$ and κ, we must have $\kappa(\mathbf{x}_\sigma) < \kappa(\mathbf{x}_{\sigma+1})$. We will need three subcases according to the position of $\kappa(\mathbf{x}_{\sigma+2})$ with respect to $\kappa(\mathbf{x}_\sigma)$ and $\kappa(\mathbf{x}_{\sigma+1})$.

Case 2.a: Assume that $\kappa(\mathbf{x}_\sigma) < \kappa(\mathbf{x}_{\sigma+2}) < \kappa(\mathbf{x}_{\sigma+1})$.

In this case as illustrated in Fig. 3.11(b), the line segment $[\tilde{\mathbf{x}}_\sigma, \tilde{\mathbf{x}}_{\sigma+1}]$ must intersect the two edges $[\mathbf{x}_\sigma, \mathbf{x}_{\sigma+2}]$ and $[\mathbf{x}_{\sigma+1}, \mathbf{x}_{\sigma+2}]$ of the triangle E_i. Because of the definition of μ_0, there could not exist any vertex in the stripe S. On the other hand, we have $(E_i \cap R) \subset \tilde{E}_i$. As a consequence, the triangle E_i cannot contain any vertex of P (other than the apices of E_i).

Case 2.b: Assume that $\kappa(\mathbf{x}_{\sigma+1}) \leq \kappa(\mathbf{x}_{\sigma+2})$.

In $E_i \cap S$, there could not exist any node. Additionally, no nodes lies strictly inside $E_i \cap R$. Indeed, if there was a node $\mathbf{x}_\theta$ in $E_i \cap R$ as illustrated in Fig. 3.10(b), then there would exist an edge traversing the cut $[\tilde{\mathbf{x}}_\sigma, \tilde{\mathbf{x}}_{\sigma+2}]$ which is in contradiction with the fact that $\tilde{E}_i$ can be chopped off from $\tilde{P}$.



(a) (b)

Figure 3.10: (a)$[\mathbf{x}_\sigma, \mathbf{x}_{\sigma-2}, \mathbf{x}_{\sigma-1}]$ can be chopped off. (b) $\kappa(\mathbf{x}_{\sigma+1}) \leq \kappa(\mathbf{x}_{\sigma+2})$.

Case 2.c: Assume that $\kappa(\mathbf{x}_{\sigma+2}) \leq \kappa(\mathbf{x}_\sigma)$.

Let τ be the intersection of the line (Fig. 3.11(a)) passing through $[\tilde{\mathbf{x}}_\sigma, \tilde{\mathbf{x}}_{\sigma+1}]$. As above, there is no node of P which is located strictly inside $E_i \cap S$. If we choose the value of α as in (3.22), then there could not exist any vertex of P within the region $[\tau, \tilde{\mathbf{x}}_\sigma, \mathbf{x}_{\sigma+2}]$. Therefore, the triangle E_i can be chopped from the polygon P off.

Case 3: Let us now assume that $\tilde{E}_i$ is not incident upon the edge $[\tilde{\mathbf{x}}_\sigma, \tilde{\mathbf{x}}_{\sigma+1}]$ but it has $\tilde{\mathbf{x}}_\sigma$ or $\tilde{\mathbf{x}}_{\sigma+1}$ as apex. Let us only treat the situation where $\tilde{E}_i := [\tilde{\mathbf{x}}_\sigma, \mathbf{x}_{\sigma-2}, \mathbf{x}_{\sigma-1}]$ because the other case where $\tilde{E}_i = [\tilde{\mathbf{x}}_{\sigma+1}, \mathbf{x}_{\sigma+2}, \mathbf{x}_{\sigma+3}]$ can be treated in a similar manner. Let us introduce

$$\begin{array}{lcl} \delta_3 & := & \min\{\text{dist}\,([\mathbf{x}_{\sigma-2}, \mathbf{x}_\sigma], \mathbf{x}_r)\,,\ r \neq \sigma \qquad r \neq \sigma-2 \qquad r \neq \rho+1\} > 0 \\ \delta_4 & := & \min\{\text{dist}\,([\mathbf{x}_{\sigma+1}, \mathbf{x}_{\sigma+3}], \mathbf{x}_r)\,,\ r \neq \sigma+1 \qquad r \neq \sigma+3 \qquad r \neq \rho\} > 0. \end{array} \tag{3.25}$$

By introducing $\delta' := \min\{\delta_3, \delta_4\}$, we proceed the way we have done in (3.22) in order to obtain an $\alpha_1 > 0$ sufficiently small such that

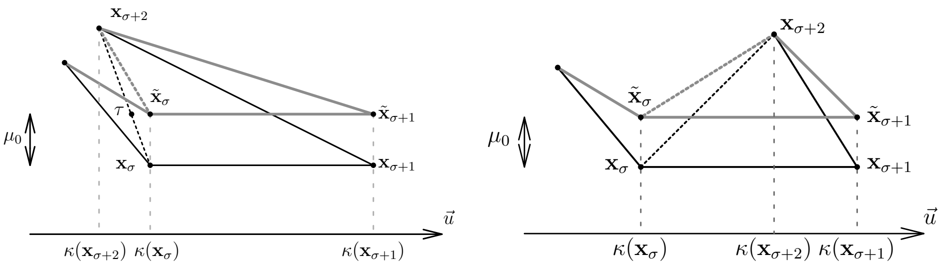
$$\text{if} \quad \alpha < \alpha_1 \quad \text{then} \quad \text{dist}\,([\mathbf{x}_\sigma, \mathbf{x}_{\sigma-2}], \tilde{\mathbf{x}}_\sigma) < \delta'. \tag{3.26}$$

Thus, if α is chosen as in (3.26), the open triangle $[\tilde{\mathbf{x}}_\sigma, \mathbf{x}_{\sigma-2}, \phi]$ does not contain any node where ϕ is the intersection of $\tilde{P}$ and $[\mathbf{x}_\sigma, \mathbf{x}_{\sigma-2}]$ as displayed in Fig. 3.10(a). As a consequence, the triangle $[\mathbf{x}_\sigma, \mathbf{x}_{\sigma-2}, \mathbf{x}_{\sigma-1}]$ can be chopped off from the polygon P.

All in all, if α is chosen so that $\alpha < \min\{\alpha_0, \alpha_1\}$ where α_0 and α_1 are obtained from relations (3.22) and (3.26) then the claim is true for $m = 1$. As hypothesis of induction, we suppose that the claim holds for any m. For a polygon having $m + 1$ double edges, we apply the same procedure as above in order to have an auxiliary polygon $\tilde{P}$ having m double edges and apply the hypothesis of induction to $\tilde{P}$.

$\square$

From the above proof, we see that the theorem holds even for polygons in which the condition (C3) is relaxed. More precisely, we have the following result.


$\mathbf{x}_{\sigma+2}$
$\tilde{\mathbf{x}}_\sigma$
τ
$\tilde{\mathbf{x}}_{\sigma+1}$
μ_0
$\mathbf{x}_\sigma$
$\mathbf{x}_{\sigma+1}$
$\vec{u}$
$\kappa(\mathbf{x}_{\sigma+2})$ $\kappa(\mathbf{x}_\sigma)$
$\kappa(\mathbf{x}_{\sigma+1})$
$\mathbf{x}_{\sigma+2}$
$\tilde{\mathbf{x}}_\sigma$
$\tilde{\mathbf{x}}_{\sigma+1}$
μ_0
$\mathbf{x}_\sigma$
$\mathbf{x}_{\sigma+1}$
$\vec{u}$
$\kappa(\mathbf{x}_\sigma)$
$\kappa(\mathbf{x}_{\sigma+2})$ $\kappa(\mathbf{x}_{\sigma+1})$

(a) (b)

Figure 3.11: (a) $\kappa(\mathbf{x}_{\sigma+2}) \leq \kappa(\mathbf{x}_\sigma)$ (b) $\kappa(\mathbf{x}_\sigma) < \kappa(\mathbf{x}_{\sigma+2}) < \kappa(\mathbf{x}_{\sigma+1})$.

Corollary 1 Suppose that we have a polygon P fulfilling conditions (C1), (C2), (C4) and (C5). Let us adopt the notations of the proof of Theorem 2. Furthermore, we suppose that we have the following condition:

($\widetilde{C3}$) The eight open segments $]\mathbf{x}_s, \mathbf{x}_{s+2}[$, $s = \sigma$, $\sigma - 1$, ρ, $\rho - 1$, $\sigma - 2$, $\sigma + 1$, $\rho - 2$, $\rho + 1$ do not contain any node.

Then one can remove two ears from the polygon P.

Proof

The proof can basically remain unchanged because the parameters δ_1, δ_2, δ_3, δ_4 from relations (3.21) and (3.25) are still strictly positive. Please note that we do not penalize the existence of nodes inside the triangles which are next to the double edge such as $[\mathbf{x}_\sigma, \mathbf{x}_{\sigma+1}, \mathbf{x}_{\sigma+2}]$ in theorem 1.

$\square$

We can even remove the condition ($\widetilde{C3}$) completely as in the following statement.

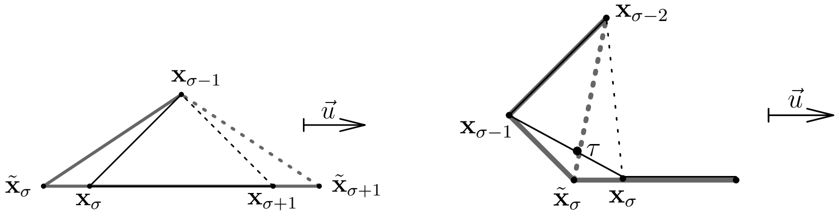
Corollary 2 Suppose that a polygon P meets the four conditions (C1), (C2), (C4), (C5). Then one can remove two ears from the polygon P.

Proof

Adopt the definition of the vectors $\vec{n}$ and $\vec{u}$ as in the previous proof. Now we would like to shift some vertices in the direction of $\vec{u}$. Let us define:

$$\delta_s := \min\{\mathrm{dist}([\mathbf{x}_s, \mathbf{x}_{s+2}], \mathbf{x}_r), \quad \mathbf{x}_r \notin [\mathbf{x}_s, \mathbf{x}_{s+2}]\} > 0. \tag{3.27}$$

$$\gamma := 0.5 \cdot \min\{\delta_s : s = \sigma, \sigma-1, \rho, \rho-1, \sigma-2, \sigma+1, \rho-2, \rho+1\}.$$



(a) (b)

Figure 3.12: Chopping an ear off next to a shifted node.

Let us introduce also an auxiliary polygon $\tilde{P}$ which has the following vertices in place of $\mathbf{x}_\sigma$, $\mathbf{x}_{\rho+1}$ $\mathbf{x}_{\sigma+1}$, $\mathbf{x}_\rho$ (the other vertices are the same as those of P):

$$\begin{array}{lcl} \tilde{\mathbf{x}}_\sigma & := & \mathbf{x}_\sigma - \gamma.\vec{u} \\ \tilde{\mathbf{x}}_{\rho+1} & := & \mathbf{x}_{\rho+1} - \gamma.\vec{u} \\ \tilde{\mathbf{x}}_{\sigma+1} & := & \mathbf{x}_{\sigma+1} + \gamma.\vec{u} \\ \tilde{\mathbf{x}}_\rho & := & \mathbf{x}_\rho + \gamma.\vec{u}. \end{array} \tag{3.28}$$

As a consequence, there could not exist any polygon vertex on the eight open segment $]\tilde{\mathbf{x}}_s, \tilde{\mathbf{x}}_{s+2}[$, $s = \sigma, \sigma-1, \rho, \rho-1, \sigma-2, \sigma+1, \rho-2, \rho+1$ of the auxiliary polygon $\tilde{P}$.

Apply theorem 1 to the auxiliary polygon $\tilde{P}$ so that we can remove two triangles $\tilde{E}_1$ and $\tilde{E}_2$. In the easy case that $\tilde{E}_i$ $(i = 1, 2)$ is not incident upon any double node, we simply take $E_i := \tilde{E}_i$. For the situation that $\tilde{E}_i$ is incident upon the nodes $\tilde{\mathbf{x}}_\sigma$ or $\tilde{\mathbf{x}}_{\sigma+1}$, we will consider the four next cases.

Case 1: $\tilde{E}_i = [\mathbf{x}_{\sigma-1}, \tilde{\mathbf{x}}_\sigma, \tilde{\mathbf{x}}_{\sigma+1}]$

In this case as illustrated in Fig. 3.12(a), the triangle $E_i := [\mathbf{x}_{\sigma-1}, \mathbf{x}_\sigma, \mathbf{x}_{\sigma+1}]$ can be chopped from the polygon P off because E_i must be included in $\tilde{E}_i$.

Case 2: $\tilde{E}_i = [\mathbf{x}_{\sigma-2}, \mathbf{x}_{\sigma-1}, \tilde{\mathbf{x}}_\sigma]$.

Let us define by τ the intersection of the segment $[\tilde{\mathbf{x}}_\sigma, \mathbf{x}_{\sigma-2}]$ and $[\mathbf{x}_{\sigma-1}, \mathbf{x}_\sigma]$ as illustrated by Fig. 3.12(b). According to the choice of δ from relation (3.27), there cannot exist any node within the triangle $[\mathbf{x}_{\sigma-2}, \tau, \mathbf{x}_\sigma]$.

Case 3: If $\tilde{E}_i = [\tilde{\mathbf{x}}_\sigma, \tilde{\mathbf{x}}_{\sigma+1}, \mathbf{x}_{\sigma+2}]$, proceed as in case 1.

Case 4: If $\tilde{E}_i = [\tilde{\mathbf{x}}_{\sigma+1}, \mathbf{x}_{\sigma+2}, \mathbf{x}_{\sigma+3}]$, proceed as in case 2.

The cases where $\tilde{E}_i$ is incident upon $\tilde{\mathbf{x}}_\rho$ or $\tilde{\mathbf{x}}_{\rho+1}$ are treated exactly in the same fashion.

□

Remark 4 From a simply connected polygon P which has more than four vertices and which might have double nodes but without double edges such that the internal domain is connected, one can chop off two triangles E_1 and E_2.

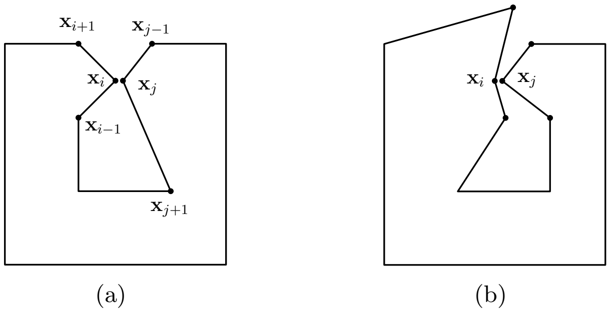


Figure 3.13: Double nodes: (a) $\mathbf{x}_i$ and $\mathbf{x}_j$ are convex (b) $\mathbf{x}_i$ is reflex.

Proof [sketch]

We will only sketch the proof in which we suppose that we have only one double node $\mathbf{x}_i$ and $\mathbf{x}_j$ as illustrated in Fig. 3.13. The general case can be demonstrated by induction. Let us denote by α_s the internal angle made by $\mathbf{x}_{s-1}$, $\mathbf{x}_s$, $\mathbf{x}_{s+1}$. At least one of the internal angles α_i and α_j must be smaller than π because it is impossible that two reflex vertices coincide (Fig. 3.13). Thus, let us suppose $\alpha_i < \pi$.

Let $\vec{w}$ be the normalized form of

$$\overrightarrow{\mathbf{x}_i\mathbf{x}}_{i-1} + \overrightarrow{\mathbf{x}_i\mathbf{x}}_{i+1}. \tag{3.29}$$

Choose a positive value μ so that the shifted vertex

$$\tilde{\mathbf{x}}_i := \mathbf{x}_i + \mu.\vec{w} \tag{3.30}$$

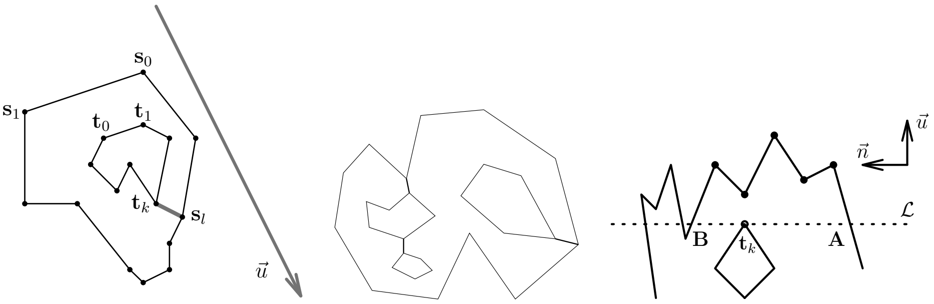
does not interfere with any other vertex. Apply the usual 2-ear theorem to the new polygon with $\tilde{\mathbf{x}}_i$ in place of $\mathbf{x}_i$ in order to chop two triangles $\tilde{E}_i$. Then, consider the following three cases. First, $\tilde{E}_i$ is not incident upon $\tilde{\mathbf{x}}_i$. Second, the segment $[\mathbf{x}_{i-1}, \mathbf{x}_{i+1}]$ is an edge of $\tilde{E}_i$. Last, the segment $[\mathbf{x}_i, \mathbf{x}_{i+2}]$ or $[\mathbf{x}_i, \mathbf{x}_{i-2}]$ is an edge of $\tilde{E}_i$.

□

Remark 5 By using a similar argument as in the above proof, one can easily see that the claim in corollary 2 holds true even for polygons having double nodes which are not incident to double edges.

3.6 Quadrilating multiply connected polygons

Lemma 1 Consider a multiply connected polygon P having $\mathcal{E}$ as exterior boundary and $\mathcal{J}_j$ as interior ones $j = 1, \cdots, m$. Denote by the $\mathbf{s}_i$ the vertices of $\mathcal{E}$ and by $\mathbf{t}_{j,i}$ those of $\mathcal{J}_j$. We suppose that $\mathcal{E}$ is a double-edged polygon.



(a) (b) (c)

Figure 3.14: (a) $[\mathbf{s}_l, \mathbf{t}_k]$ is a cut joining the interior and the exterior boundaries (b) cuts inside a multiply connected polygon. (c) the members of $\mathcal{M}$ are the bold points above $\mathcal{L}$

There must exist an interior curve $\mathcal{J}_{j0}$ and two vertices $\mathbf{s}_l$ and $\mathbf{t}_{j0,k}$ which are mutually visible. Thus, the segment $[\mathbf{s}_l, \mathbf{t}_{j0,k}]$ is a cut inside the polygon P.

Proof

Consider any direction $\vec{u}$ such that $\|\vec{u}\| = 1$ and consider the axis $[\mathbf{0}, \vec{u})$ centered at the origin and directed by $\vec{u}$. Denote by $\lambda(\mathbf{x})$ the coordinate of the projection of a point $\mathbf{x}$ on the axis $[\mathbf{0}, \vec{u})$. Let n_j be the number of vertices of $\mathcal{J}_j$.

Denote by $\mathcal{J}_{j0}$ the curve which contains the node $\mathbf{t}_{j0,k}$ having the largest λ value:

$$\mathbf{t}_{j0,k} := \text{argmax}\{\lambda(\mathbf{t}_{j,i}) \,:\, \forall j = 1, \cdots, m \quad \forall i = 1, \cdots, n_j\}. \tag{3.31}$$

Define $\vec{n}$ the unit normal such that $det(\vec{u}, \vec{n}) = 1$. The line $\mathcal{L}$ passing through $\mathbf{t}_{j0,k}$ and parallel to $\vec{n}$ must intersect the exterior boundary at least twice. Define **A** and **B** the two closest intersections having opposite scalar product with respect to $\vec{n}$ computed from $\mathbf{t}_{j0,k}$ (ref Fig. 3.14(c)). Consider the set $\mathcal{M}$ of the exterior vertices traversed from **A** to **B** in counter-clockwise orientation.

We would like to introduce now the set

$$\kappa := \{\mathbf{s}_i \text{ vertex of } \mathcal{M} : \lambda(\mathbf{s}_i) > \lambda(\mathbf{t}_{j0,k})\} \tag{3.32}$$

which must be nonempty because the polygonal boundary $\mathcal{E}$ is outside $\mathcal{J}_{j0}$. Therefore, we may define

$$\mathbf{s}_l := \text{argmin}\{\lambda(\mathbf{s}_i) \,:\, \mathbf{s}_i \in \kappa\}. \tag{3.33}$$

We note that no edge of the polygon P can intersect the cut $[\mathbf{s}_l, \mathbf{t}_{j0,k}]$ because there is no internal node in the region

$$\{\mathbf{x} \in \mathbf{R}^2 : \lambda(\mathbf{x}) > \lambda(\mathbf{t}_{j0,k})\}. \tag{3.34}$$

$\square$

Theorem 4 From any multiply connected polygon P with h internal boundaries, one can generate a double-edged polygon $\tilde{P}$ by inserting h cuts.

Proof

We will proceed by induction according to the number h of internal curves. Let us adopt the notations $\mathbf{s}_i$ and $\mathbf{t}_{j,k}$ of lemma 1 to denote the vertices on the exterior and interior boundaries respectively. In the case that $h = 1$, let us denote $\mathbf{t}_k := \mathbf{t}_{1,k}$ and $n := n_1$. According to lemma 1, we may insert a cut $[\mathbf{s}_l, \mathbf{t}_k]$ inside the polygon. In order to simplify the notation, let us extend the indices in the following periodic manner:

$$\begin{aligned} \mathbf{s}_r &:= \mathbf{s}_{r-m} \quad \text{if } r \geq m && (3.35)\\ \mathbf{t}_r &:= \mathbf{t}_{r-n} \quad \text{if } r \geq n. && (3.36) \end{aligned}$$

Let us now introduce the polygon $\tilde{P}$ whose vertices are defined by

$$\begin{array}{lcll} \mathbf{y}_i & := & \mathbf{s}_{l+i} & \text{for} \quad i = 0, \cdots, m \\ \mathbf{y}_{m+i} & := & \mathbf{t}_{k+i-1} & \text{for} \quad i = 1, \cdots, n \\ \mathbf{y}_{m+n} & := & \mathbf{t}_{k+n}. & \end{array} \qquad (3.37)$$

As one can easily observe, the polygon $\tilde{P}$ having $(n + m + 2)$ vertices is a double-edged polygon in which the edge $e := [\mathbf{s}_l, \mathbf{t}_k]$ is traversed twice in opposite direction.

Let us suppose that the theorem is true for a polygon with h holes. Consider a polygon P with $(h + 1)$ holes. Introduce a cut f by applying the former lemma to P and generate a new exterior polygon $\tilde{\mathcal{E}}$ from $\mathcal{E}$ and $\mathcal{J}_{j_0}$ as we did in (3.37). Now, define Q as the multiply connected polygon having $\tilde{\mathcal{E}}$ as exterior polygon and $\mathcal{J}_j$ $j \neq j_0$ as interior ones. Since the number of holes of the polygon Q is h, we may apply the hypothesis of induction in order to get the double-edged polygon $\tilde{Q}$ by inserting h cuts $e_1,\cdots,e_h$. Thus, we have generated a double-edged polygon $\tilde{P}$ by inserting the following $(h + 1)$ cuts:

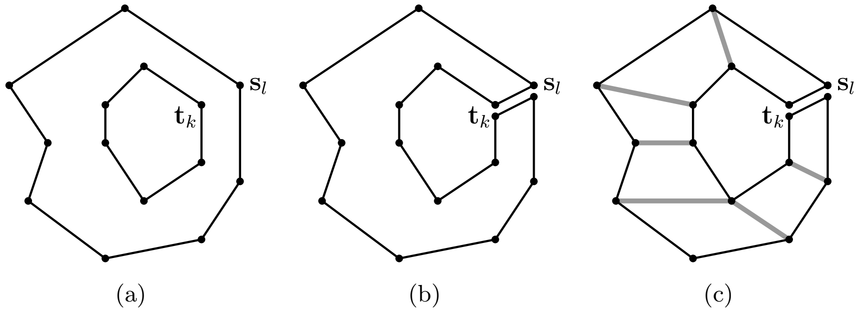
$$f, e_1, \cdots, e_h. \qquad (3.38)$$

□

Theorem 5 Consider a double-edged polygon P having at least four vertices, one of the following statements holds:

(F1) One can remove a quadrilateral which is not necessarily convex by inserting a single cut as in Fig. 3.3(a).

(F2) There exists a point ω located strictly within P such that one can remove a *convex* quadrilateral by inserting two line segments emanating from ω to two vertices of P (Fig. 3.3(b)).



(a) (b) (c)

Figure 3.15: Quadrilating a multiply connected polygon

Proof

Corollary 2 shows that the two-ear theorem holds true for double-edged polygons. Therefore, we can repeat the proof of theorem 2 without any change because the generalized two-ear theorem can be used to reduce the number of vertices in the induction.

□

3.7 Cut search

Suppose we have a multiply connected polygon P having $\mathcal{E}$ as exterior boundary and $\mathcal{J}_i$ as interior boundaries. We want to find a cut connecting an internal boundary $\mathcal{J}_i$ and the exterior boundary $\mathcal{E}$. It seems reasonable to prefer short cuts from long ones. Therefore, we introduce a threshold L for the length of an admissible cut. Since we aim at nicely shaped quadrilaterals we also want to control the angles that are created by the introduction of a cut. For a cut connecting an internal vertex $\mathbf{x}$ and an external vertex $\mathbf{y}$, we introduce the four angles α, β, γ, δ which are made with the boundary as depicted in Fig. 3.16(a). In order to evaluate the angular quality of the cut $[\mathbf{x},\mathbf{y}]$, we use the minimum of those four angles. That is, we will say that the cut $[\mathbf{x},\mathbf{y}]$ has an acceptable angle if for some prescribed angle θ_0 we have

$$\theta := \min\{\alpha,\beta,\gamma,\delta\} \geq \theta_0. \tag{3.39}$$

According to the definition of a cut from section 3.3, the most straightforward approach to find a cut would be to perform intersection tests between every possible segment connecting two vertices of P with all edges. Since that approach is very costly, we will propose a more efficient method.

Our cut search proceeds in two phases. In the first phase, we search for cuts which connect extreme vertices (see Definition in section 3.7.1) and the exterior boundary. In most practical cases, this phase is enough to find a cut. But if it fails, then we must consider more general cuts in the second phase. That means, we search for a cut which connects the exterior boundary and a non-extreme internal vertex. After giving some details about those two phases, we will summarize the process in form of an algorithm.

3.7.1 Cuts having extreme vertices

In the sequel, we will denote by E (resp. I) the set of all exterior (resp. interior) vertices of a given polygon P.

Definition 9 Consider a direction specified by a unit vector W in the plane. An extreme vertex with respect to W is a vertex $\mathbf{x}_W \in I$ which maximizes the scalar product $\left\langle \overrightarrow{\mathbf{0x}}_W, W \right\rangle$, that is

$$\mathbf{x}_W := \operatorname{argmax}\left\{\left\langle \overrightarrow{\mathbf{0x}}, W \right\rangle \,:\, \mathbf{x} \in I\right\}. \tag{3.40}$$

Additionally, we define:

$$\mathcal{R}_W := \left\{\mathbf{y} \in E \,:\, \left\langle \overrightarrow{\mathbf{0y}}, W \right\rangle \geq \left\langle \overrightarrow{\mathbf{0x}}_W, W \right\rangle\right\}. \tag{3.41}$$

In order to test whether $[\mathbf{x}_W, \mathbf{y}]$ is a cut for $\mathbf{y} \in \mathcal{R}_W$, we do not need to do any intersection test between the segment $[\mathbf{x}_W, \mathbf{y}]$ and internal edges. Let us denote by ζ the set of exterior edges which lie partly or completely inside $\mathcal{R}_W$. We have only to make intersection tests of $[\mathbf{x}_W, \mathbf{y}]$ with members of ζ. Thus, let us introduce

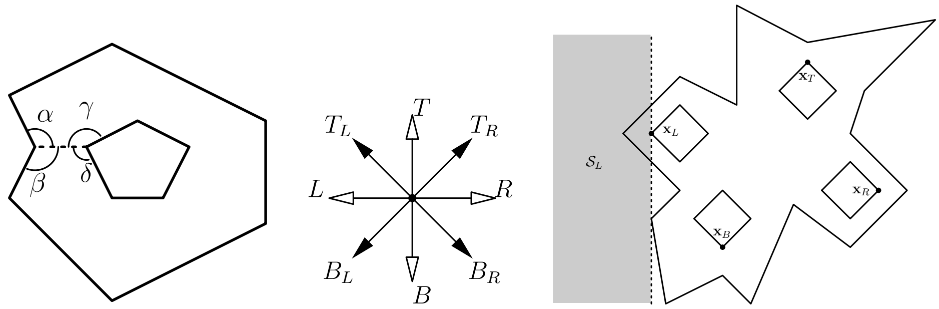
$$\mathcal{S}_W := \{\mathbf{y} \in \mathcal{R}_W \,:\,]\mathbf{x}_W, \mathbf{y}[\cap e = \emptyset \qquad \forall e \in \zeta\}. \tag{3.42}$$

In practical implementations, one will use four to eight directions (see Fig. 3.16(b)) specified by the following vectors $T := (0,1)$, $B := (0,-1)$, $R := (1,0)$, $L := (-1,0)$, $T_R := (\sqrt{2}/2, \sqrt{2}/2)$, $T_L := (-\sqrt{2}/2, \sqrt{2}/2)$, $B_L := (-\sqrt{2}/2, -\sqrt{2}/2)$, $B_R := (\sqrt{2}/2, -\sqrt{2}/2)$.

3.7.2 Cuts having nonextreme vertices

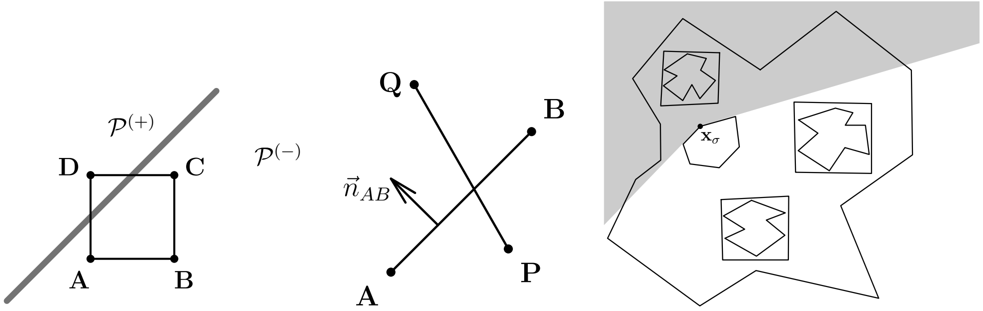
Finding a cut $[\mathbf{x}_\sigma, \mathbf{y}]$ connecting a non-extreme vertex $\mathbf{x}_\sigma \in I$ and an exterior vertex $\mathbf{y} \in E$ is difficult because intersections with internal boundaries may be involved. In order to make the intersection tests more efficient, we introduce the axiparallel bounding boxes $\mathcal{B}_i$ of the internal boundaries $\mathcal{J}_i$. Before performing intersection tests between $[\mathbf{x}_\sigma, \mathbf{y}]$ and edges belonging to the internal boundary $\mathcal{J}_i$, we do an intersection test with the bounding box $\mathcal{B}_i$. If that fails, then the line segment $[\mathbf{x}_\sigma, \mathbf{y}]$ does not intersect any edge of $\mathcal{J}_i$.

We only use cuts $[\mathbf{x}_\sigma, \mathbf{y}]$ such that $\mathbf{x}_\sigma$ are *reflex* internal vertices. Suppose that we want to test if a cut emanating from an internal reflex vertex $\mathbf{x}_\sigma$ exists. Note that



(a) (b) (c)

Figure 3.16: (a)Angles made by a cut (b)Eight directions for cut search (c)The extreme vertex with respect to V_L and the region $\mathcal{S}_L$



(a) (b) (c)

Figure 3.17: (a),(b)Segment rectangle intersection (c)Domain of interest

we do not need to perform intersection tests with all existing bounding boxes. Denote by $\mathbf{x}_s$ and $\mathbf{x}_p$ the successor and predecessor of $\mathbf{x}_\sigma$ in the internal polygon containing $\mathbf{x}_\sigma$. The relevant domain $\mathcal{R}_\sigma$ corresponding to the vertex $\mathbf{x}_\sigma$ is the complement of the intersection of the half-planes $(\mathbf{x}_\sigma\mathbf{x}_p)^+$ and $(\mathbf{x}_\sigma\mathbf{x}_s)^-$. As an illustration, we identify by a shaded region in Fig. 3.17(c) the relevant domain of $\mathbf{x}_\sigma$. We need only to perform intersection tests with the bounding boxes which intersect the relevant domain $\mathcal{R}_\sigma$. Let us also introduce the set T_σ which is the list of exterior vertices $\mathbf{y} \in \mathcal{R}_\sigma \cap E$ such that the line segment $]\mathbf{x}_\sigma, \mathbf{y}[$ has no intersection with any exterior edge.

For the polygon P, let $(r(i))_{i=1}^q$ be the indices of the extreme vertices with respect to a considered set of directions. Define

$$\mathcal{X} := \{\mathbf{x}_{r(i)} : \quad i = 1, \cdots, q\} \tag{3.43}$$

One way of searching for a cut emanating from a nonreflex vertex $\mathbf{x}_\sigma \in I \setminus \mathcal{X}$ is to traverse the set $I \setminus \mathcal{X}$ in a random order. As an alternative to that, we would like to assemble a priority queue $(\mathbf{w}_k)_{k=1}^K$ which is a sequence of non-reflex vertices $\mathbf{x}_\sigma \in I$ that are not extreme ones. Before defining $(\mathbf{w}_k)_{k=1}^K$ recursively, let us introduce some notations. Let a be an internal vertex belonging to an internal curve $\mathcal{J}_w$ which has a clockwise orientation. We will denote by $a^{(s)}$ the reflex vertex following a along $\mathcal{J}_w$ in clockwise direction. Similarly $a^{(p)}$ is the reflex vertex following a along $\mathcal{J}_w$ in counter-clockwise direction. The superscripts (s) and (p) are descriptive abbreviations of 'predecessor' and 'successor' respectively. Let $\mathcal{N}$ be the set of internal reflex vertices which do not belong to $\mathcal{X}$. Before introducing the sequence $(\mathbf{w}_k)_{k=1}^K$, let us define a sequence $\mathbf{z}_{k,i}$ for every $i = 1, ..., q$. Initialize

$$\mathbf{z}_{0,i} := \mathbf{x}_{r(i)}^{(p)} \qquad \mathbf{z}_{1,i} := \mathbf{x}_{r(i)}^{(s)}. \tag{3.44}$$

The sequence $\mathbf{z}_{k,i}$ is defined recursively by

$$\mathbf{z}_{k+1,i} := \mathbf{z}_{k-1,i}^{(p)} \qquad \mathbf{z}_{k+2,i} := \mathbf{z}_{k,i}^{(s)}. \tag{3.45}$$

We stop the above updating as soon as $\mathbf{z}_{k,i}^{(p)}$ and $\mathbf{z}_{k-1,i}$ coincide. The set $\mathcal{N}_0 := \mathcal{N} \setminus \{\mathbf{z}_{k,i}\}$ is the collection of internal reflex vertices which come from curves that do not contain extreme vertices. The sequence $(\mathbf{w}_k)_{k=1}^K$ is defined to be the lexicographical ordering of $\mathbf{z}_{k,i}$ followed by the vertices of $\mathcal{N}_0$.

Since segment-rectangle intersection operations are frequently used, we are going to briefly describe a good method to perform them. The process of intersection between a rectangular domain $[\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}]$ and a segment $[\mathbf{P}, \mathbf{Q}]$ is very well-known in computer graphics as a clipping operation. As a first step, we can consider the infinite line which is generated by the segment $[\mathbf{P}, \mathbf{Q}]$. That line splits the plane into two half-planes $\mathcal{P}^{(+)}$ and $\mathcal{P}^{(-)}$ which are supposed to be closed (thus, the line belongs to both of them). As necessary condition so that there is an intersection, every half-plane must contain at least one vertex from among **A**, **B**, **C**, **D**. If that first step succeeds, we use a simplified Liang-Barsky method for testing the intersection of $[\mathbf{P},\mathbf{Q}]$ and the four edges of the rectangle. While the original Liang-Barsky algorithm which treats rectangles having edge parallel to the axes aims at determining the coordinates of intersection points, we need only a Boolean answer of whether or not the segment intersects the rectangle. In order that $[\mathbf{P},\mathbf{Q}]$ intersects $[\mathbf{A},\mathbf{B}]$, the following two products have to be nonnegative (see Fig. 3.17)

$$<\overrightarrow{\mathbf{BQ}},\vec{n}_{AB}>\cdot<\overrightarrow{\mathbf{BP}},\vec{n}_{AB}>,\qquad <\overrightarrow{\mathbf{QA}},\vec{n}_{PQ}>\cdot<\overrightarrow{\mathbf{QB}},\vec{n}_{PQ}>,\tag{3.46}$$

where $\vec{n}_{XY}$ denotes the unit normal vector to the segment $[\mathbf{X},\mathbf{Y}]$.

3.7.3 Summary

Now we will describe our cut search algorithm. The following algorithm EXTREME() searches for a cut e which emanates from an extreme vertex. Its input includes a length threshold L and an angle threshold θ_0. We will say that an edge has the desired quality if it fulfills the angle criterion on (3.39) with respect to the angle threshold θ_0 and if its length is smaller than the length threshold L.

Algorithm (EXTREME)

```
step 0 : Initialize the list of directions: $\mathcal{D} :=\{T,B,R,L\}$ or
         $\mathcal{D} :=\{T,B,R,L,T_L,T_R,B_L,B_R\}$
step 1 : For every direction $D \in \mathcal{D}$,
           • Find the extreme vertex $\mathbf{x}_D \in I$,
           • Find the set $\mathcal{S}_D$ as defined in (3.42),
           • For every member $\mathbf{y}_D$ of $\mathcal{S}_D$, consider $e = [\mathbf{x}_D, \mathbf{y}_D]$. If
             $e$ has the desired quality it is an acceptable cut and we
             terminate. Otherwise, $e$ is included into a list $\mathcal{C}$.
step 2 : If no acceptable cut is found, return FAILURE.
```

If the algorithm EXTREME() succeeds, we have found a cut. Otherwise, we need the assembled list $\mathcal{C}$ for the following algorithm NONEXTREME() which needs also an angle threshold θ_0 and a length threshold L.

Algorithm (NONEXTREME)

```
step 0 : Find the bounding boxes $\mathcal{B}_i$ of the internal curves $\mathcal{J}_i$.
step 1 : Determine the priority queue $(\mathbf{w}_k)_{k=1}^{K}$ from (3.45).
step 2 : For every $k = 1, ..., K$ define $\mathbf{x}_\sigma := \mathbf{w}_k$ and do the following:
```

```
step 2.1 : Find the relevant domain $\mathcal{R}_\sigma$ of $\mathbf{x}_\sigma$ and the set $T_\sigma$.
step 2.2 : For every $\mathbf{y}_\sigma \in T_\sigma$ define $e := [\mathbf{x}_\sigma, \mathbf{y}_\sigma]$, find $J$ with $\mathcal{B}_i \cap \mathcal{R}_\sigma \neq \emptyset$
           for $i \in J$, then
             • Test if $\mathcal{B}_i \cap e = \emptyset$ $(i \in J)$. If not, test intersection with
               the edges of the internal curves $\mathcal{J}_i$ $(\forall i \in J)$.
             • If all intersection tests with relevant internal curves fail
               and if $e$ has the desired quality, it is an acceptable cut
               and we terminate. Otherwise, $e$ is included into $\mathcal{C}$.
step 3   : If no acceptable cut is found, return FAILURE.
```

The cut search can then be summarized as combination of the above two algorithms.

Algorithm (Single cut search)

```
step 0 : Prescribe an angle threshold $\theta_0$ of relation (3.39) and a length
         threshold $L$. Initialize the list of cuts $\mathcal{C}$ as the empty set
step 1 : Invoke EXTREME($\mathcal{C}$,$\theta_0$,$L$). If it succeeds in finding an acceptable cut, we terminate the algorithm.
step 2 : Invoke NONEXTREME($\mathcal{C}$,$\theta_0$,$L$). If it succeeds in finding an
         acceptable cut, we terminate the algorithm.
step 3 : If step 2 fails because the prescribed angle $\theta_0$ from step 0 is too
         large, reduce its value by a positive factor (say $\theta_0 := .75 * \theta_0$)
         and go back to step 0.
```

Note that most cuts are found based on extreme vertices in practice.

3.8 Conversion into convex quadrangulation

The purpose of this section is to describe (see also [16, 100]) how to convert a quadrangulation which has non-convex quadrilaterals, such as those obtained from former sections, into another quadrangulation whose members are exclusively convex quadrilaterals. Before giving details about the conversion, let us consider the following remark. Two adjacent quadrilaterals q and p form a single quadrilateral or a hexagon. In the first case, the quadrilaterals q and p share two edges and it is possible that the union $q \cup p$ is a nonconvex or a convex quadrilateral as illustrated in Figs. 3.18(a) and 3.18(b). In the second case, only one edge is shared by q and p as illustrated in Fig. 3.18(c).

Now, let us recall the following result about hexagon quadrangulations which are

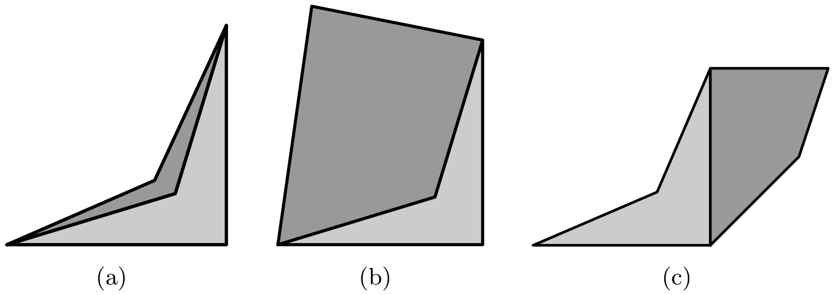

(a) (b) (c)

Figure 3.18: The union of two quadrilaterals is (a) a nonconvex quadrilateral (b) a convex quadrilateral (c) a hexagon.

needed in the description of the conversion algorithm.

Theorem 6 (Bremner *et. al*[16]) Every hexagon (which may include reflex vertices) can be decomposed into a set of convex quadrilaterals by using at most three internal Steiner points.

Remark 6 There are a lot of different cases in the proof and implementation of this theorem . Those cases are listed in section 3.16.2 about special splittings.

Based on those facts, we may describe the conversion algorithm in two steps:

Step1: For every nonconvex quadrilateral p having a neighboring quadrilateral q such that $p \cup q$ is a quadrilateral, replace p by $p \cup q$ and remove the quadrilateral q from the quadrangulation. We repeat this step until such a union does not exist any more. After this step, there can only exist nonconvex quadrilaterals whose union with a neighboring quadrilateral forms a hexagon.

Step2: We merge a nonconvex quadrilateral q of Q with a neighboring quadrilateral p in order to have a hexagon $(q \cup p)$. If we have the choice then we select a nonconvex neighbor p. Then, we re-quadrilate the resulting hexagon by using the hexagon quadrangulation method from theorem 6 in order to obtain a local *convex* quadrangulation Q_{loc}. Afterwards, we substitute the union $(q \cup p)$ by Q_{loc} in the quadrangulation Q.

3.9 Cleanup

In many decomposition techniques [78, 91, 48], *cleanup* is the process of generating a polygon tessellation by improving an available one according to some quality criteria. Some cleanup procedures aim at obtaining quadrilateral quality enhancement by introducing new nodes and edges. Contrary to this approach, we are going to keep the numbers of nodes and edges unchanged but we change the position of nodes and edges in order to enhance the quadrangulation quality. Before describing our method, let us note that before cleanup operation, we have already a quadrangulation which fulfills our desired properties in four-sided splitting: we use only boundary nodes and all quadrilaterals are convex. All that we want to achieve with cleanup operations is to enhance the quality of the quadrangulation.

3.9.1 Quality control

Before describing the cleanup operations, let us review the way qualities of a quadrilateral, a node or an edge can be evaluated. We consider two techniques of assessing the quality of a quadrilateral $[A, B, C, D]$. The first one requires the introduction of the following distortion coefficient [80] of any triangle $[a, b, c]$:

$$\alpha := 2\sqrt{3}\frac{\|\overrightarrow{ca} \times \overrightarrow{cb}\|}{\|\overrightarrow{ca}\|^2 + \|\overrightarrow{ab}\|^2 + \|\overrightarrow{bc}\|^2} \in [0, 1]. \tag{3.47}$$

We can easily see that the triangular distortion α is unity if the triangle $[a, b, c]$ is equilateral. From a convex quadrilateral $[A, B, C, D]$, we may derive four triangles $[A, B, C]$, $[A, C, D]$, $[A, B, D]$ and $[D, B, C]$. Let us denote by α_i the triangular distortions of those four triangles such that $\alpha_1 \geq \alpha_2 \geq \alpha_3 \geq \alpha_4$. We define the first quality measurement of the quadrilateral $[A, B, C, D]$ to be

$$\beta := (\alpha_3\alpha_4)/(\alpha_1\alpha_2) \in [0, 1]. \tag{3.48}$$

Effectively, the value of β is unity for rectangles and it approaches zero as a quadrilateral becomes triangular shaped. The second way of measuring qualities of quadrilaterals is by means of the smallest internal angle in the above four triangles. In the following discussion, we will denote by $\mu(q)$ a quality measure of a quadrilateral q.

We would like to consider quality measurements of a node and an edge inside a quadrangulation. To that end, let us consider an internal node ω which is shared by the quadrilaterals q_i $i \in \mathcal{J}$. We may now evaluate the quality of the node ω by

$$\mu(\omega) := \frac{1}{\mathrm{card}(\mathcal{J})} \sum_{i \in \mathcal{J}} \mu(q_i). \tag{3.49}$$

Similarly, for an internal edge e upon which two quadrilaterals q_1 and q_2 are incident, its quality can be measured by

$$\mu(e) := \frac{1}{2}[\mu(q_1) + \mu(q_2)]. \tag{3.50}$$

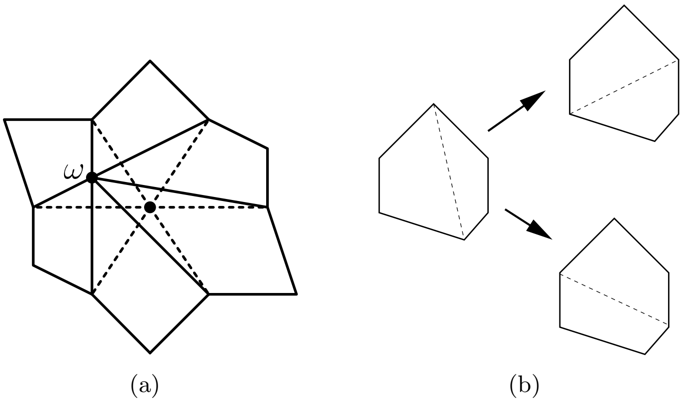


Figure 3.19: Cleanup: (a) Shifting a node ω(b) Flipping an edge

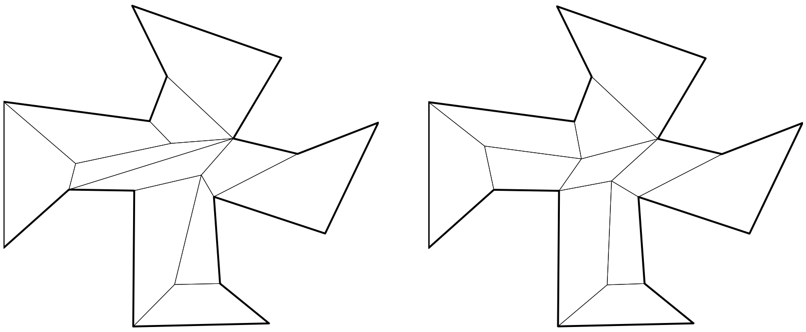
3.9.2 Cleanup operations

We will treat two types of cleanup operations: *node repositioning* and *edge flipping*. The first one consists in shifting an internal node to another position in order to improve the quality of the neighboring quadrilaterals. In the course of node shifting, we have to make sure that all incident quadrilaterals remain convex. The second operation modifies the endpoints of an internal edge.

Let us first show how to find the region inside which a node ω can be shifted. Consider the region $\mathcal{R}$ which is formed by the kernel of the union of the surrounding quadrilaterals. On account of the convexity of the incident quadrilaterals, we note that $\mathcal{R}$ is nonempty because at least ω itself is a member of this region. The node repositioning consists in moving ω inside the interior of $\mathcal{R}$ in order to minimize $\mu(\omega)$.

Since determining the set $\mathcal{R}$ is expensive in practice, we want to propose now another method which executes faster. Let us denote by E_ω the set of edges which emanate from the node ω. Then we take the shortest edge $\tilde{e}$ from among E_ω and consider a circle centered at the node ω and having radius $\rho := \lambda \cdot \text{length}(\tilde{e})$ where λ is a user defined parameter (say λ=0.25) from $]0, 1[$. The new position of ω is then searched inside this circle. The practical realization of such a shifting is to pick p (say $p = 5$) positions q_i inside the circle. For every q_i, we test if by replacing ω by q_i, we would still have incident convex quadrilaterals. That can be easily checked by testing if the new angles made with edges emanating from q_i are smaller than π. We replace then ω by q_i which gives new incident quadrilaterals and which minimizes $\mu(q_i)$. If none of the q_i fulfills those desirable properties, then we keep ω in its current position.

The second operation consists in flipping an edge in order to improve the qualities of the neighboring quadrilaterals. In the best case, there are two possibilities for flipping an edge by considering the union of the incident quadrilaterals as



(a) (b)

Figure 3.20: The numbers of nodes and edges are unchanged: (a)Before cleanup (b)After cleanup

explained in Fig. 3.19(b). We flip an internal edge e to a position which keeps the two incident quadrilaterals convex and which improves the value of $\mu(e)$.

In Fig. 3.20, we can see a result of the applications of the cleanup operations. There is no exact rule about where to start but the method that many people [34, 122, 78, 91, 48] use in such a retouching technique is to start from the worst entities. That means, one measures the qualities $\mu(\omega_i)$ of all internal nodes. One searches for the m (say $m = 5$ or m is the number of all internal nodes) nodes having the largest μ values. Then one applies the cleanup operations to those nodes. One can repeat that operation, a few number of times. One can do also the same thing for edges. It is also possible to alternate the edge and node quality improvements.

3.10 Converting a triangulation into a quadrangulation

Now we would like to describe briefly another quadrangulation method that can be used as an alternative to the one that we presented before. It starts by generating a triangular mesh $\mathcal{M}$ on the initial multiply connected polygon. Now we want to discuss how to transform the triangular mesh $\mathcal{M}$ into a quadrangulation $\mathcal{Q}$. The simplest method is to generate three quadrilaterals from one triangle τ by inserting Steiner points in the edges of τ and at its center of gravity as graphically illustrated in Fig. 3.22(a). We do not favor that idea because it introduces unnecessarily many Steiner points which creates too many quadrilaterals. In addition, Steiner points can be located on the boundary of the mesh $\mathcal{M}$. We need a conversion technique [16] that only inserts very few Steiner points which all reside in the interior of $\mathcal{M}$.

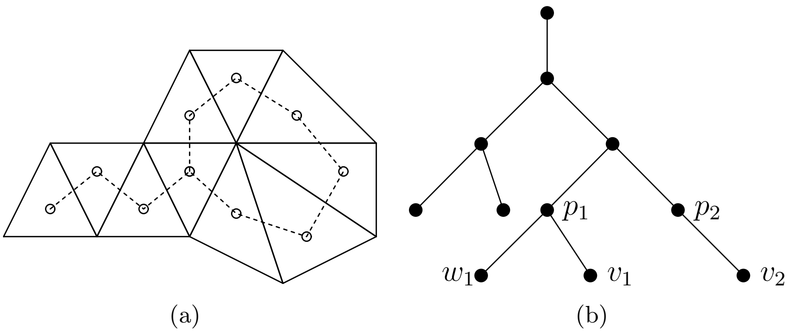


Figure 3.21: (a) Edge-graph of a mesh (b) w_1 is the sibling of the vertex v_1

Conversion approach:

Let us consider the edge-graph $\mathcal{G}$ (Fig. 3.21(a)) of the triangular mesh $\mathcal{M}$ which is supposed to have an even number of triangles. Readers are warned because the terms *vertex* or *node* could become confusing when considering the mesh $\mathcal{M}$ and the graph $\mathcal{G}$ since a vertex in the graph $\mathcal{G}$ corresponds to a triangle in the mesh $\mathcal{M}$. The fundamental idea in the conversion is to merge neighboring triangles to form quadrilaterals. First, we initialize $\mathcal{Q}$ as an empty quadrangulation which has a zero number of quadrilaterals. From the graph $\mathcal{G}$, we can consider a spanning tree $\mathcal{T}$ whose root is supposed to have only one child. We note that every vertex of the tree $\mathcal{T}$ has at most a valence three because a triangle in the mesh $\mathcal{M}$ has at most three neighbors. The algorithm of conversion considers a bottom leave v of the spanning tree $\mathcal{T}$. We need to distinguish two cases according to the nature of the parent vertex p of v in the spanning tree $\mathcal{T}$.

Case 1: If the vertex p has valence three, we consider the sibling w of the vertex v in the tree $\mathcal{T}$. Let us denote by $P := [b, d, e]$, $V := [a, b, e]$, $W := [b, c, d]$ the triangles (Fig. 3.22(b)) which correspond to the vertex p, v, w respectively. We generate a node ω inside the triangle P and we define two new quadrilaterals $q_1 := [a, b, \omega, e]$ and $q_2 := [d, \omega, b, c]$ as illustrated in Fig. 3.22(c). After incorporating the quadrilaterals q_1 and q_2 in the quadrangulation $\mathcal{Q}$, we update the spanning tree $\mathcal{T}$ by removing the two leave vertices v and w. Thus, the vertex p becomes a node vertex of $\mathcal{T}$ and we substitute the vertices of P by $[\omega, d, e]$.

Case 2: If the parent vertex p has a valence two or if it is the root of the spanning tree $\mathcal{T}$, then we merge the triangles $P := [b, d, e]$ and $V := [a, b, e]$ in order to have a single quadrilateral $q := [a, b, d, e]$ which we incorporate in the quadrangulation $\mathcal{Q}$. As before, we update $\mathcal{T}$ by removing the bottom vertex v.

Minimum spanning tree:

Note that a bad choice of the spanning tree $\mathcal{T}$ affects the quality of the ultimate quadrangulation Q. That can be illustrated by considering the simple triangular mesh in Fig. 3.23(a). Both of the quadrangulations in Fig. 3.23(b) and

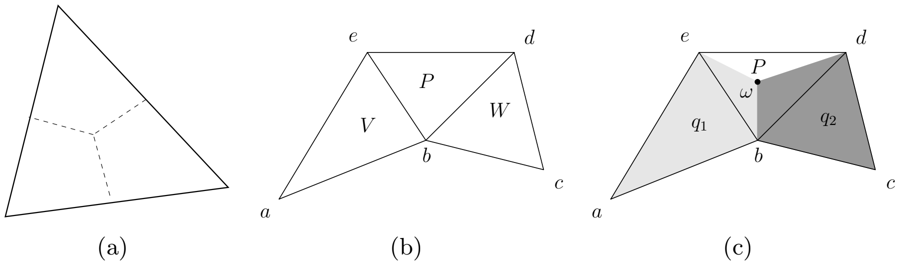


Figure 3.22: (a)Simplest conversion method (b),(c)Form two quadrilaterals $[a, b, \omega, e]$ and $[d, \omega, b, c]$ and modify the triangle P.

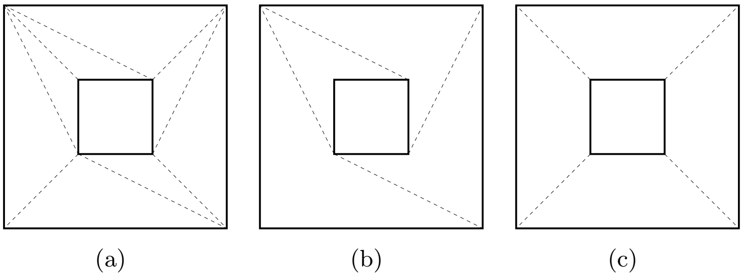


Figure 3.23: (a)Triangulation (b),(c)Two possible quadrangulations

Fig. 3.23(c) result from the former conversion algorithm by using two different spanning trees. One can immediately observe that Fig. 3.23(b) displays a bad quadrangulation whose quadrilaterals are all nonconvex. In order to properly select a spanning tree, we assign weights to the edges of the graph $\mathcal{G}$ in the following way. Consider a mesh-edge E which is incident upon two triangles P and Q. If the union $P \cup Q$ forms a convex quadrilateral, we assign a small weight (say 0.1) to the graph-edge e corresponding to E. Otherwise, a heavy weight (say 10) will be assigned to e. Thus, the selection is performed by searching the spanning tree $\mathcal{T}_{\min}$ with minimal weights. That way, the likelihood that a nonconvex quadrilateral appears in the quadrangulation Q is minimized. In order to determine the minimum spanning tree (MST) of a weighted graph, several algorithms such as Prim or Kruskal methods [83] can be used.

Although MST algorithms minimize the occurrence of nonconvex quadrilaterals, it is still possible that there are very few ones in the final quadrangulation Q. In such a rare case, we merge a nonconvex quadrilateral q of Q with a neighboring quadrilateral $\overline{q}$. Then, we requadrangulate the hexagon composed of $q \cup \overline{q}$ by using the hexagon quadrangulation method that we described in section 3.16.2 in order to obtain a local *convex* quadrangulation Q_{loc} of $q \cup \overline{q}$. Afterwards, we substitute the union $q \cup \overline{q}$ by Q_{loc} in the quadrangulation Q.

Apart from the cost of generating an initial triangulation, the main drawback of the triangular conversion is that the process of finding the minimal spanning tree is computationally expensive.

3.11 Regions having curved boundaries

In the preceding sections, we have focused on multiply connected regions having polygonal boundaries. From this section on, we would like to describe a method of splitting a multiply connected trimmed surface $\mathcal{D} \subset \mathbf{R}^2$ having curved boundaries into several four-sided domains:

$$\mathcal{D} = \bigcup_k Q_k. \tag{3.51}$$

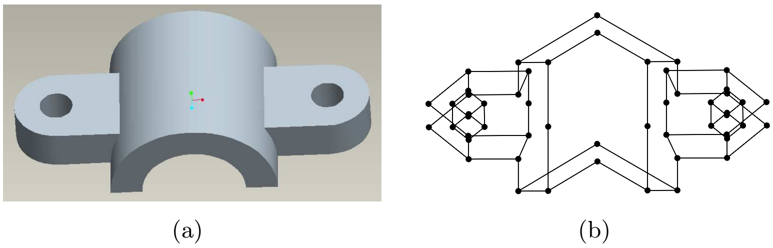

(a) (b)

Figure 3.24: (a)Given model in IGES format (b)Initial polygonal approximation

Basically, the decomposition method is performed as follows. First, we take a coarse polygonal approximation P of the curved boundaries of $\mathcal{D}$. Then, we generate a quadrangulation of the resulting polygon P in order to have $P = \cup_k q_k$ where q_k are convex quadrilaterals. In order to generate the four-sided subregion Q_k in (3.51) from the quadrilateral q_k, we keep internal edges of q_k and we replace boundary edges of q_k by the corresponding curved boundary portion. At this point, there could exist intersections between boundary curves and internal edges of the quadrangulation. We will show a method of repairing such boundary interferences. In the following sections, we will give details of the above process one by one.

3.12 Structure of the polygonal model

Consider a set of surfaces $\{S_i\}_{i\in\Lambda}$ such that each S_i is the image by ψ_i of a 2D multiply connected domain $\mathcal{D}_i$ having curved boundaries. A direct application of the above polygonal approximation to each $\mathcal{D}_i$ may generate hanging nodes. That is, there might exist two adjacent surfaces S_i and S_j such that an image by ψ_i of a vertex of the polygonal approximation of $\mathcal{D}_i$ does not coincide to the image by ψ_j of any vertex of the polygonal approximation of $\mathcal{D}_j$.

3.12.1 Discretization of curved boundaries

The discretization of the domain $\mathcal{D}_i$ which maps to S_i involves the other surfaces S_j that are incident upon S_i. Therefore, let us introduce the following notion of adjacency graph.

Definition 10 Consider a given set of surfaces $\{S_i\}_{i\in\Lambda}$ whose union forms a closed surface S. The surface S_i will be called the i-th face of S. We can generate a graph $\mathcal{G}$ which stores the adjacency of the faces S_i. That is, for every face S_i, we generate a node n_i in the graph $\mathcal{G}$ and we add a graph edge $[n_i, n_j]$ if the corresponding faces S_i and S_j are adjacent. Such a graph will be called *adjacency graph* corresponding to $\{S_i\}_{i\in\Lambda}$.

Before giving any description of the discretization method, let us explain explicitly the process of having polygonal approximations (Fig. 3.24) of a set of domains $\{\mathcal{D}_i\}_{i\in\Lambda}$.

Definition 11 Suppose that we have trimmed surfaces S_i which are given as images of

$$\psi_i : \mathcal{D}_i \longrightarrow S_i. \tag{3.52}$$

Finding polygonal approximations of $\{\mathcal{D}_i\}_{i\in\Lambda}$ amounts to doing the following. For each trimmed surface S_i, find a polygon $P^{(i)}$ whose nodes $\mathbf{x}_k^{(i)}$ are taken from the curved boundary of the 2D domain $\mathcal{D}_i$ such that for two adjacent different surfaces S_i and S_j which share a curve $\mathcal{C}$, if $\psi_i(\mathbf{x}_k^{(i)}) \in \mathcal{C}$, then there must exist a vertex $\mathbf{x}_l^{(j)} \in P^{(j)}$ with

$$\psi_i(\mathbf{x}_k^{(i)}) = \psi_j(\mathbf{x}_l^{(j)}). \tag{3.53}$$

Remark 7 As we have mentioned in section 2.3 of chapter 2, the segment separators which are provided by the IGES files already satisfy the condition in (3.53). Therefore, it is a good choice to select the segment separators of $\mathcal{D}_i$ as initial vertices of its polygonal approximation $P^{(i)}$.

3.12.2 Edge splitting

The process of generating a polygonal approximation $P^{(i)}$ of $\mathcal{D}_i$ consists in starting from a very coarse polygon having vertices from the boundary of $\mathcal{D}$. Then, one refines the polygonal approximation by inserting new nodes repeatedly (see Fig. 3.25). In this section, we show the tasks that have to be performed in order to generate new nodes on polygonal approximations . The most difficult process is to ensure that there are no hanging nodes at the interfaces of the surfaces S_i. In order to facilitate the presentation, we suppose that each $\mathcal{D}_i$ is simply connected and its boundary is given by a parametric curve κ_i. Consider the polygon $P^{(i)}$

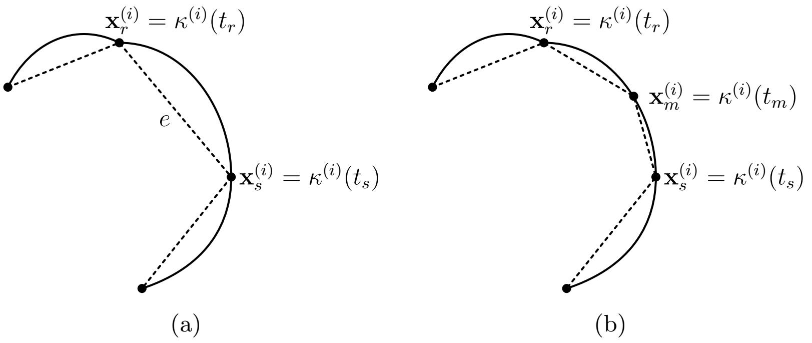

$\mathbf{x}_r^{(i)} = \kappa^{(i)}(t_r)$
e
$\mathbf{x}_s^{(i)} = \kappa^{(i)}(t_s)$
(a)
$\mathbf{x}_r^{(i)} = \kappa^{(i)}(t_r)$
$\mathbf{x}_m^{(i)} = \kappa^{(i)}(t_m)$
$\mathbf{x}_s^{(i)} = \kappa^{(i)}(t_s)$
(b)

Figure 3.25: Inserting a new node $\mathbf{x}_m^{(i)}$ between the endpoints of the edge e.

which approximates the domain $\mathcal{D}_i$. Let $e = [\mathbf{x}_s^{(i)}, \mathbf{x}_r^{(i)}]$ be an edge of $P^{(i)}$ and let us suppose that $\mathbf{x}_s^{(i)} = \kappa^{(i)}(t_s)$ and $\mathbf{x}_r^{(i)} = \kappa^{(i)}(t_r)$. Refining the edge e amounts to doing the following:

(S1) Define $\mathbf{x}_m^{(i)} := \kappa^{(i)}(t_m)$ with some $t_m \in]t_r, t_s[$, say $t_m := 0.5(t_r + t_s)$.

(S2) Replace the edge e by two edges $[\mathbf{x}_s^{(i)}, \mathbf{x}_m^{(i)}]$ and $[\mathbf{x}_m^{(i)}, \mathbf{x}_r^{(i)}]$ in the polygon $P^{(i)}$.

(S3) Consider the face S_j which is incident upon S_i and which contains $\psi_i(\mathbf{x}_m^{(i)})$. In order that the condition in (3.53) is fulfilled, search for $\tau \in \mathbf{R}$ such that

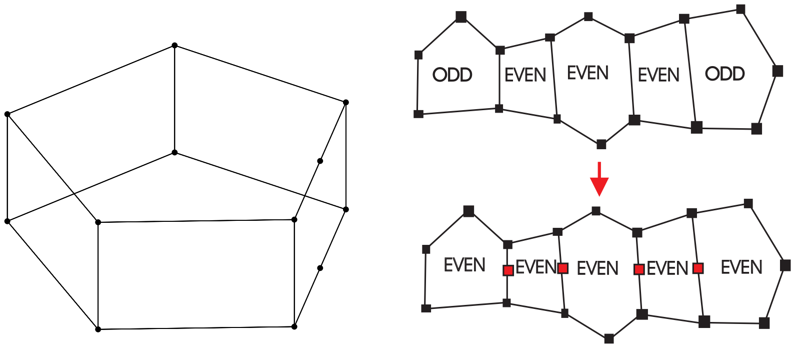
$$\psi_i(\mathbf{x}_m^{(i)}) = \psi_j[\kappa^{(j)}(\tau)].$$

(S4) Insert the point $\mathbf{x}_m^{(j)} := \kappa^{(j)}(\tau)$ in the polygonal approximation $P^{(j)}$ of $\mathcal{D}_j$ as we have done in (S2).

3.12.3 Even polygonal approximation

In this section, we suppose that we have polygonal approximations $\{P^{(i)}\}_{i\in\Lambda}$ of the 2D domains $\{\mathcal{D}_i\}_{i\in\Lambda}$. Let us assume further that the set of polygonal approximations $\{P^{(i)}\}_{i\in\Lambda}$ contains some odd polygons. Since having even polygons is a prerequisite for quadrangulation, we will show how to make all polygonal approximations even.

Definition 12 Suppose that we have polygonal approximations $\{P^{(i)}\}_{i\in\Lambda}$. We will say that the face S_i is odd (resp. even) if the corresponding polygon $P^{(i)}$ has an odd (resp. even) number of boundary vertices.



(a) (b)

Figure 3.26: (a)We have to insert more nodes to make every face even (b)Process of making two odd faces even

Before introducing an important theorem, the following observation is worth mentioning. Suppose that we have two odd faces S_i and S_j that are separated by k faces $S_{r(1)}$, $S_{r(2)}$, $\cdots$, $S_{r(k)}$ which are all even. Adding one extra vertex into the common curve of each pair $(S_i, S_{r(1)})$, $(S_{r(1)}, S_{r(2)})$, $(S_{r(2)}, S_{r(3)})$, $\cdots$, $(S_{r(k-1)}, S_{r(k)})$, $(S_{r(k)}, S_j)$ will make the parities of S_i and S_j even (see Fig. 3.26(b)). The parities of the intermediate faces $S_{r(i)}$ are kept unchanged (i.e. even). That is due to the simple fact that adding a vertex in a face will toggle its parity.

Theorem 7 Let us assume that the set of faces $\{S_i\}_{i\in\Lambda}$ defines a closed surface. Suppose further that we have polygonal approximations $\{P^{(i)}\}_{i\in\Lambda}$ of $\{S_i\}_{i\in\Lambda}$. Denote by ν the number of odd faces S_i. Then, ν must be an even number.

Proof

We will prove this theorem by contradiction. Suppose that the number ν of odd faces S_i were odd. Without loss of generality we may assume that $\nu = 1$ and we denote by S_σ the odd face. Because of the 2-ear theorem and the cut insertion techniques that we discussed in former sections, it is possible to generate a triangulation $\mu^{(i)}$ for each polygon $P^{(i)}$ without using additional boundary nodes. Let $M^{(i)}$ be the triangulation which is the image of $\mu^{(i)}$ by ψ_i. Let $E^{(i)}$ be the set of boundary edges of $M^{(i)}$, i.e. each edge if $E^{(i)}$ has only one incident triangle from $M^{(i)}$. Introduce $p^{(i)} := \mathrm{card}(E^{(i)})$. Since, every face S_i is even for $i \neq \sigma$, the number $p^{(i)}$ must be even for $i \neq \sigma$.

Consider now two adjacent faces S_i and S_j such that $i \neq \sigma$ and $j \neq \sigma$. On account of the condition in (3.53), some boundary edges of $M^{(i)}$ and $M^{(j)}$ must coincide. Let q be the number of common boundary edges of $M^{(i)}$ and $M^{(j)}$. Let us consider the mesh Q which is obtained by merging the meshes $M^{(i)}$ and $M^{(j)}$.

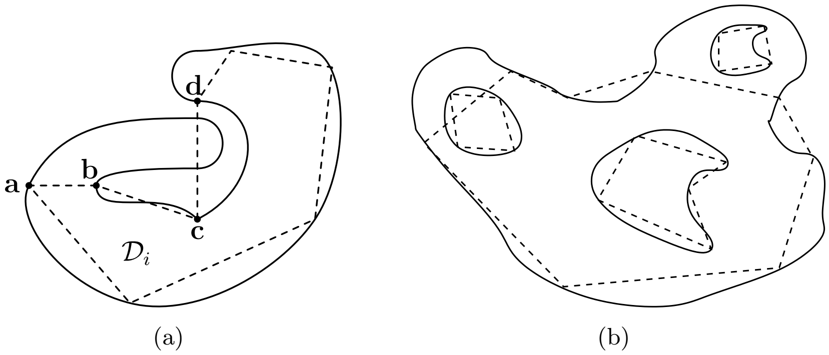


Figure 3.27: Discretization artifacts: (a)The edge $[\mathbf{c}, \mathbf{d}]$ interferes with the curve portion which is approximated by $[\mathbf{a}, \mathbf{b}]$ (b)The polygonal approximations do not form the boundary of a multiply connected polygon

The number of boundary edges of Q is therefore

$$p^{(i)} + p^{(j)} - 2 \cdot q. \tag{3.54}$$

That means that the mesh Q has an even number of boundary edges.

By repeating the former technique in which we merge Q with every S_k with $k \neq \sigma$, $k \neq i$, $k \neq j$, we obtain a mesh R which has an even number of boundary edges. That is in contradiction with the fact that S_σ is an odd face because the boundaries of R and S_σ coincide.

□

For a pair of odd faces (S_i, S_j), we can find a path p from the adjacency graph $\mathcal{G}$ such that p connects the graph nodes n_i and n_j which correspond to the faces S_i and S_j. Suppose that the path p is a sequence of N graph-nodes $n_{r(s)}$ $s = 1, ..., N$ such that $n_{r(1)} = n_i$ and $n_{r(N)} = n_j$. In order to make S_i and S_j even, we insert a node between each pair $(S_{r(k)}, S_{r(k+1)})$ for each $k = 1, ..., N - 1$. The above theorem ensures that the odd faces can be converted into even faces two by two.

3.13 Metric aspect of the polygonal model

This section will be occupied by the description of the actual problems that one encounters in polygonal approximations if we want to use them as input in a quadrangulation process.

3.13.1 Discretization artifact

Consider a multiply connected domain $\mathcal{D}$ having curved boundaries. Let us denote by $\mathbf{B}_{\text{out}}$ and $\mathbf{B}_{\text{in}}^{j}$ its exterior and interior boundaries respectively. By picking some vertices $\mathbf{x}_{\text{out}}^k$ and $\mathbf{x}_{\text{in}}^{j,k}$ from $\mathbf{B}_{\text{out}}$ and $\mathbf{B}_{\text{in}}^j$, some polygons P_{out} and P_{in}^j are generated. We want to mention that P_{out} and P_{in}^j may not form the boundary of a multiply connected polygonal region as illustrated in Fig. 3.27(b). The following reasons may cause such a problem:

- A vertex of some P_{in}^j may lie outside the exterior polygon P_{out}.
- Some edges e and f which belong respectively to P_{in}^k and P_{in}^l with $k \neq l$ may intersect.
- A tiny internal polygon P_{in}^j may even be located completely outside the external polygon P_{out}.

On the other hand, it is possible that P_{out} and P_{in}^j form acceptable polygons but the curve portion between the endpoints of some edge $[\mathbf{a},\mathbf{b}]$ of P_{out} or P_{in}^j intersects another edge $[\mathbf{c},\mathbf{d}] \neq [\mathbf{a},\mathbf{b}]$ as in Fig. 3.27(a).

Such discretization artifacts often occur when some of the polygons P_{out} and P_{in}^j are too coarse at some positions. The only remedy is to refine those polygons by inserting more nodes from $\mathbf{B}_{\text{out}}$ and $\mathbf{B}_{\text{in}}^j$. The main difficulty in treating discretization artifacts is the detection of such problems. One needs to know if a point lies outside a polygon which might be nonconvex. The detection may also involve polygon-polygon intersections. The solution to those problems is very long and difficult especially if one wants to achieve efficiency. For further discussion about point locations techniques or polygon intersections, see [90, 29, 75] and the references there.

The way we avoid discretization artifacts is the use of ε-length closeness which we introduce now.

Definition 13 Let P be a polygonal approximation of a domain $\mathcal{D}$ having curved boundaries. We say that P has an ε-length closeness when the difference between the length of each straight edge of P and the length of the corresponding curved boundary portion from $\mathcal{D}$ is smaller than some prescribed parameter $\varepsilon > 0$.

Throughout this chapter, we suppose that in any polygonal approximation P, the value of ε is chosen sufficiently small so that the ε-length closeness guarantees that problems related to discretization artifacts do not occur.

3.13.2 Edge quality

The number and positions of vertices of a polygon P may affect the shapes of the quadrilaterals which are members of a quadrangulation of P. In this section, we will see the notion of edges of bad quality which are very undesired in practice.

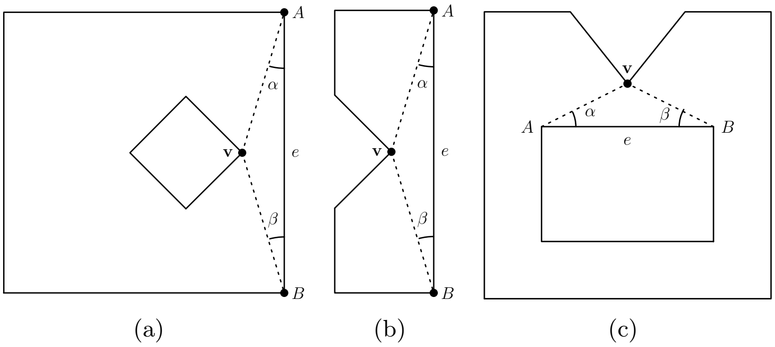


Figure 3.28: Edges of bad quality

Definition 14 Consider a polygon P which can be multiply connected. Let $(e, \mathbf{v})$ be a pair of an edge $e = [A, B]$ and a reflex vertex $\mathbf{v}$ of P. We will say that the pair $(e, \mathbf{v})$ is of bad quality if:

- None of the vertices A and B is located in the wedge $\mathcal{W}$ of $\mathbf{v}$.
- The intersection $\mathcal{W} \cap e$ is not empty.
- The angle $\delta := \max\{\alpha, \beta\}$ is smaller than some user defined parameter $\delta_0 > 0$, where α and β are the angles (Fig. 3.27) made respectively by $(e, A, \mathbf{v})$ and $(e, B, \mathbf{v})$.

In Fig. 3.28, we see three kinds of vertices of bad quality where the edge e may be on the external or the internal boundary of the polygon P. In practice, we use only a pair $(e, \mathbf{v})$ where $\mathbf{v}$ is an extreme vertex as introduced in section 3.7.1.

The quadrangulation of a polygon having edges of bad quality generally contains badly shaped quadrilaterals which are long and thin. If we have an edge e of bad quality on a polygonal approximation $P^{(i)}$ of $\mathcal{D}_i$ which maps to S_i, we have the option of inserting a new node $\mathbf{w}$ inside it. It is worth mentioning that inserting a new node $\mathbf{w}$ in e will affect the boundary approximation of a neighboring trimmed surface S_j which is incident upon e. Consider the preimage f by ψ_j of $\psi_i(e)$:

$$\psi_j(f) = \psi_i(e)$$

If the edge f is not of bad quality in $P^{(j)}$, then inserting the new node $\mathbf{w}$ on e may deteriorate the quality of S_j. Therefore, one can take a convention that a node $\mathbf{w}$ is inserted only if both e and f are of bad quality with respect to the polygonal approximations $P^{(i)}$ and $P^{(j)}$ respectively.

3.13.3 Polygonal approximation

Now that we have seen all components that are required in polygonal approximations, we can describe the generation of $\{P^{(i)}\}_{i\in\Lambda}$ from $\{\mathcal{D}_i\}_{i\in\Lambda}$. The discretizations are performed as follows.

1. For every $i \in \Lambda$, initialize the nodes of $P^{(i)}$ to consist only of the segment separators of the domain $\mathcal{D}_i$. Choose some $\varepsilon > 0$ and determine the adjacency graph $\mathcal{G}$ (see Definition 10).
2. For every $i \in \Lambda$ (start from those $P^{(i)}$ having many boundary edges), refine the edges of $P^{(i)}$ repeatedly until ε-length closeness is attained such that each $P^{(i)}$ does not have any discretization artifacts. In this step, we have to make sure that condition (3.53) is met by using the edge splitting process introduced in section 3.12.2.
3. This step is optional. Insert additional nodes at edges of bad quality.
4. Convert the odd polygons among $\{P^{(i)}\}_{i\in\Lambda}$ into even ones by using the adjacency graph $\mathcal{G}$ as described in section 3.12.3.

3.14 Postprocessing

Since we know how to obtain even polygonal approximations $\{P^{(i)}\}_{i\in\Lambda}$, we can apply a quadrangulation process to each of them. We will describe here the tasks that have to be performed if one wants to deduce decompositions of the domains $\mathcal{D}_i$ from quadrangulations. Additionally, we mention the problems that arise by using that deduction method. Further problems about diffeomorphism from the unit square are described.

3.14.1 Boundary replacement

Suppose that we have a quadrangulation $\mathcal{Q}$ of the polygonal approximation $P^{(i)}$:

$$P^{(i)} = \bigcup_k q_{k,i}$$

where either the nodes of $q_{k,i}$ are taken from the vertices of $P^{(i)}$ or they are located strictly inside $P^{(i)}$. Let us describe how to generate the splitting into four-sided subregions of the domain $\mathcal{D}_i$ which has curved boundaries:

$$\mathcal{D}_i = \bigcup_k Q_{k,i}.$$

That is, we show how to deduce the region $Q_{k,i}$ having curved sides from the convex quadrilateral $q_{k,i}$ as in Fig.3.29. In order to facilitate the presentation, we will drop the index i when there could be no confusion. For any straight edge $f = [\mathbf{x}_r, \mathbf{x}_s]$ of the quadrilateral q_k, we distinguish two cases. If the edge f is an

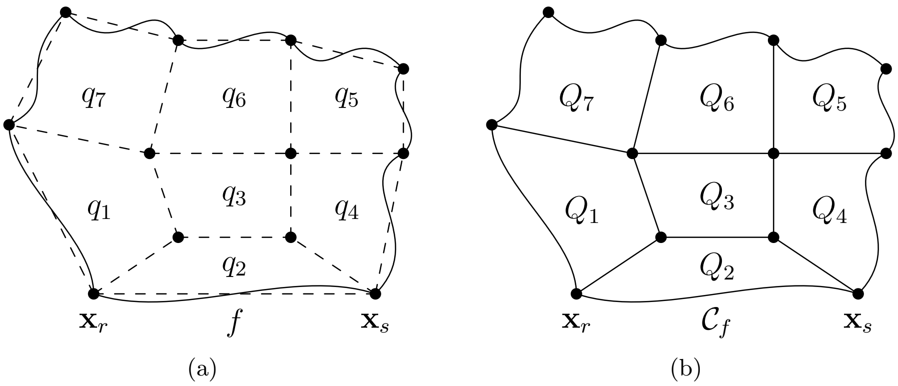


Figure 3.29: Subregions with curved sides from quadrangulation

internal one in the quadrangulation $\mathcal{Q}$, then f becomes an edge of Q_k. Otherwise, we consider the boundary curve $\mathcal{C}$ which contains $\mathbf{x}_r$ and $\mathbf{x}_s$. In the sequel, we will denote by $\mathcal{C}_f$ the portion of the curve $\mathcal{C}$ which is located between the endpoints $\mathbf{x}_r$ and $\mathbf{x}_s$ of f. That is, if $\mathcal{C}$ is defined as the image of a parametric curve κ and $\mathbf{x}_r = \kappa(t_r)$, $\mathbf{x}_s = \kappa(t_s)$ then $\mathcal{C}_f = \kappa([t_r, t_s])$ (we suppose that the edge f follows the orientation of the curve $\mathcal{C}$). Then, the edge of Q_k which corresponds to f is the curve $\mathcal{C}_f$. In practice, not only we store the coordinates of the vertices $\mathbf{x}_l$ of the polygon $P^{(i)}$, but also the parameters t_l at which those vertices are traversed i.e. $\mathbf{x}_l = \kappa(t_l)$.

3.14.2 Boundary interference

In this section, we would like to describe the problem of boundary interference and its possible remedies. The above process of replacing a boundary edge f by a portion of curve $\mathcal{C}_f$ can cause a problem. More accurately, it may happen that the boundary curve $\mathcal{C}_f$ intersects an (internal) edge e other than f if the polygonal approximation (see Fig. 3.30(a)) is not fine enough. It is even possible that a node ω of the quadrangulation $\mathcal{Q}$ becomes exterior to the curved boundaries after replacing the edge f by C_f as in Fig. 3.31(a). This situation happens typically for trimmed surfaces having holes or for boundary curves which are highly nonconvex.

The most obvious approach of removing such boundary interferences is to consider a finer polygonal approximation of the trimmed surface $\mathcal{D}$. Then, one repeats the whole process to the resulting finer polygon. Since that approach could be inefficient when dealing with many patches, we would like to propose a method of modifying only a part of the quadrangulation $\mathcal{Q}$ as follows.

1. Generate two points P_1 and P_2 on the curve $\mathcal{C}_f$ (Figs. 3.30(b), 3.31(b)) which is bounded by $\mathbf{x}_r$ and $\mathbf{x}_s$.

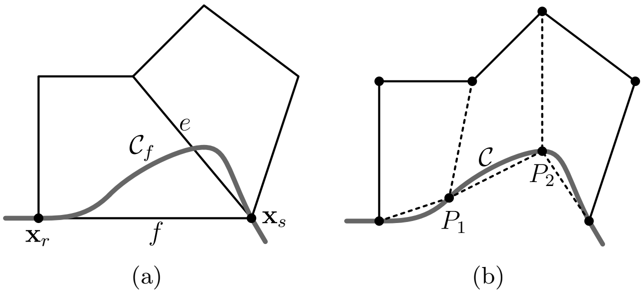


Figure 3.30: (a)Boundary curve $\mathcal{C}$ interferes with internal edge e (b)Requadrangulation

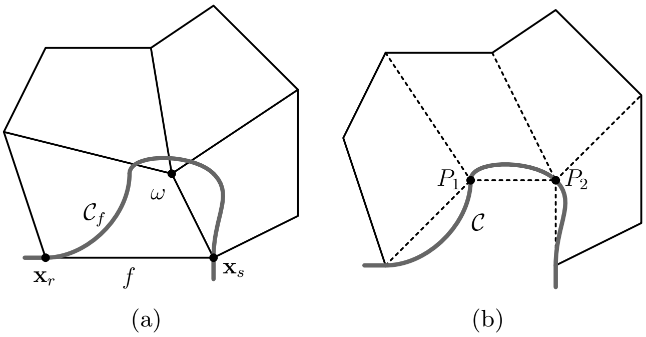


Figure 3.31: (a)The vertex P of the quadrangulation resides outside the curved boundary $\mathcal{C}$. (b)Requadrangulation

2. Find the set of quadrilaterals $\mathcal{Z}$ which have nonempty intersection with $\mathcal{C}_f \setminus \{\mathbf{x}_r, \mathbf{x}_s\}$.

3. Find the polygon P_{loc} which consists of the boundary of $\mathcal{Z}$ and the points P_1 and P_2 as follows. Consider the polygonal region $\tilde{P}_{\text{loc}}$ which is formed by the boundary of the union of the quadrilaterals of $\mathcal{Z}$. We define the polygon P_{loc} to have the edges of $\tilde{P}_{\text{loc}}$ except that we replace the edge $[\mathbf{x}_r, \mathbf{x}_s]$ by three edges $[\mathbf{x}_r, P_1]$, $[P_1, P_2]$ and $[P_2, \mathbf{x}_s]$.

4. Quadrilate P_{loc} to have the quadrangulation $\mathcal{Q}_{\text{loc}}$.

5. Replace the quadrilaterals of $\mathcal{Z}$ in the large quadrangulation $\mathcal{Q}$ by $\mathcal{Q}_{\text{loc}}$.

Since we keep the quadrangulation which is far from the edge f intact, we believe that this method is more efficient than repeating the whole quadrangulation on a finer polygon. The reason for introducing two new vertices (and not just one) is that we must have an even number of vertices in the local polygon P_{loc}.

Remark 8 It is obvious that the above process of introducing P_1 and P_2 should not be done on one single face S_i. In order to have splitting without hanging nodes, we have to modify also the surface S_j which is incident upon S_i and which contains the image of $\mathcal{C}_f$ by ψ_i. That means, we consider the boundary curve $\mathcal{C}'_g$ of $\mathcal{D}_j$ and we insert two nodes Q_1 and Q_2 on $\mathcal{C}'_g$ such that

$$\psi_i(P_s) = \psi_j(Q_s) \qquad s = 1, 2. \tag{3.55}$$

Afterwards, we repair the quadrangulation of $P^{(j)}$ in the neighborhod of $\mathcal{C}'_g$ just as we have done it to $\mathcal{C}_f$.

The determination of the set of quadrilaterals $\mathcal{Z}$ of the second step is done as follows. We are interested in finding the edges of $\mathcal{Q}$ which intersect the curve $\mathcal{C}_f$. We do not need to test intersections with all edges of the quadrangulation $\mathcal{Q}$. Since we have made an assumption about the polygonal approximation that the curve $\mathcal{C}_f$ cannot intersect a boundary edge other than f, the curve portion $\mathcal{C}_f$ could only have an intersection with an edge of $\mathcal{Q}$ if $\mathcal{C}_f$ enters through quadrilaterals. As a consequence, the process of finding $\mathcal{Z}$ is summarized in the following algorithm.

```
step 0  :  Define $f^{(0)} := f$ and $t^{(0)}$ is the quadrilateral which is incident
           upon $t^{(0)}$. Initialize $\mathcal{Z} := \{t^{(0)}\}$ and set $k = 0$.
step 1  :  Take a discretization $\mathcal{C}_f^{\text{disc}}$ of the curve portion $\mathcal{C}_f$.
step 2  :  Find the three edges $e^{(1)}$, $e^{(2)}$, $e^{(3)}$, of $t^{(k)}$ such that $e^{(s)} \neq$
           $f^{(k)}$ $s = 1, 2, 3$.
step 3  :  For every $e^{(s)}$ with $s = 1, 2, 3$:
```

- If $\mathcal{C}_f^{\text{disc}} \cap e^{(s)} \neq \emptyset$, define $f^{(k+1)} := e^{(s)}$ and let $t^{(k+1)}$ be the quadrilateral which is incident upon $f^{(k+1)}$ and which is different from $t^{(k)}$ and go to step 4.
- Else terminate.

step 4 : $\mathcal{Z} := \mathcal{Z} \cup \{t^{(k+1)}\}$ set $k := k + 1$ go to step 1.

Remark 9 The previous algorithm about finding $\mathcal{Z}$ can be improved in various ways. First, one can consider the bounding box of the curve $\mathcal{C}_f$ in order to accelerate the intersection tests. Second, one can take a coarse discretization $\mathcal{C}_f^{\text{disc}}$ of $\mathcal{C}_f$ initially. We stop if it is enough to find $\mathcal{Z}$. Otherwise, one refines the approximation $\mathcal{C}_f^{\text{disc}}$. We can also make an assumption that if the cardinality of $\mathcal{Z}$ exceeds some number m (say $m = 4$), then we give up local repairing. Instead, we refine the whole polygonal approximation $P^{(i)}$ by using a smaller value of ε in the ε-length closeness and we repeat the whole quadrangulation.

3.14.3 Treating G^1 vertices

Definition 15 Consider a parametric curve $\mathcal{C}(t)$ which is supposed to be a composite curve. A point $\mathbf{x} = \mathcal{C}(t_0)$ on the curve is called a G^1 vertex if we have G^1 geometric continuity [68, 39] there and $\mathbf{x}$ does not reside strictly within a line segment. In Fig. 3.32(a) B and D are G^1 vertices while A and C are not.

We do not want to have a four-sided subregion R in which one (or more) of its four vertices is a G^1 vertex (as in Fig. 3.32(b)). This is because of the fact that we cannot find any diffeomorphism φ from the unit square $D = [0, 1] \times [0, 1]$ to such a subregion R where φ transforms the four vertices of D to those of R.

After replacing the boundary straight edges of a quadrangulation by the corresponding curve portion as described in section 3.14.2, there could exist some G^1 vertices which have no emanating internal edge. This section will be occupied by the reparation of such problems. Suppose we have a quadrangulation such that if we replace the edges of the quadrilateral $Q = [A, B, C, D]$ by boundary curves, then we obtain a four-sided region having a G^1 vertex at B. In this section, we want to describe two methods of repairing the quadrangulation so that an edge emanates from B.

The first method consists in refining the regions Q into five four-sided sub-regions as depicted in Fig. 3.33(a). Since that approach could yields too many quadrilaterals, we propose the following method. We take a neighboring convex quadrilateral P of Q and we form the hexagon $H := P \cup Q$. Let us denote by E and F the nodes of P which do not belong to Q as shown in Fig. 3.33(b). Our

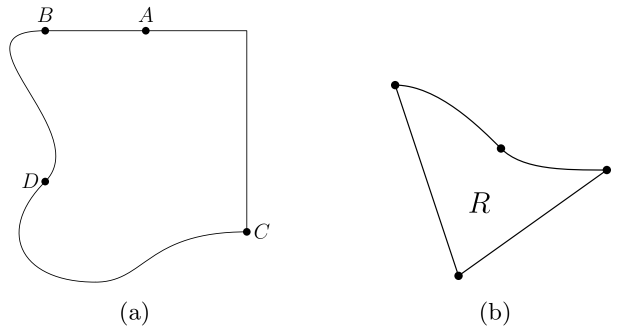


Figure 3.32: (a) B and D are G^1 vertices while A and C are not (b) a four sided domain having a G^1 vertex

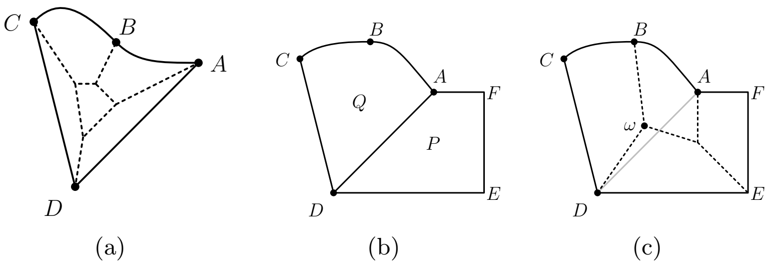


Figure 3.33: Treating a G^1 vertex. An edge should emanate from the G^1 vertex B

second method consists in generating a quadrangulation Q_{loc} of the hexagon H in such a way that an edge emanates from the G^1 vertex B. Note that simply quadrilating the hexagon H may generate a quadrangulation having no edge emanating from B. In order to guarantee that an edge emanates from B, we pick any point ω strictly inside the triangle $[A, B, D]$. Then, we form the quadrilateral $[D, \omega, B, C]$ which must be convex because Q is convex. Now we quadrilate the auxiliary hexagon $\tilde{H} := [F, A, B, \omega, D, E]$ by using Bremner's method which is summarized in Theorem 6:

$$\tilde{H} = \cup_s q_s. \tag{3.56}$$

As a consequence, the quadrangulation Q_{loc} of the hexagon H is given by

$$H = [D, \omega, B, C] \cup (\cup_s q_s). \tag{3.57}$$

Note that in the quadrangulation of (3.57), the edge $[\omega, B]$ emanates from the node B. We need to note that after doing those two processes, we may obtain boundary interference which we have to repair by using the method described in section 3.14.2.

3.14.4 Subdivision for diffeomorphisms

In the former sections, we have shown a method of splitting a multiply connected domain $\mathcal{D}_i$ having curved boundaries into four-sided subdomains $Q_{k,i}$. After the splitting, one can distinguish three types of four-sided domains:

(Tp1) Those which are convex quadrilaterals,

(Tp2) Those which have only one curved side,

(Tp3) Those which have several curved sides.

In any case, the corners of a four-sided domain form a convex quadrilateral.

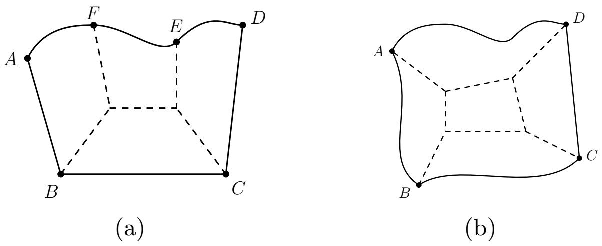


Figure 3.34: Refinement: (a)Inserting two nodes E and F between A and D (b) Every quadrilaterals has at most one curved side

Since the function ψ_i defined on the trimmed surface $\mathcal{D}_i$ (see relation (3.1)) is supposed to be a diffeomorphism, the generation of a diffeomorphism from the unit square to the image $\psi_i(Q_{k,i}) \subset \mathbf{R}^3$ amounts to generating a diffeomorphism onto the four-sided 2D-domain $Q_{k,i}$. In the next chapter, we develop theoretical and practical approaches using transfinite interpolation related to diffeomorphisms for 2D-domains. For a convex quadrilateral, the generation of a diffeomorphism can be done with the help of a Coons patch as stated in Corollary 3 of the following chapter. That means, problems could only arise to four-sided domains having curved sides. Thus, we need to mention now the treatment of four-sided domains of second and third types. In section 4.8 of the following chapter, we introduce a method of generating internal curves for Gordon patches. That method may fail to directly yield a diffeomorphism if the bounding curves are too complicated. In such a case, the remedy that we will propose in section 4.9 of the following chapter is to subdivide the four-sided region in order to simplify the bounding curves. On that account, we would like to show now a method of splitting a four-sided domain so that the bounding curves become less complicated. Let us consider a four-sided domain q which has $[A, B, C, D]$ as corners and which has one curved side $\mathcal{C}$ bounded by A and B. In order to make the curve $\mathcal{C}$ less complicated, we insert two nodes E and F inside it as illustrated in Fig. 3.34(a). Afterwards, we quadrilate the hexagon $[A, B, C, D, E, F]$. As we have done in section 3.14.2, we replace the four-sided domain q by the resulting quadrangulation. For a four-sided domain having several curved edges (of third type), one can use similar techniques by introducing new nodes on the curved sides. Another possibility is to try to split it into a few four-sided domains having single curved side (i.e. of second type) as in Fig. 3.34(b).

3.15 Summarizing algorithm

The former long discussion about splitting 2D-domains $\{\mathcal{D}_i\}_{i\in\Lambda}$ into four-sided subdomains can be summarized in form of an algorithm whose input includes polygonal approximations $\{P^{(i)}\}_{i\in\Lambda}$.

1. Quadrilate each polygon $P^{(i)}$ in oder to have a quadrangulation $\mathcal{Q}_i$ which uses only the vertices of $P^{(i)}$ as boundary nodes:

   $$P^{(i)} = \bigcup_k q_{k,i}.$$

   The quadrangulation approach from section 3.6 which uses no boundary Steiner points fits well that objective.

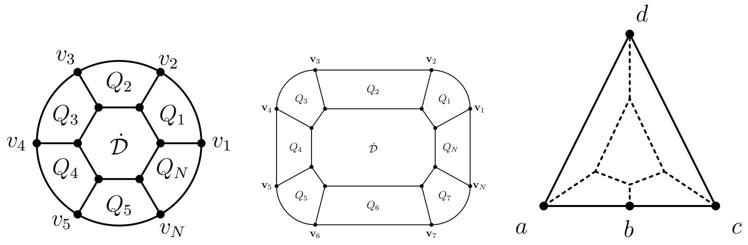
2. Deduce from the quadrangulation $\mathcal{Q}_i$ a decompositions of $\mathcal{D}_i$ into four-sided subregions with curved sides.

   $$\mathcal{D}_i = \bigcup_k Q_{k,i}.$$

3. Repair the decomposition in the neighborhood of the remaining G^1 vertices.
4. Repair boundary interference.
5. Check diffeomorphism from $[0,1]^2$ onto each $Q_{k,i}$ as described in the next chapter.
6. Subdivide each four-sided region $Q_{k,i}$ yielding no diffeomorphisms and go back to the above fifth point.

3.16 Special splittings

There are a few special situations which often occur in practice. Treating these cases in a special way will be much more efficient than applying the general procedure. In this section, we would like to describe a case study of those special splittings.



(a) (b) (c)

Figure 3.35: (a),(b)Special splittings: all vertices are G^1. (c) Quadrilate a triangular-shaped quadrilateral without using additional boundary nodes

3.16.1 All G^1 vertices

The first splitting is applied to a domain $\mathcal{D}$ in which all boundary vertices are G^1-vertices and they form a convex polygon P (see Fig. 3.35(b)). This situation occurs very often for mechanical parts which contain closed smooth curves such as circles. Let us denote those vertices by $\mathbf{v}_1, \cdots, \mathbf{v}_N$. Our method consists in constructing another polygon $\dot{\mathcal{D}}$ which is inside P and which has the same shape as P. For example, one can define the vertices of $\dot{\mathcal{D}}$ by

$$\mathbf{w}_i := \lambda \mathbf{G} + (1-\lambda)\mathbf{v}_i, \tag{3.58}$$

with $\mathbf{G}$ being the center of gravity of P and $\lambda \in]0,1[$. The tessellation of $\mathcal{D}$ is constituted of the four-sided regions Q_1,...,Q_N by introducing the line segments $[\mathbf{v}_i, \mathbf{w}_i]$ and the quadrilaterals resulting from the quadrangulation of the internal polygon $\dot{\mathcal{D}}$ (see Fig. 3.35(b)).

3.16.2 Simple polygons

The second group of special splittings is devised for the quadrangulation of simple polygons. We will consider only simple polygons with at most six vertices and we will distinguish a few cases according to the number of reflex vertices. Let us observe the simple fact that an n-gon has at most $(n-3)$ reflex vertices. Although our aim is to have a small number of quadrilaterals, we do not claim that these are the only (nor the optimal) way of splitting those polygons.

Non-convex quadrilateral

In the following discussion, we describe the way a non-convex quadrilateral $[a,b,c,d]$ can be split into convex ones. We suppose that b is the reflex vertex. The approach consists in generating one internal Steiner point ω and two boundary ones e and f. The splitting is then processed as it is graphically described in

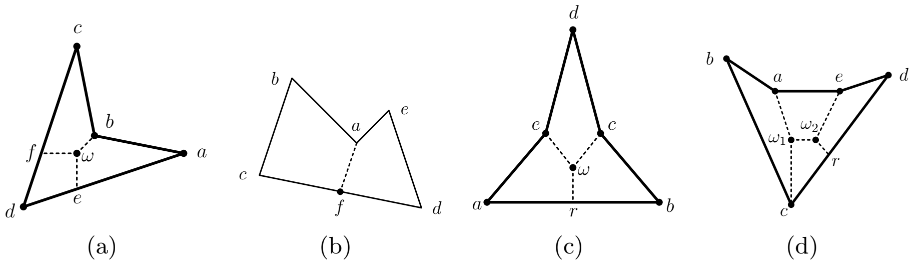


Figure 3.36: Special splittings: (a)Nonconvex quadrilateral, (b) Pentagon with one reflex vertex, (c,d) Pentagon with two reflex vertices.

Fig. 3.36(a). We need only to explain how the coordinates of the Steiner points are chosen. The vertex e and f should lie respectively on the segments $[d, a]$ and $[d, c]$ and the internal node ω is chosen in the triangle $[e, f, b]$. We require further that the point e (resp. f) is visible from the corner c (resp. a). Otherwise, the point w could reside outside the initial quadrilateral.

Pentagon

For the quadrangulation of a pentagon $P := [a, b, c, d, e]$, we are going to describe three cases according to the number and the positions of the reflex vertices.

Suppose there is only one reflex vertex which we suppose to be a. Let us observe that intersection of the segment $[c, d]$ and the wedge of a is never empty. We need therefore to take one Steiner point f from that region and the quadrangulation results by inserting one cut $[a, f]$ as depicted in Fig. 3.36(b). This ensures that the two resulting quadrilaterals are convex.

If we have two reflex vertices, then we have to distinguish two cases according to whether those reflex vertices are separated by one non-reflex vertex or they are consecutive. Observe that if they are separated by two non-reflex vertices, then following the pentagon in the opposite direction (clockwise) reduces it to the former case.

Let us first consider the first case in which e suppose without loss of generality that the reflex vertices are c and e as in Fig. 3.36(c). We need to insert one boundary Steiner point r which is chosen so that it resides on the segment $[a, b]$ and that it is visible from the vertex d. Then, an internal Steiner point ω is selected inside the triangle $[r, c, e]$. The splitting is then done as explained in Fig. 3.36(c).

Suppose now that the reflex vertices are consecutive. Further, let us assume that they are a and e (Fig. 3.36(d)). We are now explaining how an internal Steiner point ω_1 is chosen. In order to ensure convexity, we choose ω_1 inside $P \cap (ac)^-$. The new pentagon $[a, \omega_1, c, d, e]$ has two reflex vertices (namely ω_1 and e) which

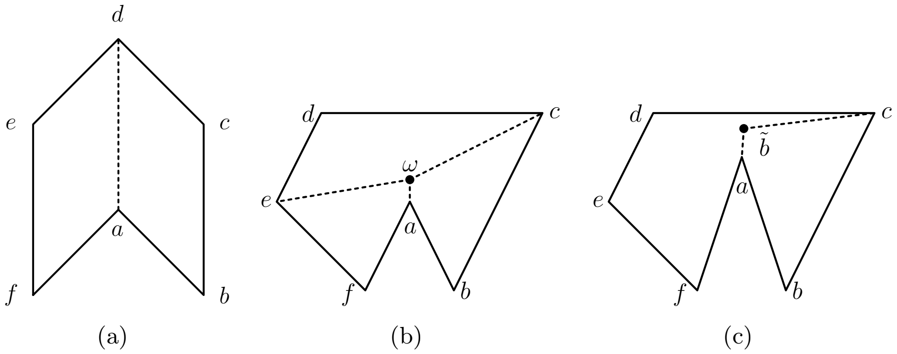


Figure 3.37: Hexagon: one reflex vertex

are separated by one reflex vertex. Therefore, we can apply the former case in order to determine the remaining Steiner points ω_2 and r.

Hexagon

The quadrangulation of a hexagon $P = [a, b, c, d, e, f]$ that we present below has two remarkable properties [16]. First, we do not introduce any new boundary node. Second, all resulting quadrilaterals are strictly convex.

We distinguish three cases according to the number of reflex vertices of the hexagon P. Those cases are listed below and they can be seen graphically in Fig. 3.37 through Fig. 3.39.

Case 1: one reflex vertex

Without loss of generality, we suppose that the reflex vertex is a. We need now to consider a few subcases.

Case 1.1 : If the vertex d belongs to the wedge $\mathcal{W}(a)$, we simply need to insert a cut $[a, d]$ to have the quadrangulation as illustrated in Fig. 3.37(a).

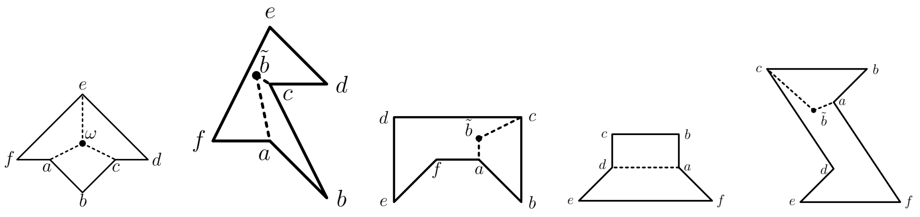
Case 1.2 : If $d \notin \mathcal{W}(a)$ and c sees e, we generate a Steiner point $\omega \in [a, c, e] \cap \ker(P)$ and we quadrilate P as described in Fig. 3.37(b).

Case 1.3 : If $d \notin \mathcal{W}(a)$ and c does not see e, we generate a point $\tilde{b} \in \mathcal{W}(a) \cap \mathcal{W}(c) \cap (ac)^-$ and we apply Case 1.2 to the new hexagon $\tilde{P} = [a, \tilde{b}, c, d, e, f]$.

Case 2: two reflex vertices

Case 2.1: Suppose that the two reflex vertices are separated by a non-reflex one. We assume that the reflex vertices are a and c. If a sees e and c sees e, then we pick a $\omega \in [a, c, e] \cap \ker(P)$ and perform the splitting in Fig. 3.38(a). Otherwise, we search some $\tilde{b}$ in $\mathcal{W}(a) \cap \mathcal{W}(c) \cap (ac)^-$ as in Fig. 3.38(b) and we apply Case 1 to the new hexagon $\tilde{P} = [a, \tilde{b}, c, d, e, f]$.

Case 2.2: If the two reflex vertices are consecutive as in Fig. 3.38(c), we find $\tilde{b}$



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

Figure 3.38: Hexagon: two reflex vertices

in $\ker(Q) \cap (ab)^-$ where Q denotes the quadrilateral $[a, c, d, f]$. Then, apply Case 2.1 to the new hexagon $\tilde{P} = [a, \tilde{b}, c, d, e, f]$.

Case 2.3: Suppose that the two reflex vertices a and d are located between two non-reflex vertices. Test if a sees d and if the two quadrilaterals resulting from inserting the cut $[a, d]$ are convex as in Fig. 3.38(d). If so, then insert the internal edge $[a, d]$. Otherwise, proceed as in case 2.2 (see Fig. 3.38(e)).

Case 3: three reflex vertices

Case 3.1 : Suppose that we have alternation of reflex and non-reflex vertices. If the vertices a, c and e are mutually visible, then find $\omega \in [a, c, e] \cap \ker(P)$ and split as in Fig. 3.39(a). Otherwise, find $\tilde{d}$ such that $[d, e, \tilde{d}, c]$ is convex and c is non-reflex in the new hexagon $\tilde{P} = [a, b, c, \tilde{d}, e, f]$ (Fig. 3.39(b)). Then, apply Case 2 to $\tilde{P}$.

Case 3.2 : Suppose that two of the reflex vertices are consecutive and that they are a and b. If e sees a or e sees b, then apply Case 3.1 to the new hexagon $\tilde{P} = [a, b, c, d, e, \tilde{f}]$ where $\tilde{f}$ is chosen in $(ae)^+ \cap (bd)^- \cap \ker(P)$ as in Fig. 3.39(c). Otherwise, choose $\tilde{e}$ such that $[d, e, f, \tilde{e}]$ is convex as in Fig. 3.39(d) and apply case 3.1 to the new hexagon $\tilde{P} = [a, b, c, d, \tilde{e}, f]$.

Case 3.3 : Suppose that the three reflex vertices are consecutive. Choose $\tilde{f}$ such that $[a, \tilde{f}, e, f]$ is convex and c sees e in the new hexagon $\tilde{P} = [a, b, c, d, e, \tilde{f}]$ (Fig. 3.39(e)). Then, apply Case 3.1 to the new hexagon $\tilde{P}$.

Coordinates of the Steiner points:

The real difficulty in realizing the above simple quadrangulation in practice is the determination of the exact coordinates of the Steiner points ω, $\tilde{b}$, $\tilde{d}$, $\tilde{e}$, $\tilde{f}$. Let us observe that the determination of the Steiner points can be done by repeated applications of convex polygon intersections. That is due to the fact that all the involved regions such as $ker(P)$, wedge, triangle, half-plane, can be derived from intersections of convex polygons. In practice, it is advisable to determine first a bounding box B of the hexagon and to represent a half-plane as a large rectangle which does not interfere with points outside B. Therefore, we would like to present below [29, 90] a simple intersection algorithm of two convex polygons P and Q. We will classify the intersection points of P and Q in two categories. An

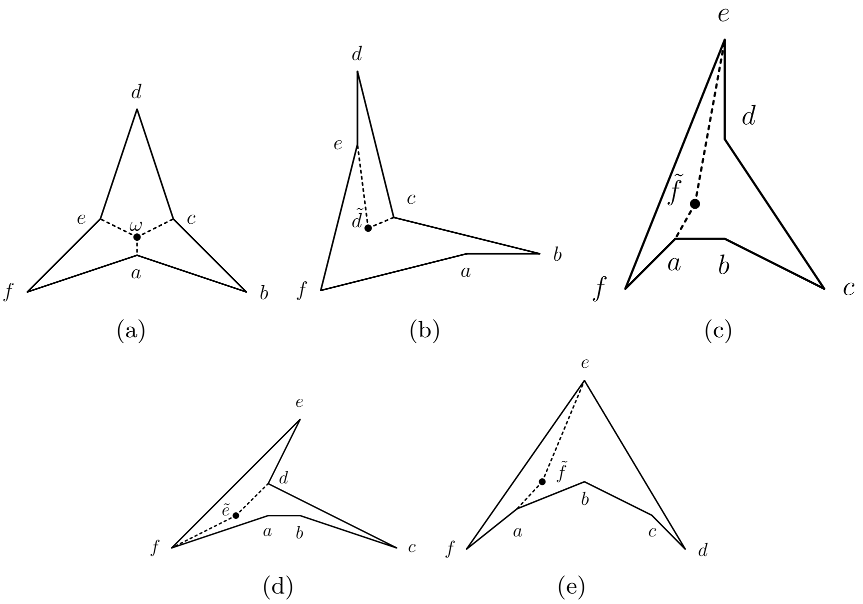


Figure 3.39: Hexagon: three reflex vertices

intersection of first kind is $I \in P \cap Q$ such that the following (in counterclockwise orientation) edges of P and Q coincide. An intersection which is not of first kind is of second kind. Observe that an intersection of first kind is always a corner of P or Q. As an illustration, in Fig. 3.40 the point I_2 is an intersection of first kind and the other intersection points are of second kind.

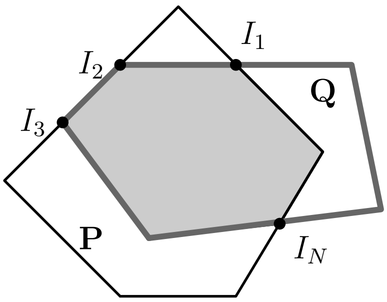


Figure 3.40: Intersection of two convex polygons **P** and **Q**: I_2 is an intersection of first kind.

Algorithm: Convex polygon intersection

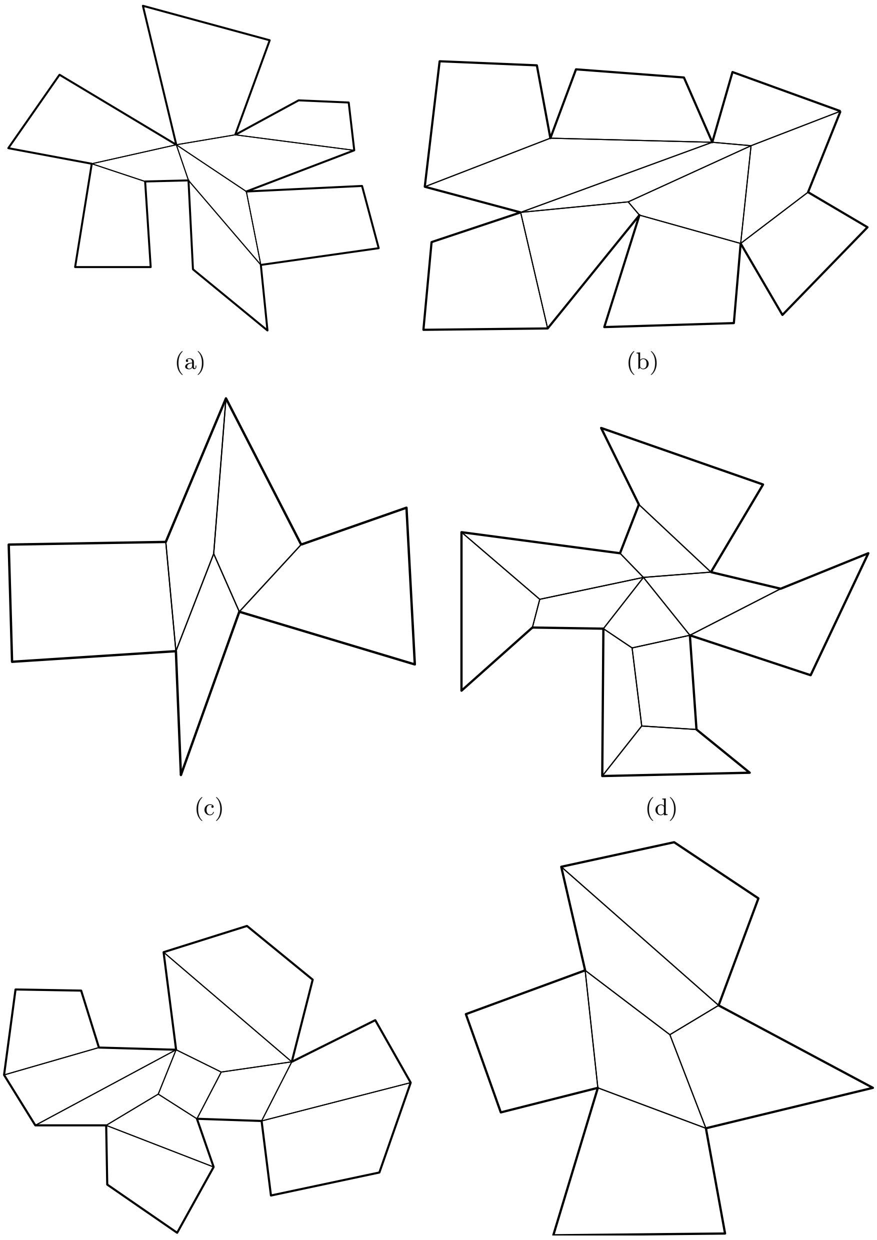
```
step 1 : Find all the intersection points $I_1, \cdots, I_N$ of the convex polygons $P$ and $Q$. Determine also their types (first or second).
step 2 : Define $\mathcal{L}$ to be the list of vertices of $P \cap Q$; initialize $\mathcal{L}[0] := I_1$ and set $k = 0$.
step 3 : So as to find $\mathcal{L}[k+1]$, distinguish three cases:
    case 1 : If $\mathcal{L}[k]$ is an intersection point of first kind $I_s$, the next entry of $\mathcal{L}$ is the following intersection point: $\mathcal{L}[k+1] := I_{s+1}$.
    case 2 : If $\mathcal{L}[k]$ is an intersection point of second kind $I_s$, define $B \in \{P, Q\}$ as the internal polygon after $I_s$ in counterclockwise orientation and find its next vertex $B[r]$. By following the polygon $B$ from the point $I_s$ counterclockwise, determine which is closer $B[r]$ or $I_{s+1}$. Set $\mathcal{L}[k+1] := B[r]$ or $\mathcal{L}[k+1] := I_{s+1}$ whichever is closer.
    case 2 : If $\mathcal{L}[k]$ is a polygon vertex: say $\mathcal{L}[k] = B[r]$ where $B = P$ or $B = Q$. Find the next intersection point $I_q$ by following $B$ counterclockwise. And choose again from $B[r+1]$ and $I_q$ as in the previous point.
step 4 : If all the intersection points have been traversed then terminate. Otherwise, increment $k$ and go back to step 3.
```

3.17 Numerical results

In this section, we present results of the formerly discussed methods. First, we will show results about splitting simply and multiply connected polygons. Then, we will see results from real CAD data. We investigate five sets of polygons whose quadrangulations are to be found in Fig. 3.41 through Fig. 3.45. We display there different polygons with various complexities . Some polygons are highly nonconvex while others are only slightly nonconvex. In the first three sets, we display only results about simply connected polygons. The next two sets are devoted for multiply connected polygons. Please note that the boundaries of the multiply connected polygons are drawn with bold lines in order not to confuse internal four-sided boundaries and quadrilaterals.

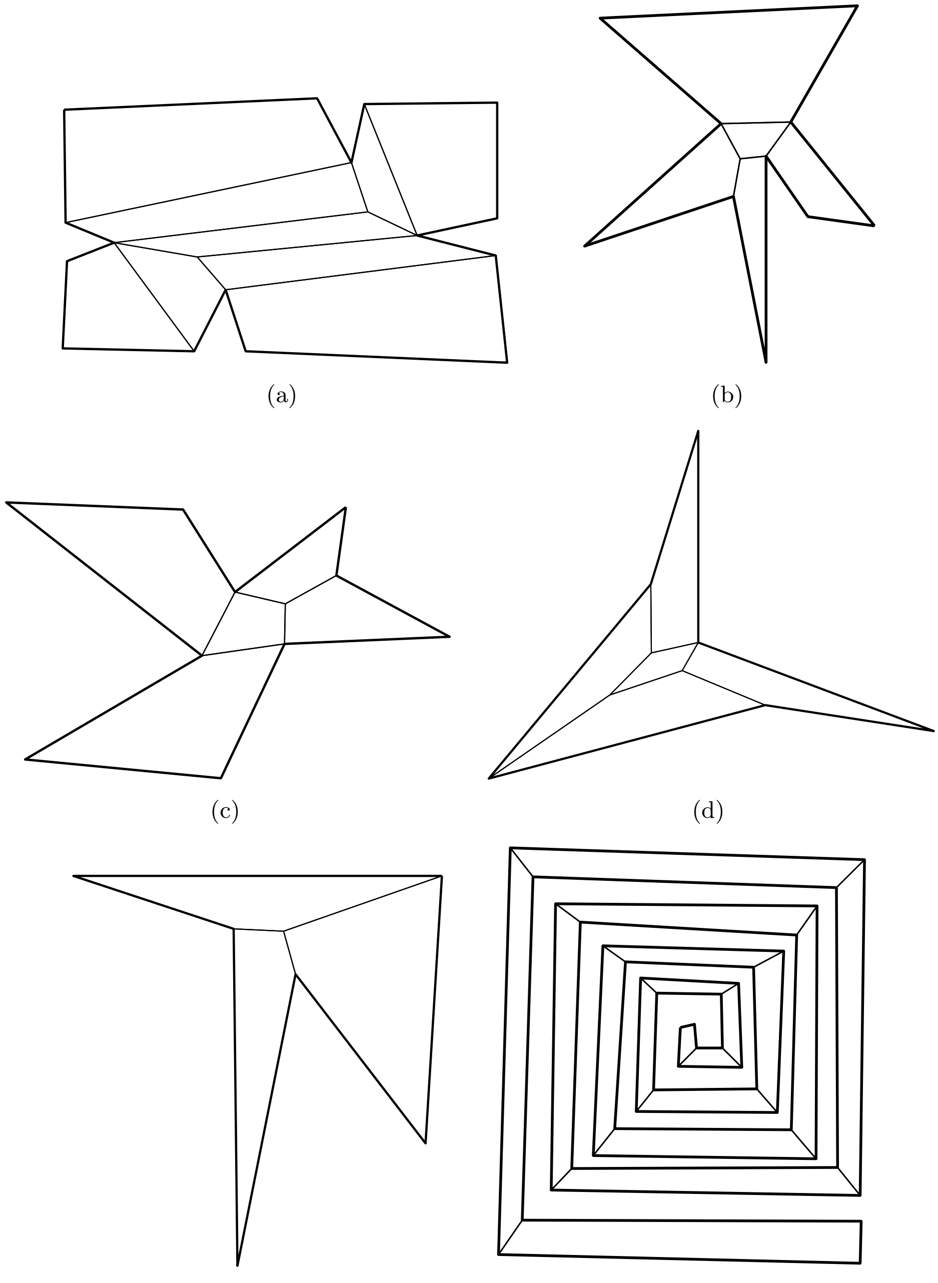
Now we would like to present some numerical results of decomposing CAD objects which are taken from IGES files. We consider eight objects which are displayed in Fig. 3.46 and Fig. 3.47 showing the initial CAD-surfaces and the final four-sided tessellations which have been obtained by using the afore mentioned methods.

We want particularly to present the runtimes of the approaches. The following tests have been done with an Intel Pentium 4 processor 2,66 GHz running Win-



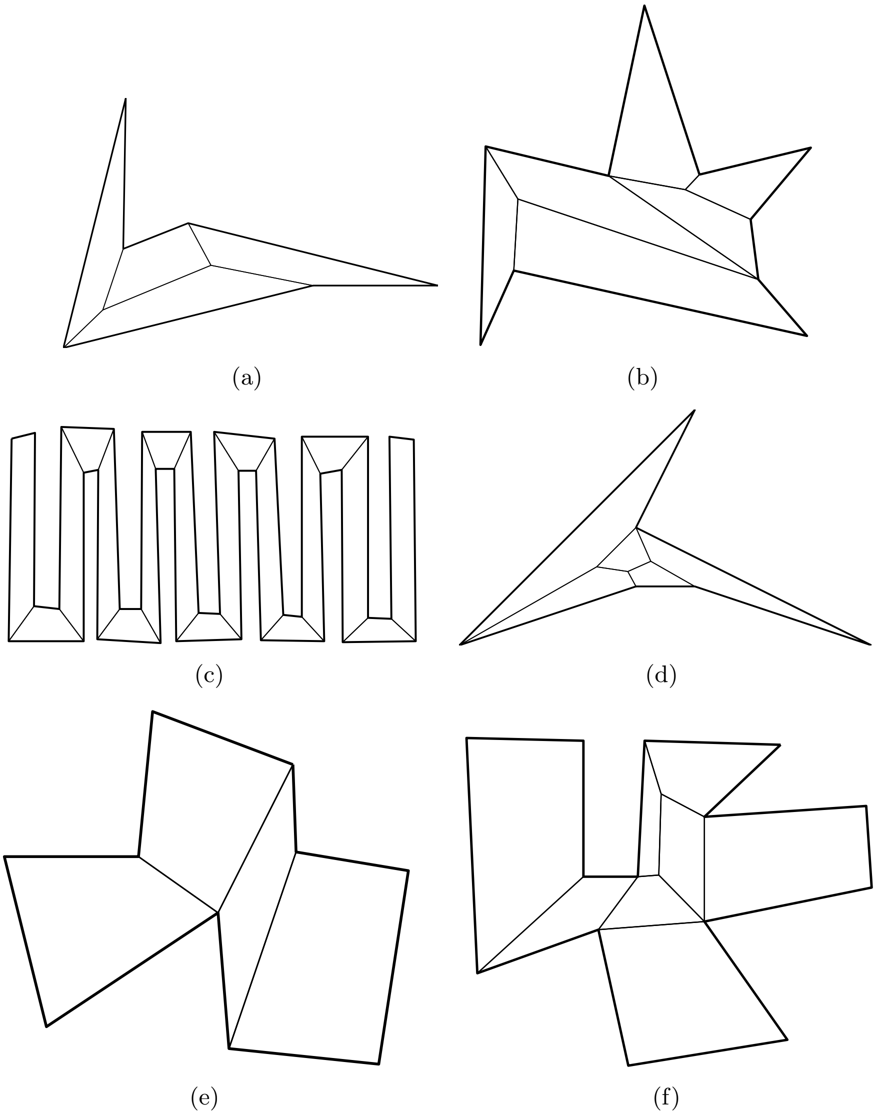
(a) (b) (c) (d) (e) (f)

Figure 3.41: First set of results.



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

Figure 3.42: Second set of results.

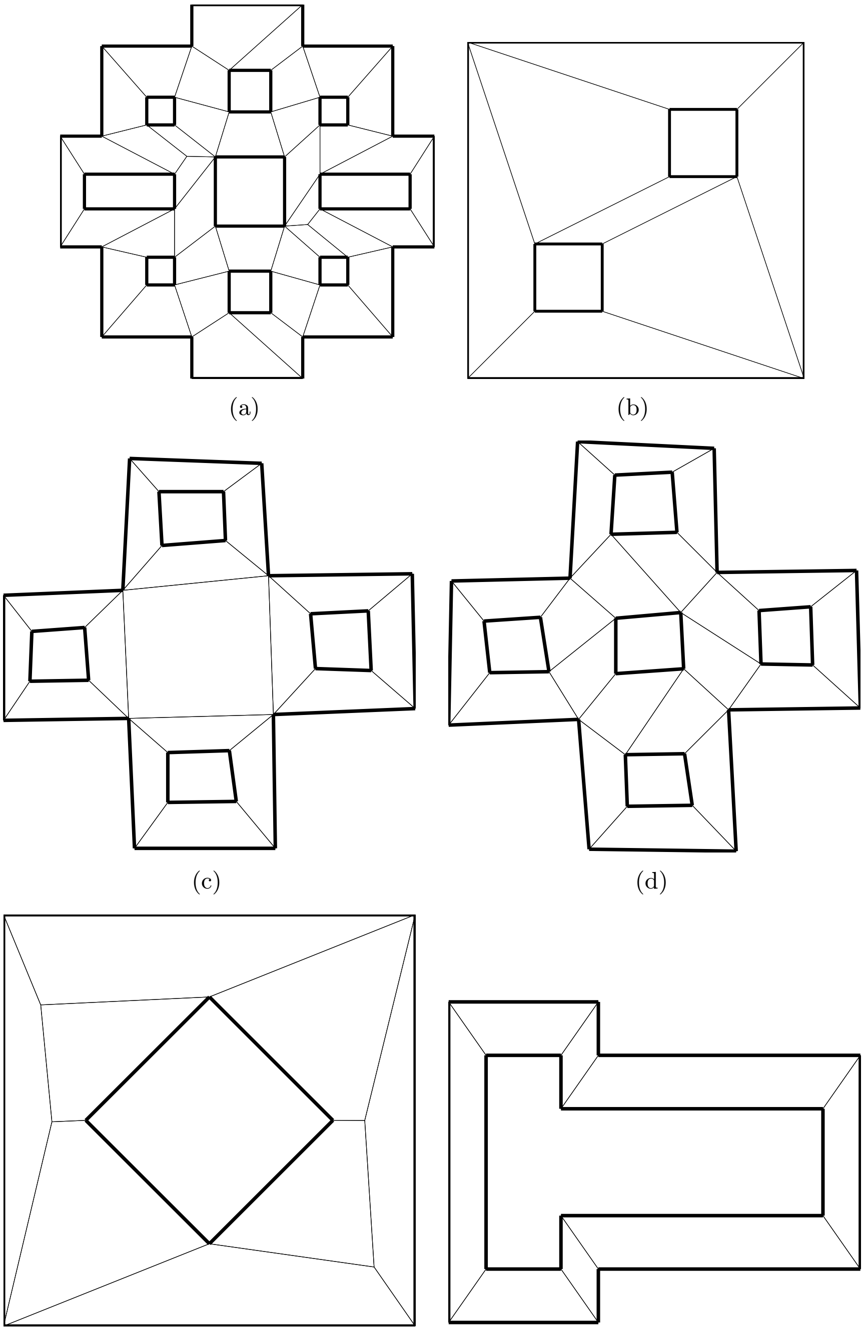


(a) (b)

(c) (d)

(e) (f)

Figure 3.43: Third set of results.

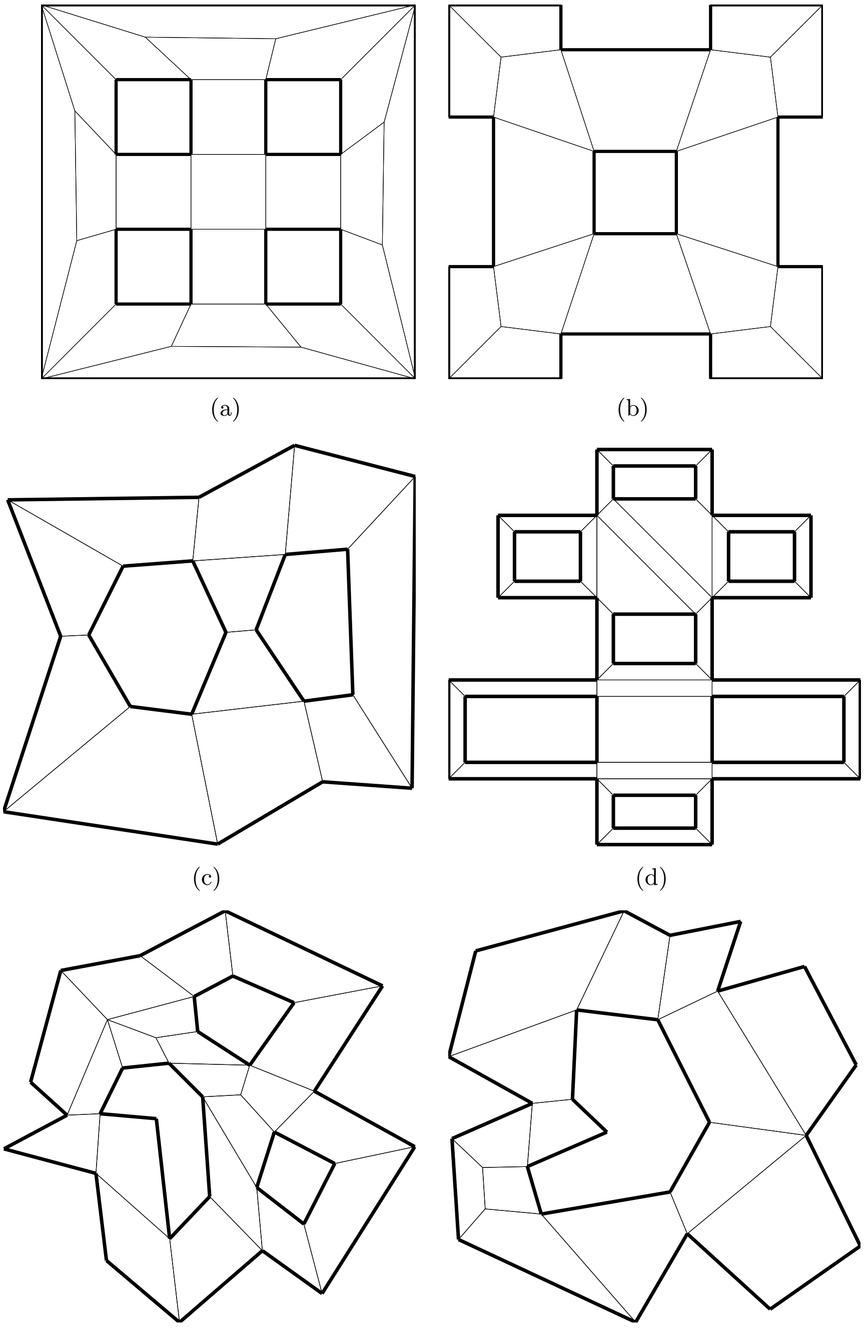


(a) (b)

(c) (d)

(e) (f)

Figure 3.44: Fourth set of results.



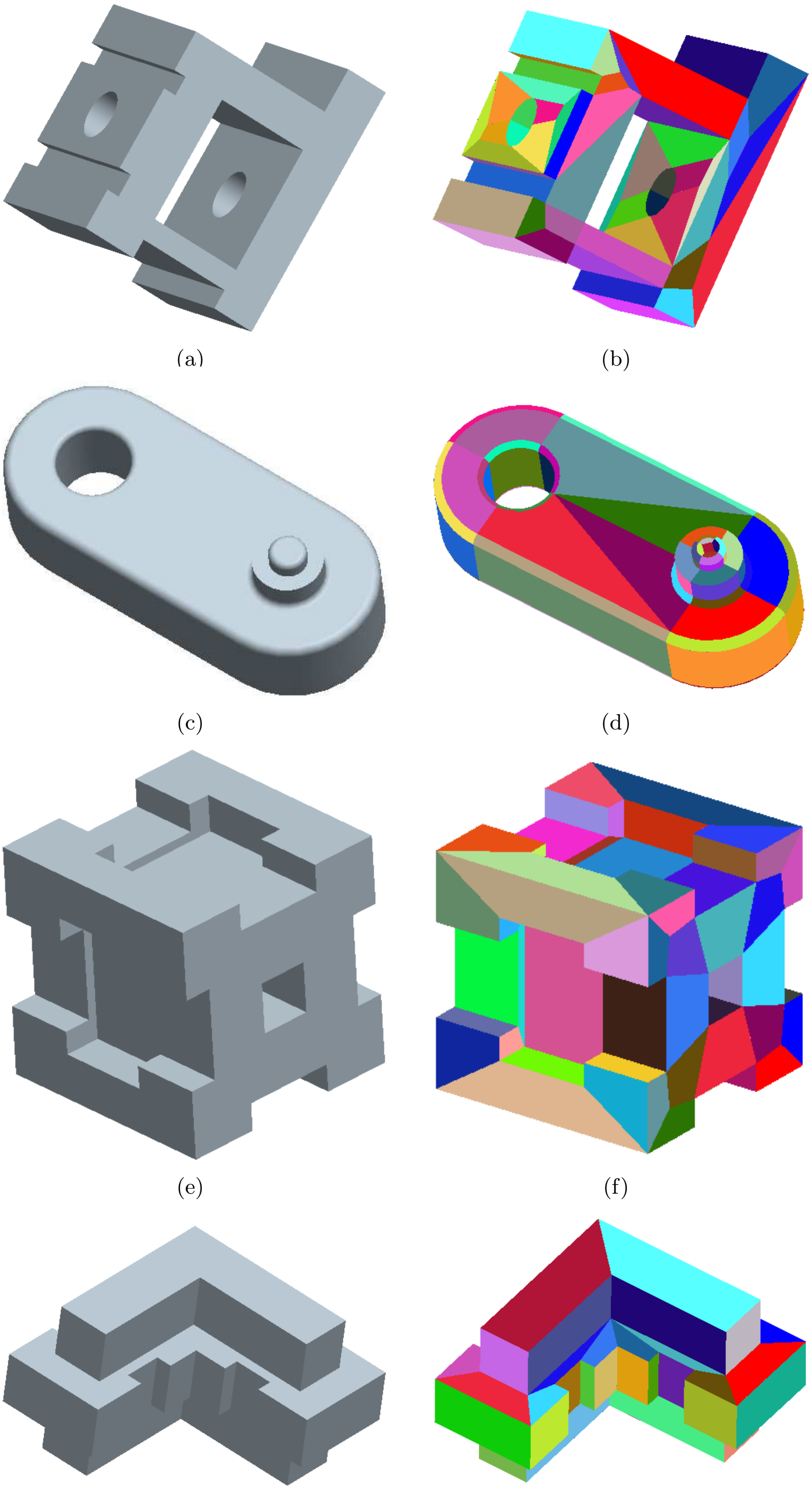
(a) (b) (c) (d) (e) (f)

Figure 3.45: Fifth set of results.

Objects	Trim. surf	Nb. 4-sided patches	Extraction	Tessellation
1-(a)(b)	28	94	0.391 sec	3.696 sec
1-(c)(d)	40	96	0.461 sec	3.395 sec
1-(e)(f)	58	120	0.431 sec	7.581 sec
1-(g)(h)	46	90	0.291 sec	5.137 sec
2-(a)(b)	22	62	0.380 sec	1.712 sec
2-(c)(d)	54	135	0.481 sec	5.257 sec
2-(e)(f)	35	104	0.401 sec	4.176 sec
2-(g)(h)	58	144	0.401 sec	7.781 sec

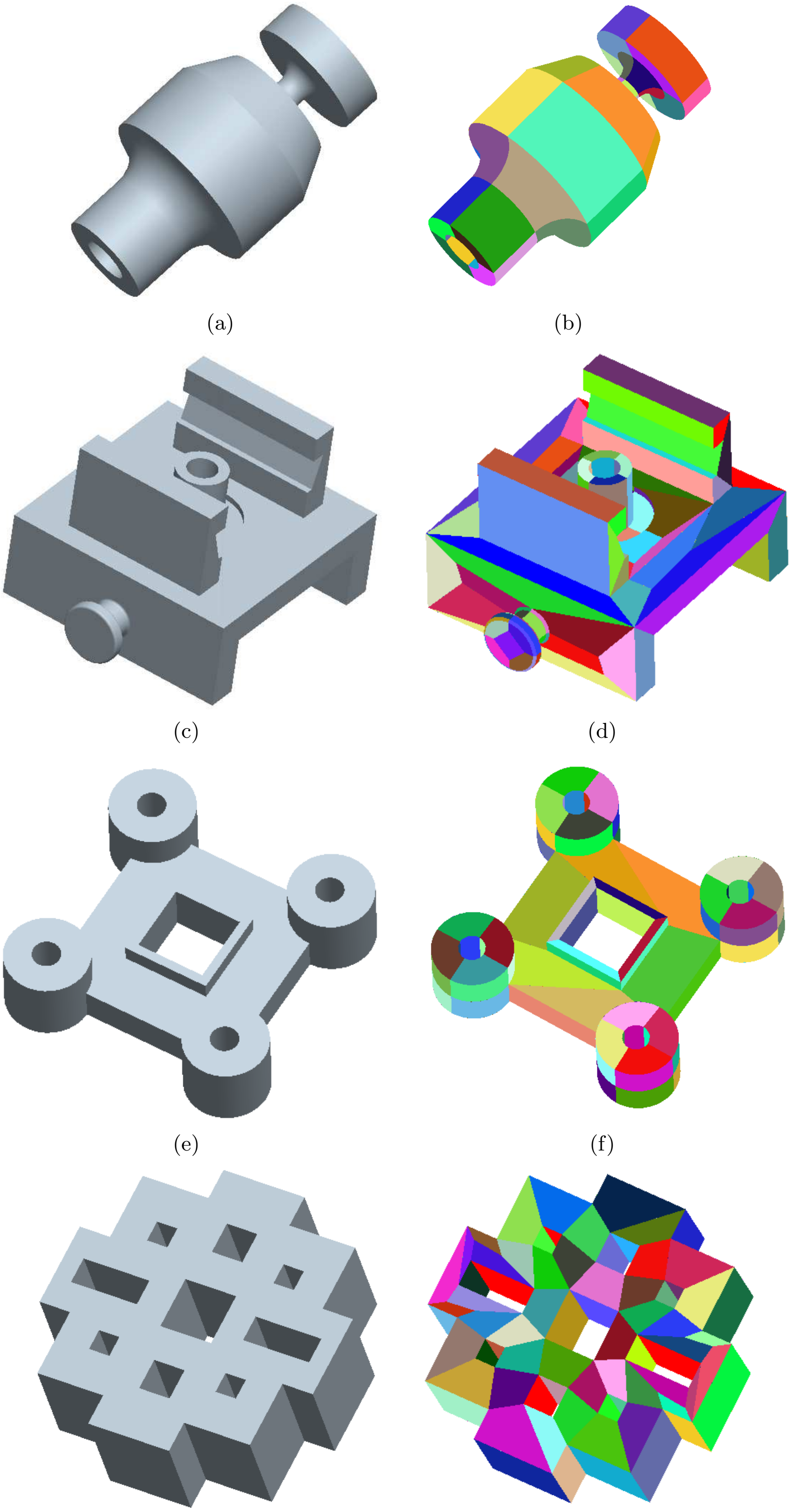
Table 3.1: Runtimes for parsings and tessellations

dows XP. In Table 3.1 we gather the time needed to parse the corresponding IGES files and to perform the tessellation. The information extraction includes automatic loading of the files, establishment of the information pertaining to the trimmed surfaces and conversion into C-data structures. We can find in the same table the numbers of initial multiply connected surfaces as well as the final numbers of four-sided subregions.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 3.46: First set of four-sided splittings



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 3.47: Second set of four-sided splittings

Chapter 4

MAPPING REGULARITY

Abstract: We consider the problem of characterizing whether a Coons map is a diffeomorphism from the unit square onto a planar domain delineated by four given curves. We aim primarily at having not only theoretically correct conditions but also practically efficient methods. Throughout the chapter we suppose that the given four boundary curves are presented in Bézier forms. We will prove three sufficient conditions: the first one is based upon the tangents of the boundary curves, the second one exploits the representation of the Jacobian in Bézier surface with a degree elevation when relevant, and the last one invokes the subdivision and polar forms techniques. Further, we will prove that the last condition is also necessary for sufficiently many subdivisions. We present a way of adaptive subdivision so as to make it efficient. On the other hand, we aim at having a diffeomorphism which is easy and fast to evaluate. Our attempt to generate a diffeomorphism from $[0,1]^2$ to a four-sided domain having curved sides is based upon Gordon patches. We will describe an automatic manner to specify internal cubic Bézier-spline curves that are to be subsequently interpolated by a Gordon patch. If we cannot find a diffeomorphism, then we subdivide the four-sided domain into simpler subregions. Numerical results are reported in order to illustrate the approaches.

4.1 Introduction

Determining whether a Coons map is a diffeomorphism is not just an interesting problem but it could have interesting applications. We would like to mention the numerical solution of integral equations on CAD objects [26, 27]. If the wavelet Galerkin scheme [109, 63] is used to solve the integral equation then the surface of the CAD object has to be split into patches, each having four sides, and we need a diffeomorphism from the unit square onto each patch. In an earlier work [101], we utilized transfinite interpolation in order to generate a parametric mapping from the unit square to a four-sided domain. It is on that account that we need to have an efficient method to characterize if a Coons map is a diffeomorphism. For given four curves α, β, γ, δ which enclose a planar domain (Fig. 4.1), the purpose of this chapter is to recognize if the corresponding Coons map is a diffeomorphism. We will suppose throughout that the boundary curves are Bézier curves.

In fact, we will prove three sufficient conditions which are mainly expressed with the help of the control points of the boundary curves and the blending functions. This chapter is organized as follows. In the next section, we will make some excursus on transfinite interpolation and state our problem more clearly. The first sufficient condition which is based on the tangents of the boundary curves will be described in section 4.3. We will discuss there also some interesting case in which the blending functions take only positive values. In section 4.4, we will propose and prove a sufficient condition based on the control points of the Jacobian of the Coons patch. We do not present any necessary condition until section 4.5 where we use subdivision methods to express our condition and where we have both low computational cost and efficiency as objective . We will propose an adaptive strategy to accomplish adaptive subdivision. Based on that, we will present an algorithm whose termination is ensured by a theorem which will also investigate. Finally numerical results are presented in the last section to test the performance of the proposed approaches in practice.

4.2 Transfinite interpolation and problem setting

Let us consider four continuously differentiable parametric curves α, β, γ, δ defined on the interval $[0,1]$ and taking values in $\mathbf{R}^2$. They are supposed to fulfill the compatibility condition (see Fig. 4.1) at the corners:

$$\alpha(0)=\delta(0)\,,\quad \alpha(1)=\beta(0)\,,\quad \gamma(0)=\delta(1)\,,\quad \gamma(1)=\beta(1). \tag{4.1}$$

Since our method of generating a mapping from the unit square to the four-sided domain bounded by the four curves is based on transfinite interpolation, we would like now to briefly recall some basic facts about this technique. For a more in-depth understanding regarding transfinite interpolation in general we direct the readers to [47, 54, 55, 56, 111].

We are interested in generating a parametric surface $\mathbf{x}(u,v)$ defined on the unit square $[0,1]^2$ such that the boundary of the image of $\mathbf{x}$ coincides with the given four curves:

$$\begin{array}{lll} \mathbf{x}(u,0)=\alpha(u) & \mathbf{x}(u,1)=\gamma(u) & \forall\, u\in[0,1] \\ \mathbf{x}(0,v)=\delta(v) & \mathbf{x}(1,v)=\beta(v) & \forall\, v\in[0,1]\ . \end{array} \tag{4.2}$$

This transfinite interpolation problem can be solved by a first order Coons patch whose construction involves the operators

$$(\mathcal{P}\mathbf{x})(u,v) \ := \ F_0(v)\mathbf{x}(u,0)+F_1(v)\mathbf{x}(u,1) \tag{4.3}$$

$$(\mathcal{Q}\mathbf{x})(u,v) \ := \ F_0(u)\mathbf{x}(0,v)+F_1(u)\mathbf{x}(1,v) \tag{4.4}$$

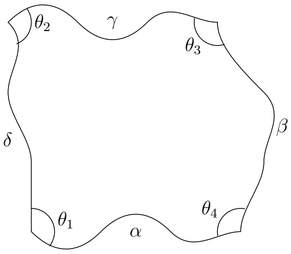


Figure 4.1: A four sided domain for Coons patch

where F_0 and F_1 denote two arbitrary smooth functions satisfying

$$F_i(j) = \delta_{ij} \quad i,j = 0,1 \quad \text{and} \qquad F_0(t) + F_1(t) = 1 \quad \forall\, t \in [0,1]. \tag{4.5}$$

Now, a Coons patch $\mathbf{x}$ can be defined by the relation

$$\mathcal{P} \oplus \mathcal{Q}(\mathbf{x}) = \mathbf{x}, \qquad \text{where} \tag{4.6}$$

$$\mathcal{P} \oplus \mathcal{Q} := \mathcal{P} + \mathcal{Q} - \mathcal{P}\mathcal{Q}. \tag{4.7}$$

The functions F_0, F_1 which are better known as blending functions can be chosen in several ways (see [39, 47, 68, 111]). On account of the fact that we need to verify diffeomorphisms, we choose in the sequel blending functions which are sufficiently smooth. The simplest case that one can take as bilinear blending function is

$$F_0(t) = 1 - t, \qquad F_1(t) = t. \tag{4.8}$$

We will see in our following discussion that the theoretical results that we derive are valid for a large range of blending functions. According to (4.7) we can express the solution to (4.2) in matrix form as:

$$\begin{aligned}\mathbf{x}(u,v) \;=\; & \begin{bmatrix} F_0(u) & F_1(u) \end{bmatrix} \begin{bmatrix} \delta(v) \\ \beta(v) \end{bmatrix} + \\ & \begin{bmatrix} \alpha(u) & \gamma(u) \end{bmatrix} \begin{bmatrix} F_0(v) \\ F_1(v) \end{bmatrix} - \\ & \begin{bmatrix} F_0(u) & F_1(u) \end{bmatrix} \begin{bmatrix} \alpha(0) & \gamma(0) \\ \alpha(1) & \gamma(1) \end{bmatrix} \begin{bmatrix} F_0(v) \\ F_1(v) \end{bmatrix}.\end{aligned} \tag{4.9}$$

From (4.6) it follows that $\mathbf{x}$ is of the form

$$\mathbf{x}(u,v) = - \begin{bmatrix} -1 \\ F_0(u) \\ F_1(u) \end{bmatrix}^T \begin{bmatrix} \mathbf{0} & \mathbf{x}(u,0) & \mathbf{x}(u,1) \\ \mathbf{x}(0,v) & \mathbf{x}(0,0) & \mathbf{x}(0,1) \\ \mathbf{x}(1,v) & \mathbf{x}(1,0) & \mathbf{x}(1,1) \end{bmatrix} \begin{bmatrix} -1 \\ F_0(v) \\ F_1(v) \end{bmatrix}. \tag{4.10}$$

This construction is due to S. M. Coons and it has been developed for free form surface modeling. The Boolean sum character of a Coons patch has been

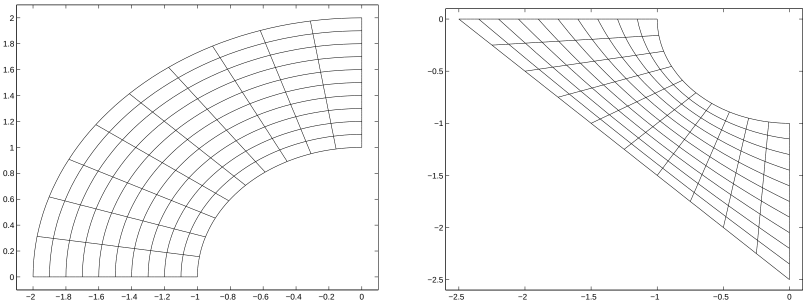

(a) (b)

Figure 4.2: Diffeomorphic Coons patches

discovered by W. Gordon. The differentiability of a Coons map is guaranteed if all curves and blending functions involved are themselves differentiable. In Figs. 4.2(a) and 4.2(b), we illustrate that for most practical cases a Coons patch is already a diffeomorphism. However, when the boundary curves become too wavy, like in Fig. 4.3, we observe overlapping isolines indicating that the mapping is not invertible.

The purpose of this chapter is to analyze under which conditions the Coons map (4.10) is a diffeomorphism . For that we need sufficient and necessary conditions which characterize the diffeomorphisms. Our fundamental aim is not only conditions which are theoretically valid. We aim also at having conditions which we can check in a fast and efficient way practically. Throughout the chapter we suppose that the boundary curves α, β, γ, δ for the Coons map are Bézier curves of degree n and that their Bézier points are α_i, β_i, γ_i, δ_i with $i = 0, \cdots, n$ respectively. That is,

$$\alpha(t) = \sum_{i=0}^{n} \alpha_i B_i^n(t), \qquad \beta(t) = \sum_{i=0}^{n} \beta_i B_i^n(t), \tag{4.11}$$

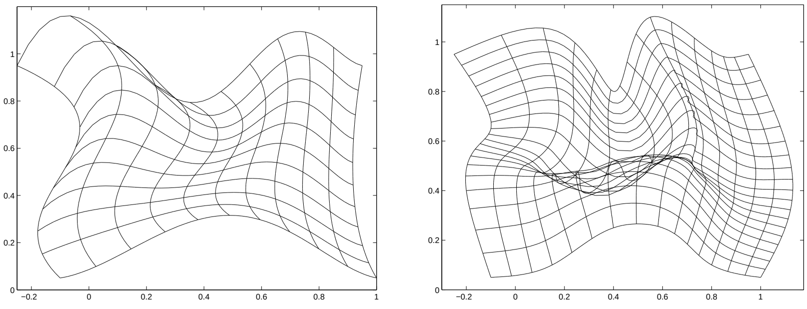
$$\gamma(t) = \sum_{i=0}^{n} \gamma_i B_i^n(t), \qquad \delta(t) = \sum_{i=0}^{n} \delta_i B_i^n(t). \tag{4.12}$$

The blending function is supposed also to be a polynomial which is represented in its Bézier form as

$$F_1(t) = 1 - F_0(t) = \sum_{i=0}^{n} \phi_i B_i^n(t). \tag{4.13}$$

Further we introduce the following constants.

$$\begin{array}{rcr} q & := & \inf\{F_1(t) : t \in [0,1]\}, \\ Q & := & \sup\{F_1(t) : t \in [0,1]\}, \\ \rho & := & \sup\{|F_1'(t)| : t \in [0,1]\}. \end{array} \tag{4.14}$$



(a) (b)

Figure 4.3: Undesired overspill phenomena

4.3 First sufficient condition

Before we introduce our first result, let us adopt some more notations. First, for $u, v \in [0,1]$ and $\zeta, \chi \in [q, Q]$, we will denote the combination of opposite tangents by

$$\begin{aligned} K_{u,\zeta} &:= (1-\zeta)\alpha'(u) + \zeta\gamma'(u) && (4.15) \\ L_{v,\chi} &:= (1-\chi)\delta'(v) + \chi\beta'(v). && (4.16) \end{aligned}$$

Then we introduce

$$M := \max\left\{ \sup_{(u,\zeta)\in[0,1]\times[q,Q]} \|K_{u,\zeta}\|, \sup_{(v,\chi)\in[0,1]\times[q,Q]} \|K_{v,\chi}\| \right\}. \tag{4.17}$$

Besides, we have the following maxima

$$\begin{aligned} S^1 &:= \max_{i=0,\cdots,n} \{\rho\|(\beta_i - \delta_i) + \phi_i(\gamma_0 - \gamma_n + \alpha_n - \alpha_0) + (\alpha_0 - \alpha_n)\|\} \\ S^2 &:= \max_{i=0,\cdots,n} \{\rho\|(\gamma_i - \alpha_i) + \phi_i(\alpha_n - \gamma_n + \gamma_0 - \alpha_0) + (\alpha_0 - \gamma_0)\|\}. \end{aligned}$$

Finally F is defined to be S^1 or S^2, whichever has the largest value.

Theorem 8 If there exists some $\kappa > 0$ such that

$$\det(K_{u,\zeta}, L_{v,\chi}) \geq \kappa \tag{4.18}$$

and

$$2MF + F^2 < \kappa, \tag{4.19}$$

then the Coons patch with respect to α, β, γ, δ is a diffeomorphism.

Proof

Some few computations reveal that the partial derivatives of the Coons patch are

$$\mathbf{x}_u(u,v) = F_1'(u)S_u + C_u\,, \qquad \mathbf{x}_v(u,v) = F_1'(v)S_v + C_v \qquad \text{where} \tag{4.20}$$

$$\begin{aligned} S_u &:= \beta(v) - \delta(v) + [F_0(v)\alpha(0) + F_1(v)\gamma(0)] - [F_0(v)\alpha(1) + F_1(v)\gamma(1)] \\ S_v &:= \gamma(u) - \alpha(u) + [F_0(u)\alpha(0) + F_1(u)\alpha(1)] - [F_0(u)\gamma(0) + F_1(u)\gamma(1)] \\ C_u &:= F_0(v)\alpha'(u) + F_1(v)\gamma'(u) \\ C_v &:= F_0(u)\delta'(v) + F_1(u)\beta'(v)\,. \end{aligned}$$

Therefore we obtain

$$S_u = \sum_{i=0}^{n}(\beta_i - \delta_i)B_i^n(v) + F_1(v)(\gamma_0 - \gamma_n + \alpha_n - \alpha_0) + (\alpha_0 - \alpha_n) \tag{4.21}$$

$$S_v = \sum_{i=0}^{n}(\gamma_i - \alpha_i)B_i^n(u) + F_1(u)(\alpha_n - \gamma_n + \gamma_0 - \alpha_0) + (\alpha_0 - \gamma_0) \tag{4.22}$$

After a few rearrangements

$$S_u = \sum_{i=0}^{n}[(\beta_i - \delta_i) + \phi_i(\gamma_0 - \gamma_n + \alpha_n - \alpha_0) + (\alpha_0 - \alpha_n)]B_i^n(v) \tag{4.23}$$

$$S_v = \sum_{i=0}^{n}[(\gamma_i - \alpha_i) + \phi_i(\alpha_n - \gamma_n + \gamma_0 - \alpha_0) + (\alpha_0 - \gamma_0)]B_i^n(u) \tag{4.24}$$

By using the definition of F, one obtains

$$|F_1'(u)|.\|S_u\| \leq F \quad \text{and} \quad |F_1'(v)|.\|S_v\| \leq F. \tag{4.25}$$

Because of multilinearity of the determinant function, we have

$$\begin{aligned} \det(\mathbf{x}_u, \mathbf{x}_v) &= F_1'(u)F_1'(v)\det(S_u, S_v) + F_1'(u)\det(S_u, C_v) + \\ &\quad + F_1'(v)\det(C_u, S_v) + \det(C_u, C_v). \\ &\geq \det(C_u, C_v) - \{|F_1'(u)F_1'(v)\det(S_u, S_v)| + \\ &\quad + |F_1'(u)\det(S_u, C_v)| + |F_1'(v)\det(C_u, S_v)|\} \\ &\geq \kappa - (F^2 + 2FM) > 0 \quad \text{due to (4.25) (4.18) and (4.19).} \end{aligned}$$

That means the Jacobian is nowhere zero. The inverse function theorem ensures therefore that the Coons patch is a diffeomorphism.

□

Remark 10 Condition (4.18) has some geometric interpretation. Suppose the bounds q and Q from relation (4.14) are 0 and 1 respectively. So, if one considers any convex combination K of the tangent vectors $\alpha'(u)$ and $\gamma'(u)$ and L of $\delta'(v)$

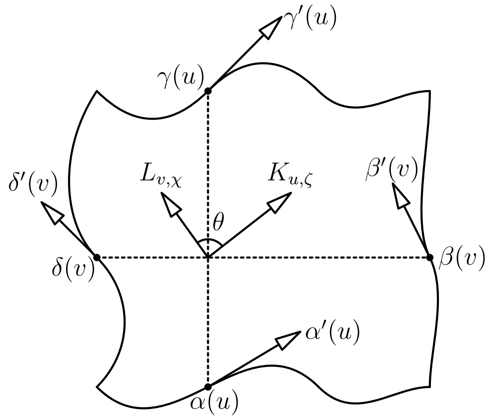


Figure 4.4: Tangents on a four sided domain for Coons patch

and $\beta'(v)$, then K and L are bounded away from being collinear and they are never zero in norm. Observe in Figure 4.4 the angle θ which represents with some respect the scaled determinant. If $(q, Q) \neq (0, 1)$ one can draw a similar figure after some rescalings.

Remark 11 A direct computation of M from relation (4.17) could be expensive or inaccurate in practice because it involves non-discrete information. In fact, the constant M could just as well be replaced by another constant that verifies for all $i = 0, \cdots, n-1$ and $j = 0, \cdots, n$

$$\begin{array}{rcl} n\|\phi_j(\gamma_{i+1} - \gamma_i + \alpha_i - \alpha_{i+1}) + (\alpha_{i+1} - \alpha_i)\| & \leq & M \\ n\|\phi_j(\beta_{i+1} - \beta_i + \delta_i - \delta_{i+1}) + (\delta_{i+1} - \delta_i)\| & \leq & M, \end{array} \tag{4.26}$$

which is easier to check. The idea of the proof remains fundamentally unchanged. Indeed, relation (4.26) implies in particular the following bounds

$$\|C_u\| \leq M \qquad \|C_v\| \leq M,$$

where

$$\begin{array}{rcl} C_u & := & F_0(v)\alpha'(u) + F_1(v)\gamma'(u) \\ C_v & := & F_0(u)\delta'(v) + F_1(u)\beta'(v). \end{array} \tag{4.27}$$

Remark 12 In the previous theorem we have treated the very general case in which the blending functions F_0 and F_1 could take any sign. Still, more can be stated if they are to take only positive values as we will see in the next remark. That is for example the case if we choose the following blending functions:

$$F_0(t) \ := \ 2t^3 - 3t^2 + 1 = B_0^3(t) + B_1^3(t) \tag{4.28}$$

$$F_1(t) \ := \ -2t^3 + 3t^2 = B_2^3(t) + B_3^3(t). \tag{4.29}$$

Interestingly, we will see that condition (4.18) could be replaced by another one that takes a discrete form which is of course of practical concern.

Remark 13 Suppose that the blending functions are positive:

$$F_0(t) \geq 0, \qquad F_1(t) \geq 0 \qquad \forall\, t \in [0,1]. \tag{4.30}$$

If we replace condition (4.18) of the previous theorem by

$$\begin{array}{rcl} n^2 \det[\alpha_{i+1}-\alpha_i, \delta_{j+1}-\delta_j] & > & 0\,, \\ n^2 \det[\alpha_{i+1}-\alpha_i, \beta_{j+1}-\beta_j] & > & 0\,, \\ n^2 \det[\gamma_{i+1}-\gamma_i, \delta_{j+1}-\delta_j] & > & 0\,, \\ n^2 \det[\gamma_{i+1}-\gamma_i, \beta_{j+1}-\beta_j] & > & 0\,, \end{array} \tag{4.31}$$

for all $i, j = 0, \cdots, n-1$ and define $\kappa > 0$ to be the minimum of them , then we can deduce the same claim.

Proof

According to the multilinearity of the determinant again, we have (see also definitions from relation (4.27))

$$\begin{array}{rcl} \det[C_u, C_v] & = & F_0(v)F_0(u)\det[\alpha'(u), \delta'(v)] + F_0(v)F_1(u)\det[\alpha'(u), \beta'(v)] + \\ & & F_1(v)F_0(u)\det[\gamma'(u), \delta'(v)] + F_1(v)F_1(u)\det[\gamma'(u), \beta'(v)]. \end{array}$$

On account of the fact that

$$\alpha'(u) = \sum_{i=0}^{n-1} n(\alpha_{i+1}-\alpha_i)B_i^{n-1}(u)$$

and similar relations for β, γ, δ, we deduce from (4.31) that

$$\det[C_u, C_v] \geq \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} \kappa B_i^{n-1}(u)B_j^{n-1}(v).$$

Since the Bernstein polynomials form a partition of unity, we deduce the result.

□

Now, we would like to state a simple corollary which is very important for the next chapter.

Corollary 3 One can generate a diffeomorphism from the unit square onto a planar strictly convex quadrilateral.

Proof

Consider a strictly convex quadrilateral $[A_1, A_2, A_3, A_4]$ and let us denote by θ_i the internal angles at the corners. One can generate four Bézier curves α, β, γ, δ as

$$\begin{aligned} \alpha(t) &:= A_1 B_0^1(t) + A_2 B_1^1(t) && (4.32)\\ \beta(t) &:= A_2 B_0^1(t) + A_3 B_1^1(t) && (4.33)\\ \gamma(t) &:= A_4 B_0^1(t) + A_3 B_1^1(t) && (4.34)\\ \delta(t) &:= A_1 B_0^1(t) + A_4 B_1^1(t). && (4.35) \end{aligned}$$

Since the quadrilateral is strictly convex, we have $\alpha_i < \pi$. Thus, the conditions in (4.31) are satisfied. By choosing positive blending function as in (4.30), we deduce the claim.

□

Lemma 2 Suppose the boundary curves α, β, γ, δ and the blending function are given as before. Then the Coons patch is a Bézier surface

$$\mathbf{x}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{n} \mathbf{E}_{ij} B_i^n(u) B_j^n(v) \tag{4.36}$$

where the control points are

$$\begin{aligned} \mathbf{E}_{ij} \;:=\; & \delta_j - \alpha_0 + \alpha_i + \phi_j(\gamma_i - \alpha_i + \alpha_0 - \gamma_0) \\ & \phi_i[\alpha_0 - \alpha_n + \beta_j - \delta_j + \phi_j(\alpha_n - \gamma_n + \gamma_0 - \alpha_0)] \end{aligned} \tag{4.37}$$

Proof

Obvious. See also ([40]) for a similar discussion.

□

4.4 Second sufficient condition

Theorem 9 Consider the assumption above and define

$$D(i, j, k, l) := n^2 \det[\mathbf{E}_{i+1,j} - \mathbf{E}_{ij}, \mathbf{E}_{k,l+1} - \mathbf{E}_{kl}] \tag{4.38}$$

$$\begin{aligned} C(i, j, k, l) \;:=\; & \tfrac{l}{n}\left[\tfrac{i}{n} D(i-1, j, k, l-1) + (1 - \tfrac{i}{n}) D(i, j, k, l-1)\right] + \\ & (1 - \tfrac{l}{n})\left[\tfrac{i}{n} D(i-1, j, k, l) + (1 - \tfrac{i}{n}) D(i, j, k, l)\right] \end{aligned} \tag{4.39}$$

If for all $p, q = 0, \cdots, 2n$

$$J_{pq} := \sum_{\substack{i+k=p \\ j+l=q}} C(i, j, k, l) \frac{\binom{n}{i}\binom{n}{k}}{\binom{2n}{i+k}} \frac{\binom{n}{j}\binom{n}{l}}{\binom{2n}{j+l}} > 0 \tag{4.40}$$

then the Coons patch is a diffeomorphism.

Proof

The partial derivatives are

$$\mathbf{x}_u(u,v) = \sum_{i=0}^{n-1}\sum_{j=0}^{n} n(\mathbf{E}_{i+1,j}-\mathbf{E}_{ij})B_i^{n-1}(u)B_j^n(v) \tag{4.41}$$

$$\mathbf{x}_v(u,v) = \sum_{k=0}^{n}\sum_{l=0}^{n-1} n(\mathbf{E}_{k,l+1}-\mathbf{E}_{kl})B_k^{n}(u)B_l^{n-1}(v) \tag{4.42}$$

Therefore we obtain the determinant

$$\mathbf{det}(\mathbf{x}_u,\mathbf{x}_v) = \sum_{i=0}^{n-1}\sum_{j=0}^{n}\sum_{k=0}^{n}\sum_{l=0}^{n-1} D(i,j,k,l)B_i^{n-1}(u)B_j^n(v)B_k^n(u)B_l^{n-1}(v) \tag{4.43}$$

After application of degree elevation with respect to the indices i and l we obtain

$$\mathbf{det}(\mathbf{x}_u,\mathbf{x}_v) = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=0}^{n}\sum_{l=0}^{n} C(i,j,k,l)B_i^{n}(u)B_j^n(v)B_k^n(u)B_l^{n}(v) \tag{4.44}$$

By using the product formula

$$B_i^n(t)B_k^n(t) = \frac{\binom{n}{i}\binom{n}{k}}{\binom{2n}{i+k}}B_{i+k}^{2n}(t) \tag{4.45}$$

the Jacobian can be written as

$$J(u,v) := \det(\mathbf{x}_u,\mathbf{x}_v) = \sum_{p=0}^{2n}\sum_{q=0}^{2n} J_{pq}B_p^{2n}(u)B_q^{2n}(v) \tag{4.46}$$

Therefore the Coons patch $\mathbf{x}$ is a diffeomorphism.

□

4.5 Sufficient and necessary condition

In the previous theorems we have presented two methods for verifying whether a Coons patch describes a diffeomorphism. In fact, they give only sufficient conditions. Let us describe a short contrast between those two approaches. As far as computational cost is concerned, the second approach is more computationally intensive than the first one as it will be observed in the numerical experiments from section 4.6. On the other hand, the second approach is also more sensitive. That is, it can provide some response whereas the first one fails. Additionally, as we degree-elevate the boundary Bézier curves, the second approach becomes more and more sensitive. Computational cost is of course a trade-off to consider if one chooses n large during the degree elevation of the second test. In the next discussion, we will propose a method to achieve at the same time low computational cost and effective results. We will demonstrate a condition, based upon subdivision methods, which is both necessary and sufficient.

4.5.1 Subdivision

Before we see a necessary condition, let us see the following fact. A Bézier surface F defined on $[a,b]\times[c,d]$ can be subdivided into four Bézier surfaces A, B, C, D which are respectively defined on

$$\begin{aligned} I^A &:= [a,0.5(a+b)] &\times& \quad [c,0.5(c+d)] && (4.47)\\ I^B &:= [a,0.5(a+b)] &\times& \quad [0.5(c+d),d] && (4.48)\\ I^C &:= [0.5(a+b),b] &\times& \quad [c,0.5(c+d)] && (4.49)\\ I^D &:= [0.5(a+b),b] &\times& \quad [0.5(c+d),d] && (4.50) \end{aligned}$$

by using the following recursions. Suppose the control points of F are F_{ij} $i,j=0,\cdots,n$. We define

$$\left\{ \begin{array}{lll} F_{ij}^{[0]} & := & F_{ij} \quad \text{and} \\ F_{ij}^{[k]} & := & 0.5(F_{i-1,j}^{[k-1]} + F_{ij}^{[k-1]}) \end{array} \right. \tag{4.51}$$

$$\left\{ \begin{array}{lll} P_{ij}^{[0]} & := & F_{ij}^{[i]} \quad \text{and} \\ P_{ij}^{[k]} & := & 0.5(P_{i,j-1}^{[k-1]} + P_{ij}^{[k-1]}) \end{array} \right. \left\{ \begin{array}{lll} Q_{ij}^{[0]} & := & F_{nj}^{[n-i]} \quad \text{and} \\ Q_{ij}^{[k]} & := & 0.5(Q_{i,j-1}^{[k-1]} + Q_{ij}^{[k-1]}). \end{array} \right. \tag{4.52}$$

The control points of A, B, C and D are respectively $A_{ij} := P_{ij}^{[j]}$, $B_{ij} := P_{in}^{[n-j]}$, $C_{ij} := Q_{ij}^{[j]}$, $D_{ij} := Q_{in}^{[n-j]}$. We have in particular

$$F(u,v) = Q(u,v) \qquad \text{if} \quad (u,v)\in I^Q,$$

where $Q = A, B, C$, or D.

Theorem 10 Let us adopt the same notations as in the previous statement. Suppose that the Coons patch $\mathbf{x}$ defined with α, β, γ, δ is a diffeomorphism. Suppose further that we have subdivided J into σ^2 Bézier surfaces J^{ij} which are defined on

$$I^{ij} := [(i-1)/\sigma, i/\sigma] \times [(j-1)/\sigma, j/\sigma] \quad i,j = 1,\cdots,\sigma. \tag{4.53}$$

Denote by J^{ij}_{pq}, $p,q = 0,\cdots,2n$ the control points of the Bézier surface J^{ij}.

We claim that if σ is sufficiently large then J^{ij}_{pq} is of constant sign uniformly on $i,j = 1,\cdots,\sigma$ and on $p,q = 0,\cdots,2n$.

Proof

On the one hand, the Jacobian $J(u,v)$ must be of constant sign because it is never zero. Without loss of generality we suppose that it is positive:

$$J(u,v) > 0 \qquad \forall (u,v) \in [0,1] \times [0,1]. \tag{4.54}$$

Since the function J is continuous on the compact $[0,1] \times [0,1]$, there must exist some $\rho > 0$ such that

$$J(u,v) \geq \rho \qquad \forall\, (u,v) \in [0,1] \times [0,1]. \tag{4.55}$$

On the other hand, let us fix (i,j) and let us denote by $[a,b] \times [c,d]$ the interval I^{ij} in order to simplify the notation. We are going to use the notation $s,..[m]..,s$ in order to stress that s is to be repeated m times. Further, let us introduce the blossom [113, 98] function $P^{ij}(u_1,\cdots,u_{2n}; v_1,\cdots,v_{2n})$ corresponding to the polynomial J^{ij}:

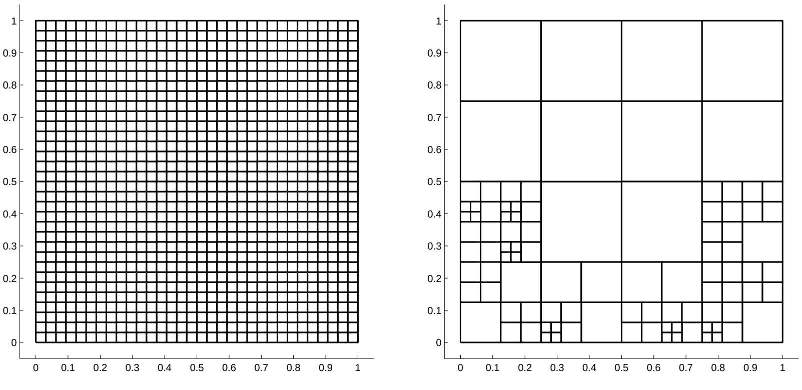
$$J^{ij}(u,v) = P^{ij}(u,..[2n]..,u; v,..[2n]..,v). \tag{4.56}$$

Define $h := 1/(2n\sigma)$ and $a_p := a + ph$, $c_q := c + qh$ for $p = 0,\cdots,2n$ and $q = 0,\cdots,2n$. We would like now to apply multivariate Taylor development of the first order to the blossom P^{ij} at $(a_p,..[2n]..,a_p; c_q,..[2n]..,c_q)$.

$$\begin{aligned}
&P^{ij}(a,..[2n-p]..,a,b,..[p]..,b; c,..[2n-q]..,c,d,..[q]..,d) = \\
&P^{ij}(a_p,..[2n]..,a_p; c_q,..[2n]..,c_q) + \\
&\sum_{r=1}^{2n-p} (a-a_p)\frac{\partial}{\partial u_r} P^{ij}(a_p,..[2n]..,a_p; c_q,..[2n]..,c_q) + \\
&\sum_{r=2n-p+1}^{2n} (b-a_p)\frac{\partial}{\partial u_r} P^{ij}(a_p,..[2n]..,a_p; c_q,..[2n]..,c_q) + \\
&\sum_{r=1}^{2n-q} (c-c_q)\frac{\partial}{\partial v_r} P^{ij}(a_p,..[2n]..,a_p; c_q,..[2n]..,c_q) + \\
&\sum_{r=2n-q+1}^{2n} (d-c_q)\frac{\partial}{\partial v_r} P^{ij}(a_p,..[2n]..,a_p; c_q,..[2n]..,c_q) + \mathcal{O}(h^2).
\end{aligned}$$

Due to the symmetry [98, 112] of the blossom function we obtain

$$\begin{aligned}
&P^{ij}(a,..[2n-p]..,a,b,..[p]..,b; c,..[2n-q]..,c,d,..[q]..d) = \\
&P^{ij}(a_p,..[2n]..,a_p; c_q,..[2n]..,c_q) + \mathcal{O}(h^2).
\end{aligned} \tag{4.57}$$



(a) (b)

Figure 4.5: Multiple subdivisions: (a)uniform (b)adaptive

In other words, we have the following equality regarding the control points

$$J_{pq}^{ij} = J^{ij}(a_p, c_q) + \mathcal{O}(h^2). \tag{4.58}$$

Combining (4.55) and (4.58), there must exist some constant $C > 0$ such that we have the following relations

$$\begin{aligned} J_{pq}^{ij} &= J^{ij}(a_p, c_q) + J_{pq}^{ij} - J^{ij}(a_p, c_q) && (4.59) \\ &\geq J(a_p, c_q) - Ch^2 && (4.60) \\ &\geq \rho - Ch^2 . && (4.61) \end{aligned}$$

Since $Ch^2 = \frac{C}{(2n\sigma)^2}$ tends to 0 as σ tends to infinity, we deduce that $J_{pq}^{ij} > 0$ for σ sufficiently large and it concludes the proof.

□

4.5.2 Adaptivity

So far, we have always described something which should work if the Coons patch $\mathbf{x}$ is a diffeomorphism. What happens if it is not? On that account, we want to state the following result.

Theorem 11 Adopt the same notations as in the previous theorem but suppose now that the Coons patch is not a diffeomorphism.

There must exist (i_1, j_1) and (i_2, j_2) such that

$$\begin{cases} J_{pq}^{i_1,j_1} > 0 & \forall\, p,q = 0,\cdots,2n \\ J_{pq}^{i_2,j_2} < 0 & \forall\, p,q = 0,\cdots,2n. \end{cases} \tag{4.62}$$

Proof

This theorem is demonstrated in a very similar way as the preceding one. Therefore we omit the proof.

□

Remark 14 The condition in (4.62) can of course be used in the next algorithm as an abortion criterion. That is, once the condition (4.62) occurs in the loop (see step 1), we terminate the algorithm and conclude at the same time that the Coons patch is not a diffeomorphism.

In the preceding theorems, we have subdivided the unit square uniformly (see Fig. 4.5(a)) which is not always essential in practice. It is advisable to apply the former Bézier subdivisions only to those patches which do not give responses (affirmative or negative) as it can be seen in Fig. 4.5(b). Before we give our algorithm of adaptive subdivision, let us observe the following simple fact. If the Bézier coefficients of the surface F from remark 4.5.1 are all positive then so are those of the resulting surfaces A, B, C, D. The theoretical results that we derived earlier give rise to the following algorithm.

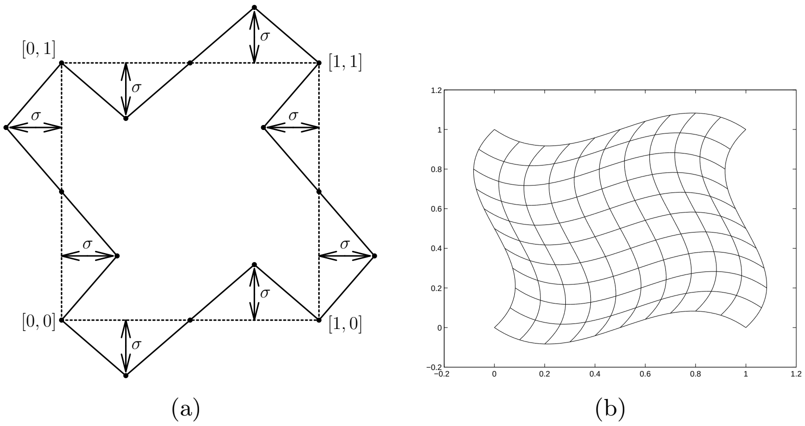
Algorithm: Adaptive regularity

step 0 : Initialize the grid G to have only one cell $[0,1] \times [0,1]$ and compute the Bézier coefficients J_{pq} according to (4.40).

step 1 : Traverse the cells $I = [a,b] \times [c,d]$ of the grid G

- Check if all coefficients J_{pq}^{I} have fixed sign irrespective of the indices $p, q = 0, \cdots, 2n$
- If not, split I into four cells $I1$, $I2$, $I3$, $I4$ and subdivide the Bézier surface J^{I} into four Bézier surfaces J^{I1}, J^{I2}, J^{I3}, J^{I4} as in Remark 4.5.1.
- If there was some cell $\tilde{I} \neq I$ for which $J_{pq}^{\tilde{I}}$ was always positive (resp. negative) for all $p, q = 0, \cdots, 2n$ and the current J_{pq}^{I} is always negative (resp. positive), then abort the whole algorithm and conclude that the Coons map is NOT a diffeomorphism as discussed in Theorem 4.

step 2 : If in step 1, all J_{pq}^{I} have fixed sign irrespective of the cell I and the indices $p, q = 0, 1, \cdots, 2n$ then terminate the algorithm and conclude that the Coons map is a diffeomorphism otherwise go to step 1.



(a) (b)

Figure 4.6: (a)Control polygons of the boundary curves (b)Coons map for $\sigma = 0.216$.

4.6 Numerical results

This section will be occupied by numerical results which support the formerly described theories as well as algorithm. As a first test, we consider a Coons map whose boundary curves can be controlled by a parameter σ. More precisely, let us consider the control polygons which are seen in Fig. 4.6(a). A special case of such a map for $\sigma = 0.216$ is portrayed in Fig. 4.6(b). If the parameter σ is zero then we retrieve the unit square. The purpose of this first test is to investigate numerically the theoretical conditions that we discussed earlier. On that account, we want to see the performance of the three sufficient conditions to verify diffeomorphism. For $\sigma > 0.36$, the Coons map does not present any diffeomorphism any more. We want therefore to vary the value of σ which will then range from 0 to 0.35.

The numerical data that are in Table 4.1 have been collected from an Intel Pentium 4 processor 2.66 GHz running Windows XP. For any given σ in the first column, we find the corresponding results of Theorems 8, 9 and 10 in the three last columns respectively.

If the conditions in the theorems are successful then we provide the time needed to run the test. If the map is a diffeomorphism but our condition could not detect that, then we report a failure information in the table ('*fails*'). For the results of the second test, additional information about the required degree n of the boundary curves is provided in parentheses. In other words, we degree-elevate the curves until the second test gives some response. The same remark applies to the third test with the number of required cells in parenthesis.

From the figures in the table, we see that the first test is only successful till the value of σ is 0.216, a case which corresponds to the map in Fig. 4.6(b). On

σ	first test	second test	adaptive subdivision
0.000	1.1 E-07 sec	1.2 E-05 sec (n=4)	1.2 E-05 sec (nb cells=1)
0.036	1.1 E-07 sec	1.2 E-05 sec (n=4)	1.2 E-05 sec (nb cells=1)
0.072	1.1 E-07 sec	1.2 E-05 sec (n=4)	1.2 E-05 sec (nb cells=1)
0.216	1.1 E-07 sec	1.2 E-05 sec (n=4)	1.2 E-05 sec (nb cells=1)
0.252	*fails*	1.2 E-05 sec (n=4)	1.2 E-05 sec (nb cells=1)
0.280	*fails*	1.2 E-05 sec (n=4)	1.2 E-05 sec (nb cells=1)
0.324	*fails*	7.8 E-002 sec (n=9)	1.5 E-002 sec(nb cells=4)
0.350	*fails*	5.6 E+010 sec (n=40)	1.5 E-002 sec(nb cells=4)

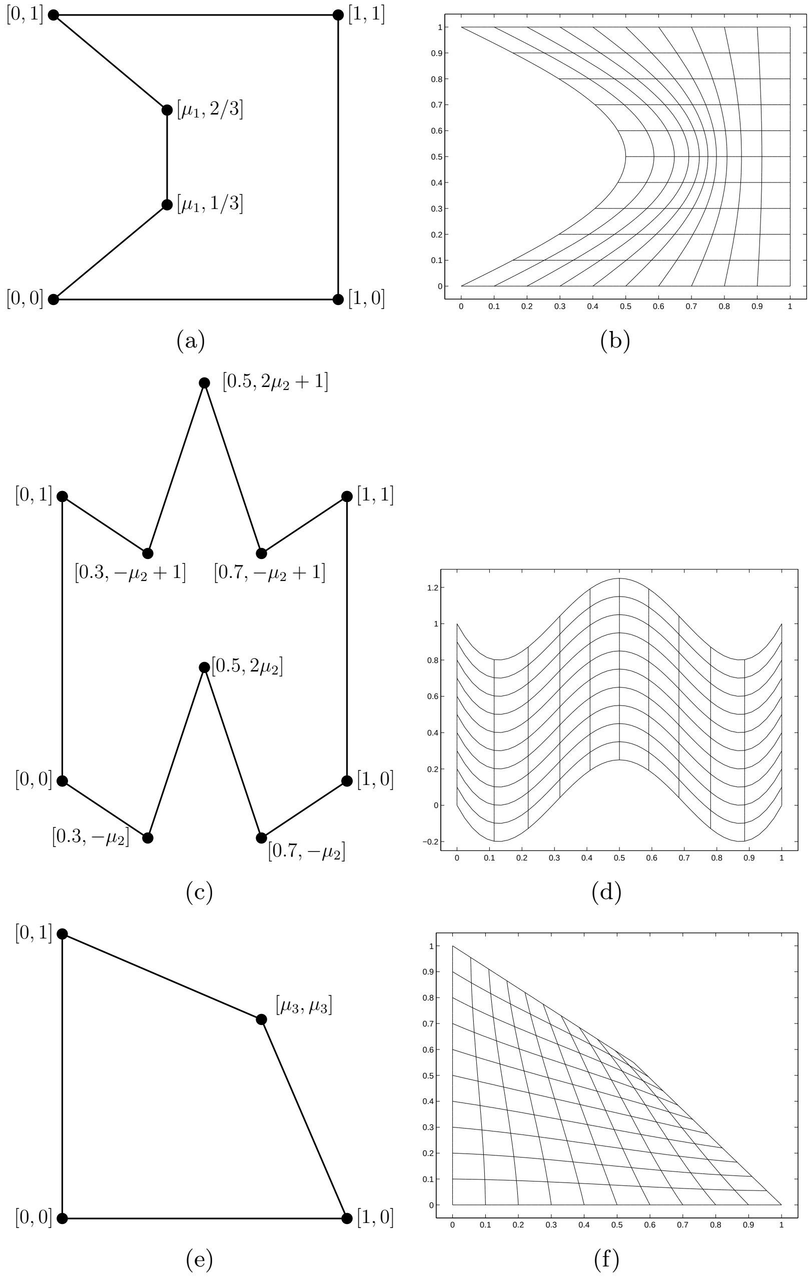
Table 4.1: Performance of the three conditions

the other hand, it is also to be noticed that the first test is comparatively less computationally intensive than the other two tests. A closer look at Table 4.1 reveals that the second test is not any longer efficient when the degree n is too large because one single test lasts approx. one minute.

Our next experiment is to consider some Coons maps and to investigate in which case they present diffeomorphisms. Let us consider the Coons map whose control polygons are seen in Fig. 4.7(a). Observe that the boundary curve δ is specified by some parameter μ_1 which could be positive or negative. For examples we see in Fig. 4.7(b) the result if the parameter takes the value $\mu_1 = 0.7$. We have used the former theorems to characterize whether the resulting Coons map is a diffeomorphism: if μ_1 is negative then we have always a diffeomorphism. If $\mu_1 \in [0, 0.88]$ then we still have a diffeomorphism. For $\mu_1 = 0.89$, we do not have any more diffeomorphism.

Now we want to do a similar test but this time we want to apply it to a control polygon where the boundary curve α is parallel to the boundary curve γ and the other curves are straight lines. As illustrated in Fig. 4.7(c), the control points of the curved boundaries α and γ are determined by some constant μ_2. In Fig. 4.7(d), we can see the Coons map in which we chose $\mu_2 = 1$. After applying the former theory in which we let μ_2 vary inside the interval $[-10, 10]$ we have concluded that the resulting Coons map is consistently a diffeomorphism irrespective of the value of μ_2.

Another example is depicted in Fig. 4.7(e) where the parameter $\mu_3 > 0$ controls the distance of one corner to the origin. Our former theorems allow us to conclude that the corresponding Coons map is a diffeomorphism as long as $\mu_3 > 0.5$ and it is not a diffeomorphism for $\mu_3 \in [0, 0.5]$.

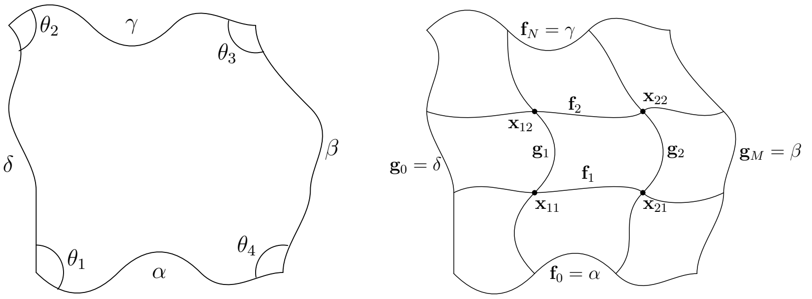


(a) (b)

(c) (d)

(e) (f)

Figure 4.7: Examples of diffeomorphisms



(a) (b)

Figure 4.8: (a)A four sided domain for Gordon patch, (b)A network of curves for Gordon patch

4.7 Mapping search

In the previous sections, we have shown methods and theorems to verify if a given Coons map is a diffeomorphism. In the next sections, we attempt to generate a diffeomorphism $\mathbf{x}$ from the unit square onto a given four-sided region $\mathcal{F}$ which is bounded by four curves α, β, γ, δ. We suppose that the bounding curves α, β, γ, δ are represented as bijective Bézier curves such that $\dot{\alpha}$, $\dot{\beta}$, $\dot{\gamma}$, $\dot{\delta}$ are continuous and

$$\dot{\alpha}(t) \neq \mathbf{0}, \qquad \dot{\beta}(t) \neq \mathbf{0}, \qquad \dot{\gamma}(t) \neq \mathbf{0}, \qquad \dot{\delta}(t) \neq \mathbf{0}, \quad \forall\, t \in]0,1[. \tag{4.63}$$

Those conditions imply in particular that the bounding curves do not contain any cusp or sharp corners. The diffeomorphism generation consists in iteratively inserting some internal curves and in considering the Gordon patch afterward. If we do not have yet any diffeomorphism after some iterations, then we subdivide the region $\mathcal{F}$. We suppose that the angles at the corners θ_1, θ_2, θ_3, θ_4 belong to $]0,\pi[$ as in Fig. 4.8(a). The property that $\mathbf{x}$ should be a diffeomorphism is not required during the CAD design process. It is only used in a subsequent application of numerical solution of integral equations [26, 27, 109, 63] on CAD objects.

Remark 15 The problem of mapping generation is very well known in the context of complex analysis (see [65] and reference there). A common approach consists in approximating the boundaries of $\mathcal{F}$ by a polygon and in generating a conformal mapping from the unit square to the polygon. That method is numerically realized by the analyses of Schwarz-Christofel [117, 65] where the mapping is given in the form of a complicated integral requiring numerical evaluation. Some coefficients in the integral should be determined by solving a nonlinear system of equations. Some generalizations of the Schwarz-Christofel approach are described in [18, 19] where one approximates the boundary with piecewise polynomials. However, those existing methods cannot be used in our problems because the following two points are of great importance in our objective:

- Retaining the exact geometry of the trimmed surface $\mathcal{F}$.
- Easy and fast evaluation of the resulting mappings $\mathbf{x}$.

We will achieve those two objectives by following the idea of transfinite interpolation described by Gordon and Hall in [54] and by giving an efficient approach of finding the curves to be inserted.

4.7.1 Overspill phenomenon

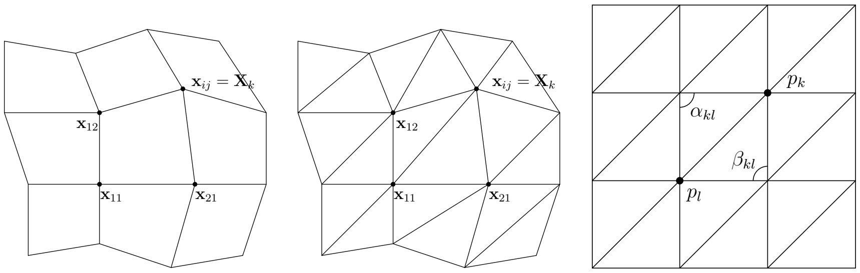
The Coons patch $\mathbf{x}$ given by (4.9) interpolates the prescribed boundary curves but it is not necessarily a diffeomorphism. The boundary curves α, β, γ, δ being continuously differentiable, the differentiability of $\mathbf{x}$ is guaranteed if the blending functions are chosen to be differentiable. The real problem in having a diffeomorphism is then the invertibility and nonvanishing of the Jacobian. In most practical cases, like in Fig. 4.2, a Coons patch is already a diffeomorphism. Still, sometimes two unsatisfactory situations may occur. First, some isoline curves may partially reside outside the boundary as in the case of Fig. 4.3(a). Another problem is that all isolines reside inside the domain $\mathcal{F}$ but some of them overlap as in Fig. 4.3(b). Those situations, better known [54] as '*overspill phenomena*', are not suitable to our objective. In the next discussion, we will propose a method based on Gordon patches, which are a generalized form of Coons patches, to eliminate the overspill phenomena.

4.7.2 Gordon patch

Since our method of generating a mapping from the unit square to a four sided domain $\mathcal{F}$ is based on Gordon patch [56, 111], let us introduce briefly some interesting definitions. We will only state things which are immediately related to our problems. We direct the readers to [47, 55] for more complete information about Gordon patches. Consider two strictly increasing sequences $\{u_0, \cdots, u_M\} \subset [0,1]$ and $\{v_0, \cdots, v_N\} \subset [0,1]$. In our case, we assume $u_0 = v_0 = 0$ and $u_M = v_N = 1$ because we are interested in a mapping from the unit square. Suppose that we have two families of curves $\mathbf{f}_j$, $\mathbf{g}_i$, $j = 0, \cdots, N$, $i = 0, \cdots, M$ (see Fig. 4.8(b)) satisfying the *compatibility condition*:

$$\mathbf{x}_{ij} := \mathbf{g}_i(v_j) = \mathbf{f}_j(u_i) \qquad \forall (i,j) \in \{0, \cdots, M\} \times \{0, \cdots, N\}. \tag{4.64}$$

A Gordon patch is a parametric surface $\mathbf{x}$ which is defined on $[0,1] \times [0,1]$ and



(a) (b) (c)

Figure 4.9: (a) Quadrilateral elements (b)The mesh $\mathcal{M}$ (c) The mesh $\mathcal{N}$ on the unit square.

which interpolates the curves $\mathbf{f}_j$ and $\mathbf{g}_i$:

$$\left\{ \begin{array}{lllll} \mathbf{x}(u, v_j) & = & \mathbf{f}_j(u) & \forall\, j = 0, \cdots, N & \forall\, u \in [0, 1], \\ \mathbf{x}(u_i, v) & = & \mathbf{g}_i(v) & \forall\, i = 0, \cdots, M & \forall\, v \in [0, 1]. \end{array} \right. \tag{4.65}$$

By choosing two sets of blending functions $(\varphi_i)_{i=0}^{M}$ and $(\psi_j)_{j=0}^{N}$ satisfying:

$$\varphi_i(u_k) = \delta_{ik} \qquad \psi_j(v_l) = \delta_{jl} \qquad i = 0, \cdots, M \quad \text{and} \quad j = 0, \cdots, N, \tag{4.66}$$

we define the Gordon patch as:

$$\mathbf{x}(u, v) := \sum_{i=0}^{M} \mathbf{g}_i(v)\varphi_i(u) + \sum_{j=0}^{N} \mathbf{f}_j(u)\psi_j(v) - \sum_{i=0}^{M}\sum_{j=0}^{N} \mathbf{x}_{ij}\varphi_i(u)\psi_j(v). \tag{4.67}$$

4.8 Generation of internal curves

We eliminate [54] the overspill phenomena by first inserting some curves $\mathbf{f}_j$ and $\mathbf{g}_i$ inside the four-sided domain $\mathcal{F}$ as illustrated in Fig. 4.11(b) and then by considering the Gordon patch $\mathbf{x}$ which interpolates those curves. The main difficulty in that process is the searching of the equations of the curves $\mathbf{f}_j$ and $\mathbf{g}_i$ to be inserted. Since we only know the equations of the curves delineating the boundary of the four-sided domain $\mathcal{F}$, we will discuss about an automatic method of generating the internal curves $\mathbf{f}_j$ and $\mathbf{g}_i$. In the following sections, we will describe our approach which consists of three stages:

- Find suitable points $\mathbf{x}_{ij}$ in the domain $\mathcal{F}$. These points will be the future intersections of the curves $\mathbf{f}_j$ and $\mathbf{g}_i$ as in relation (4.64).
- Determine the parameter values $u_0, \cdots, u_M$ and $v_0, \cdots, v_N$ which were introduced in relation (4.65).
- Interpolate the points $\mathbf{x}_{ij}$ with cubic Bézier-splines to obtain the curves $\mathbf{f}_j$ and $\mathbf{g}_i$.

4.8.1 Finding the internal points $\mathbf{x}_{ij}$

We are presenting two approaches of determining the internal points. The first one is based on the Dirichlet energy and the second one uses the Floater parametrization.

First approach (Dirichlet energy)

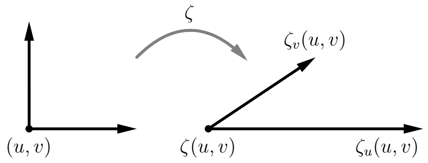


Figure 4.10: Distortion quantification

Before describing our method, let us observe that one can form quadrilaterals $[\mathbf{x}_{ij};\mathbf{x}_{i+1,j};\mathbf{x}_{i+1,j+1};\mathbf{x}_{i,j+1}]$ with the help of the vertices $\mathbf{x}_{ij}$ as seen in Fig. 4.9(a). Or analogously, if we insert a diagonal inside every quadrilateral then we have a triangular decomposition $\mathcal{M}$ as in Fig. 4.9(b). Our first approach finds a mapping which minimizes triangular distortions. Let us first introduce or recall the notion of Dirichlet energy in order to quantify the triangular distortions.

For any function ζ defined from the unit square to the plane, we define the Dirichlet energy to be

$$\mathcal{D}(\zeta) := 0.5 \int_{[0,1]^2} \left[\|\zeta_u\|^2 + \|\zeta_v\|^2\right] du\, dv. \tag{4.68}$$

Now we would like to identify the relationship between the Dirichlet energy and the shape distortion. The shape distortion of ζ can be characterized by two quantities:

$$\zeta_u \cdot \zeta_v \qquad \text{and} \qquad \|\zeta_u\|^2 - \|\zeta_v\|^2. \tag{4.69}$$

The first expression measures the angular deviation and the second one the scale nonuniformity (Fig. 4.10). We have the following inequalities

$$\|\zeta_u \times \zeta_v\| \leq \|\zeta_u\| \cdot \|\zeta_v\| \leq 0.5\left[\|\zeta_u\|^2 + \|\zeta_v\|^2\right]. \tag{4.70}$$

One can see [96] that those inequalities become equalities iff there is no shape distortion. Or equivalently, if the quantities in (4.69) are zero. Since the integral of the last term is the Dirichlet energy, if its minimal value is reached then there is no shape distortion. It is well-known [43] that if the image is a convex set and the mapping is harmonic then it must be one-to-one.

Let us fix a simple mesh $\mathcal{N}$ in the unit square as sketched in Fig. 4.9(c). The first approach consists therefore in finding the positions of $\mathbf{x}_{ij}$ which minimize the Dirichlet energy which is in our case

$$\mathcal{D}(\zeta) = 0.5 \sum_{T \in \mathcal{N}} \int_T \|\nabla \zeta\|^2 du\, dv. \tag{4.71}$$

Now let us show that minimizing the Dirichlet energy can easily be transformed into the solving of a linear system. In order to simplify the notations, we organize the nodes $\mathbf{x}_{ij}$ in lexicographic ordering as

$$\mathbf{X}_k = \mathbf{X}_{k(i,j)} = \mathbf{x}_{ij}. \tag{4.72}$$

Let us denote by p_k, $k \in \mathcal{R}$ the set of nodes of the mesh $\mathcal{N}$. The boundary nodes of the mesh $\mathcal{M}$ are samples from the boundary curves α, β, γ, δ:

$$A_r = \alpha\left(\frac{r}{N}\right) \qquad B_s = \beta\left(\frac{s}{M}\right) \qquad C_r = \gamma\left(\frac{r}{N}\right) \qquad D_s = \delta\left(\frac{s}{M}\right) \tag{4.73}$$

for $r = 0, ..., N$ and $s = 1, ..., M-1$.

The Dirichlet energy of the function ζ which is piecewise linear transformation of $\mathcal{N}$ into $\mathcal{M}$ is [97]

$$\mathcal{D}(\zeta) = 0.25 \sum_{[p_k, p_l] \in \mathcal{E}} (\cot \alpha_{kl} + \cot \beta_{kl}) \|\zeta(p_k) - \zeta(p_l)\|^2, \tag{4.74}$$

where $\mathcal{E}$ denotes the set of edges of the triangulation $\mathcal{N}$ and α_{kl}, β_{kl} the angles opposite to the edge $[p_k, p_l]$ (see Fig. 4.9(c)). We make the convention that $\cot(\alpha_{kl})$ or $cot(\beta_{kl})$ are zero if the edge $[p_k, p_l]$ is a boundary one and let us denote by $[x_k, y_k]$ the coordinates of $\mathbf{X}_k = \zeta(p_k)$ for all $k \in \mathcal{R}$. Since we are minimizing the Dirichlet energy, we are interested in their derivatives which are:

$$\frac{\partial}{\partial x_k} \mathcal{D}(\zeta) = 0.5 \sum_{p_l \in \nu(p_k)} (\cot \alpha_{kl} + \cot \beta_{kl})[x_k - x_l] \tag{4.75}$$

$$\frac{\partial}{\partial y_l} \mathcal{D}(\zeta) = 0.5 \sum_{p_l \in \nu(p_k)} (\cot \alpha_{kl} + \cot \beta_{kl})[y_k - y_l], \tag{4.76}$$

where $\nu(p_k)$ represents the set of neighbors of p_k. By splitting the set $\mathcal{R}$ into the set of boundary vertices $\mathcal{B}$ and the set of internal vertices $\mathcal{I}$ and by equating the above expressions to zero, we get a system of linear equation whose unknowns x_k and y_k for $k \in \mathcal{I}$ are exactly the coordinates of the internal points $\mathbf{x}_{ij}$.

Since the numbers $\cot \alpha_{kl}$ depend exclusively on the fixed mesh $\mathcal{N}$ on the unit square, one can deduce from (4.75) and (4.76) that the internal nodes $\mathbf{X}_k$, $k \in \mathcal{I}$ are linear functions of the boundary vertices $\mathbf{X}_k$, $k \in \mathcal{B}$. In particular, there are continuous functions σ_{ij} such that each internal point $\mathbf{x}_{ij}$ can be expressed as:

$$\mathbf{x}_{ij} = \sigma_{ij}(A_r, B_s, C_r, D_s). \tag{4.77}$$

Second approach (Floater parametrisation)

Now we are going to describe the second method of determining the internal points $\mathbf{x}_{ij}$. First, one generates a mesh $\mathcal{M}$ on the four-sided domain. Then, one uses the shape preserving parameterization method described by Michael Floater in [41] to find a mesh $\mathcal{N}$ on the unit square. One can therefore introduce a function κ which transforms the r-th node $\mathbf{u}_r$ of the mesh $\mathcal{N}$ to the r-th node of the mesh $\mathcal{M}$. A point $\mathbf{u}$ located inside a triangle $[\mathbf{u}_r, \mathbf{u}_s, \mathbf{u}_t]$ of $\mathcal{N}$ is then transformed into

$$\kappa(\mathbf{u}) := \lambda_r \kappa(\mathbf{u}_r) + \lambda_s \kappa(\mathbf{u}_s) + \lambda_t \kappa(\mathbf{u}_t), \tag{4.78}$$

where λ_r, λ_s, λ_t are the barycentric coordinates. In section 6.6.1, we will recall the Floater technique of obtaining the mesh $\mathcal{M}$. Finally, the points $\mathbf{x}_{ij}$ to be interpolated are defined to be $\kappa(i/M, j/N)$ for $i = 1, \cdots, M-1$ and $j = 1, \cdots, N-1$.

As opposed to the first approach, it is unknown which energy this second approach minimizes. Therefore, it is very difficult to make a rigorous analysis about its theoretical efficiency. It is yet worth mentioning that the second approach is very efficient in practice even for four-sided domains with extremely complicated boundary curves.

4.8.2 Generating the interpolating curves

In this section, we would like to determine a way to generate the curves $\mathbf{f}_j$ and $\mathbf{g}_i$ if the internal nodes $\mathbf{x}_{ij}$ have already been determined. Our approach should guarantee that the compatibility condition in (4.64) is fulfilled. The curve $\mathbf{g}_i$ (resp. $\mathbf{f}_j$) should interpolate the points $\mathbf{x}_{ik}$ with $k = 0, 1, \cdots, N$ (resp. $\mathbf{x}_{lj}$ with $l = 0, 1, \cdots, M$).

By introducing $u_i := i/N$ and $v_j := j/M$, the internal curves are represented as cubic Bézier-Splines:

$$\mathbf{g}_i(v) = \sum_{k=0}^{3} \mathbf{g}_{ik} B_k^3((v - v_{j-1})/(v_j - v_{j-1})) \quad \forall\, v \in [v_{j-1}, v_j] \tag{4.79}$$

$$\mathbf{f}_j(u) = \sum_{l=0}^{3} \mathbf{f}_{jl} B_l^3((u - u_{i-1})/(u_i - u_{i-1})) \quad \forall\, u \in [u_{i-1}, u_i]. \tag{4.80}$$

In order to determine the unknown control points $\{\mathbf{g}_{ik}\}$, $\{\mathbf{f}_{jl}\}$ in (4.79) and (4.80), one solves some linear system which depends only on $\mathbf{x}_{ij}$, u_i and v_j and which is diagonal dominant (see [68] for details). Because of that linearity, the coefficients $\mathbf{g}_{kl}$ and $\mathbf{f}_{kl}$ are continuous functions of $\mathbf{x}_{ij}$. That is, there are some consitnous functions μ_{kl} and ν_{kl} such that:

$$\mathbf{g}_{kl} = \mu_{kl}(\mathbf{x}_{ij}) \qquad \mathbf{f}_{kl} = \nu_{kl}(\mathbf{x}_{ij}). \tag{4.81}$$

Since the curves $\mathbf{g}_i$ and $\mathbf{f}_j$ interpolate the points $\mathbf{x}_{ij}$, the compatibility relation in (4.64) are necessarily fulfilled.

4.9 Diffeomorphic Gordon and termination guarantee

In this section, we would like to specify how to choose the blending functions φ_i and ψ_j from equation (4.66). Our objective during the choice is that we can carry over the results about the Coons patch from the former sections to Gordon patches in a piecewise manner. Additionally, we are summarizing the former discussions in form of an algorithm. We will also theoretically discuss our proposed algorithm has a termination. The usual way [39] of generating the blending functions is to utilize the Lagrange polynomials. The support of such functions is therefore the whole interval $[0,1]$. That method and similar ones are called *elastic* in [56] because a perturbation of one curve $\mathbf{f}_j$ or $\mathbf{g}_i$ propagates to the whole domain. In other words, a local variation of a curve $\mathbf{f}_j$ or $\mathbf{g}_i$ modifies the whole Gordon patch as given by equation (4.67). For our purpose, we prefer to use a local method in which φ_i and ψ_j are defined by

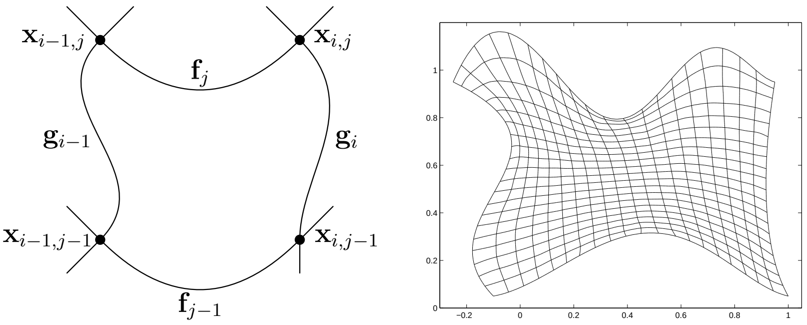
$$\varphi_i(u) = \begin{cases} \phi[(u-u_{i-1})/(u_i-u_{i-1})] & \text{if} \quad u \in [u_{i-1}, u_i] \\ 1-\phi[(u-u_i)/(u_{i+1}-u_i)] & \text{if} \quad u \in [u_i, u_{i+1}] \\ 0 & \text{otherwise,} \end{cases} \tag{4.82}$$

$$\psi_j(v) = \begin{cases} \phi[(v-v_{j-1})/(v_i-v_{j-1})] & \text{if} \quad v \in [v_{j-1}, v_j] \\ 1-\phi[(v-v_j)/(v_{j+1}-v_j)] & \text{if} \quad v \in [v_j, v_{j+1}] \\ 0 & \text{otherwise,} \end{cases} \tag{4.83}$$

where ϕ is a function defined on $[0,1]$ with $\phi(0)=0$ and $\phi(1)=1$. A candidate for such a function ϕ is given by F_1 from equation (4.28). In order to ensure important properties for φ_i and ψ_j such as global smoothness or local positivity, one can introduce additional conditions for ϕ like

- $\phi'(0)=\phi'(1)=0$,
- ϕ takes nonnegative values.

We can immediately observe that the above blending functions φ_i and ψ_j are compactly supported by $[u_{i-1}, u_{i+1}]$ and $[v_{j-1}, v_{j+1}]$ respectively. The resulting Gordon patch is known [56] to be *plastic* because every perturbation is kept local due to local support of the blending functions. Now, we want to analyze the Gordon patch in the cell $R_{ij} := [u_{i-1}, u_i] \times [v_{j-1}, v_j]$. One can check that in R_{ij} the Gordon patch, which has (4.82) and (4.83) as blending functions and which interpolates the internal curves in (4.79), (4.80), coincides with the Coons patch with respect to the boundary curves $\alpha^{\mathbf{loc}}$, $\beta^{\mathbf{loc}}$, $\gamma^{\mathbf{loc}}$, $\delta^{\mathbf{loc}}$ defined for $u, v \in [0,1]$



(a) (b)

Figure 4.11: (a) Image of R_{ij} by a Gordon patch $\mathbf{x}$ (b) An example of a Gordon patch.

by:

$$\begin{array}{ll} \alpha^{\mathbf{loc}}(u) := \mathbf{f}_{j-1}(u(u_i - u_{i-1}) + u_{i-1}) & \beta^{\mathbf{loc}}(v) := \mathbf{g}_i(v(v_j - v_{j-1}) + v_{j-1}) \\ \gamma^{\mathbf{loc}}(u) := \mathbf{f}_j(u(u_i - u_{i-1}) + u_{i-1}) & \delta^{\mathbf{loc}}(v) := \mathbf{g}_{i-1}(v(v_j - v_{j-1}) + v_{j-1}). \end{array}$$

Effectively, if we use the settings of cubic Bézier-splines as in (4.79) and (4.80), then we have the following control points for the local boundary curves $\alpha_{\mathbf{loc}}$, $\beta_{\mathbf{loc}}$, $\gamma_{\mathbf{loc}}$, $\delta_{\mathbf{loc}}$:

$$\begin{array}{llllllllllll} \alpha_0^{\mathrm{loc}} & = & \mathbf{f}_{j-1,0} & \alpha_1^{\mathrm{loc}} & = & \mathbf{f}_{j-1,1} & \alpha_2^{\mathrm{loc}} & = & \mathbf{f}_{j-1,2} & \alpha_3^{\mathrm{loc}} & = & \mathbf{f}_{j-1,3} \\ \beta_0^{\mathrm{loc}} & = & \mathbf{f}_{j-1,3} & \beta_1^{\mathrm{loc}} & = & \mathbf{g}_{i1} & \beta_2^{\mathrm{loc}} & = & \mathbf{g}_{i2} & \beta_3^{\mathrm{loc}} & = & \mathbf{f}_{j3} \\ \gamma_0^{\mathrm{loc}} & = & \mathbf{f}_{j0} & \gamma_1^{\mathrm{loc}} & = & \mathbf{f}_{j1} & \gamma_2^{\mathrm{loc}} & = & \mathbf{f}_{j2} & \gamma_3^{\mathrm{loc}} & = & \mathbf{f}_{j3} \\ \delta_0^{\mathrm{loc}} & = & \mathbf{f}_{j-1,0} & \delta_1^{\mathrm{loc}} & = & \mathbf{g}_{i-1,1} & \delta_2^{\mathrm{loc}} & = & \mathbf{g}_{i-1,2} & \delta_3^{\mathrm{loc}} & = & \mathbf{f}_{j0}. \end{array} \tag{4.84}$$

Most of the results that we had for the Coons patch in the former sections can be locally carried over to the Gordon patches. For instance, we would like to present an immediate corollary of Theorem 8.

Corollary 4 Consider the following three conditions:

(G1) There exists some $F > 0$ such that for $k = 0, \cdots, 3$

$$\begin{array}{lll} \|(g_{ik} - g_{i-1,k}) + \phi_k(\mathbf{x}_{i-1,j} - \mathbf{x}_{ij} + \mathbf{x}_{i,j-1} - \mathbf{x}_{i-1,j-1}) + (\mathbf{x}_{i-1,j-1} - \mathbf{x}_{i,j-1})\| & \leq & F/\rho \\ \|(f_{jk} - f_{j-1,k}) + \phi_k(\mathbf{x}_{i,j-1} - \mathbf{x}_{ij} + \mathbf{x}_{i-1,j} - \mathbf{x}_{i-1,j-1}) + (\mathbf{x}_{i-1,j-1} - \mathbf{x}_{i-1,j})\| & \leq & F/\rho \end{array}$$

where $\rho := \sup_{t \in [0,1]} |\phi'(t)|$.

(G2) There exists some $\kappa > 0$ such that for all $u \in [u_{i-1}, u_i]$, $v \in [v_{j-1}, v_j]$, $\zeta, \chi \in [0,1]$

$$\det(K_{u,\zeta}, L_{v,\chi}) \geq \kappa \qquad \text{for}$$

$$K_{u,\zeta} := (u_i - u_{i-1})[(1-\zeta)\mathbf{f}'_{j-1}(u) + \zeta\mathbf{f}'_j(u)] \tag{4.85}$$

$$L_{v,\chi} := (v_j - v_{j-1})[(1-\chi)\mathbf{g}'_{i-1}(v) + \chi\mathbf{g}'_i(v)] \tag{4.86}$$

(G3) There is some M such that for all $u \in [u_{i-1}, u_i]$, $v \in [v_{j-1}, v_j]$, $\zeta, \chi \in [0,1]$

$$\|K_{u,\zeta}\| \leq M \quad \text{and} \quad \|L_{v,\chi}\| \leq M. \tag{4.87}$$

Under these conditions the Gordon patch is a diffeomorphism in R_{ij} if

$$2MF + F^2 < \kappa. \tag{4.88}$$

We propose the following algorithm for generating diffeomorphism(s) with the help of transfinite interpolation.

Algorithm

step 1 : Generate the Coons patch $\mathbf{x}_C$ and use the former theorems to verify the diffeomorphism.

step 2 : If $\mathbf{x}_C$ is not a diffeomorphism, insert some curves $\mathbf{f}_j$, $\mathbf{g}_i$ and generate the Gordon patch $\mathbf{x}_G$. Verify if $\mathbf{x}_G$ is a diffeomorphism.

step 3 : If $\mathbf{x}_G$ is not a diffeomorphism, insert more curves and update the corresponding Gordon patch $\mathbf{x}_G$. Repeat this third step until a prescribed maximal number of curves is reached. If $\mathbf{x}_G$ is a diffeomorphism, abort the loop and terminate the algorithm.

step 4 : If $\mathbf{x}_G$ is not yet a diffeomorphism, then subdivide the domain $\mathcal{F}$ bounded by $[\alpha, \beta, \gamma, \delta]$ into a few four-sided patches $\mathcal{F}_i$ bounded by curves $[\alpha^i, \beta^i, \gamma^i, \delta^i]$. Then, go to step 1 and apply the algorithm to each $[\alpha^i, \beta^i, \gamma^i, \delta^i]$.

In Fig. 4.12(a) and Fig. 4.12(b), we can see subdivisions of four-sided domains into subregions having curved boundaries. The displayed subdivisions are not always possible for any four-sided domain $\mathcal{F}$ because there could be intersections between a curve and an internal straight edge. In chapter 3, we have elaborately described a method of decomposing a domain having a curved boundary into four-sided domains. In particular, we have described approaches of treating boundary interferences in which the curved boundary intersects an internal edge. As a consequence, we do not need to give much detail here about the realization of the subdivision in step 4.

In the next descriptions, we would like to analyze why the above algorithm has a termination. The following discussion is simple but we include it for the sake of completeness. Suppose we have a four-sided patch which has A, B, C and D as corners. The corner angles made by the quadrilateral composed of these vertices are θ_A, θ_B, θ_C and θ_D as in Fig. 4.12(c). We use the following angular assumption

$$0 < \theta_A < \pi, \qquad 0 < \theta_B < \pi, \qquad 0 < \theta_C < \pi, \qquad 0 < \theta_D < \pi. \tag{4.89}$$

Let us introduce the following auxiliary control points

$$\begin{array}{rcll} \overline{\alpha}_r & := & \lambda_r B + (1-\lambda_r)A & \overline{\beta}_r := \lambda_r C + (1-\lambda_r)B \\ \overline{\gamma}_r & := & \lambda_r C + (1-\lambda_r)D & \overline{\delta}_r := \lambda_r D + (1-\lambda_r)A, \end{array} \tag{4.90}$$

where λ_r denotes r/n. One can observe that those control points define four simple straight lines.

Let us introduce the points $\overline{\mathbf{x}}_{ij}$ which are the solutions of (4.75) and (4.76) if the control points of the four boundary curves are given by (4.90). Consequently, we have the following local control points of the cubic Bézier-spline:

$$\overline{\mathbf{f}}_{j0} = \overline{\mathbf{x}}_{i-1,j} \quad \overline{\mathbf{f}}_{j1} = \frac{1}{3}\overline{\mathbf{x}}_{ij} + \frac{2}{3}\overline{\mathbf{x}}_{i-1,j} \quad \overline{\mathbf{f}}_{j2} = \frac{2}{3}\overline{\mathbf{x}}_{ij} + \frac{1}{3}\overline{\mathbf{x}}_{i-1,j} \quad \overline{\mathbf{f}}_{j3} = \overline{\mathbf{x}}_{ij} \tag{4.91}$$

$$\overline{\mathbf{g}}_{i0} = \overline{\mathbf{x}}_{i,j-1} \quad \overline{\mathbf{g}}_{i1} = \frac{1}{3}\overline{\mathbf{x}}_{ij} + \frac{2}{3}\overline{\mathbf{x}}_{i,j-1} \quad \overline{\mathbf{g}}_{i2} = \frac{2}{3}\overline{\mathbf{x}}_{ij} + \frac{1}{3}\overline{\mathbf{x}}_{i,j-1} \quad \overline{\mathbf{g}}_{i3} = \overline{\mathbf{x}}_{ij} \tag{4.92}$$

By using the formula in (4.84), one can define the corresponding local control points $\overline{\alpha}_r^{\mathbf{loc}}$, $\overline{\beta}_r^{\mathbf{loc}}$, $\overline{\gamma}_r^{\mathbf{loc}}$, $\overline{\delta}_r^{\mathbf{loc}}$ with respect to the cubic Bézier splines in (4.91) and (4.92). Let us define the local corners

$$\mathbf{a} := \overline{\mathbf{x}}_{i-1,j-1} \qquad \mathbf{b} := \overline{\mathbf{x}}_{i,j-1} \qquad \mathbf{c} := \overline{\mathbf{x}}_{ij} \qquad \mathbf{d} := \overline{\mathbf{x}}_{i-1,j} \tag{4.93}$$

and let the corresponding corner angles be $\theta_A^{\mathbf{loc}}$, $\theta_B^{\mathbf{loc}}$, $\theta_C^{\mathbf{loc}}$, $\theta_D^{\mathbf{loc}}$. Because of angular relation (4.89), we have also

$$0 < \theta_A^{\mathbf{loc}} < \pi, \qquad 0 < \theta_B^{\mathbf{loc}} < \pi, \qquad 0 < \theta_C^{\mathbf{loc}} < \pi, \qquad 0 < \theta_D^{\mathbf{loc}} < \pi. \tag{4.94}$$

The partial derivatives of the Coons patch which is composed of the Bézier curves having $\overline{\alpha}_r^{\mathbf{loc}}$, $\overline{\beta}_r^{\mathbf{loc}}$, $\overline{\gamma}_r^{\mathbf{loc}}$, $\overline{\delta}_r^{\mathbf{loc}}$ as control points are

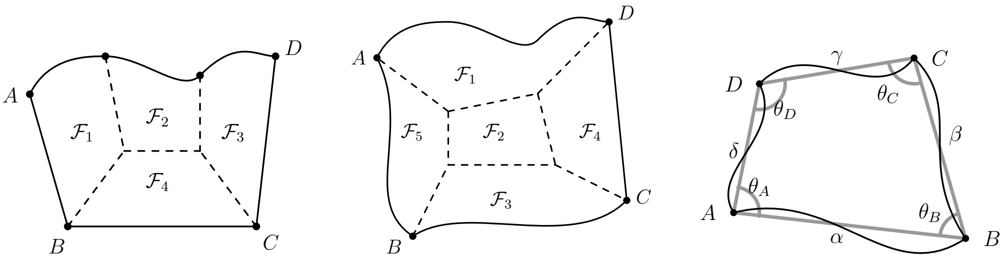
$$\overline{\mathbf{x}}_u = (1-\chi)\overrightarrow{\mathbf{ab}} + \chi\overrightarrow{\mathbf{dc}} \tag{4.95}$$

$$\overline{\mathbf{x}}_v = (1-\zeta)\overrightarrow{\mathbf{ad}} + \zeta\overrightarrow{\mathbf{bc}} \quad \text{with} \tag{4.96}$$

$$\chi(u,v) = \phi(v) + \phi'(u)(v - \phi(v)) \qquad \zeta(u,v) = \phi(u) + \phi'(v)(u - \phi(u)) \tag{4.97}$$

and ϕ is the function used in (4.82) and (4.83). We have therefore

$$\begin{array}{rcl} [\overline{\mathbf{x}}_u, \overline{\mathbf{x}}_v] & = & (1-\chi)(1-\zeta)[\overrightarrow{\mathbf{ab}}, \overrightarrow{\mathbf{ad}}] + (1-\chi)\zeta[\overrightarrow{\mathbf{ab}}, \overrightarrow{\mathbf{bc}}] + \\ & & \chi(1-\zeta)[\overrightarrow{\mathbf{dc}}, \overrightarrow{\mathbf{ad}}] + \chi\zeta[\overrightarrow{\mathbf{dc}}, \overrightarrow{\mathbf{bc}}]. \end{array} \tag{4.98}$$



(a) (b) (c)

Figure 4.12: (a),(b) Subdividing a four-sided domain (c)Termination guarantee

Because of the angular condition (4.94),

$$\min\Big\{[\overrightarrow{\mathbf{ab}},\overrightarrow{\mathbf{ad}}],[\overrightarrow{\mathbf{ab}},\overrightarrow{\mathbf{bc}}],[\overrightarrow{\mathbf{dc}},\overrightarrow{\mathbf{ad}}],[\overrightarrow{\mathbf{dc}},\overrightarrow{\mathbf{bc}}]\Big\} > 0. \tag{4.99}$$

Therefore, if we choose the function ϕ of (4.82) and (4.83) as in Remark 13, then we get $[\overline{\mathbf{x}}_u, \overline{\mathbf{x}}_v] > 0$. Let us denote by ν the maximal difference $|\alpha_r^{\mathrm{loc}} - \overline{\alpha}_r^{\mathrm{loc}}|$, $|\beta_r^{\mathrm{loc}} - \overline{\beta}_r^{\mathrm{loc}}|$, $|\gamma_r^{\mathrm{loc}} - \overline{\gamma}_r^{\mathrm{loc}}|$, $|\delta_r^{\mathrm{loc}} - \overline{\delta}_r^{\mathrm{loc}}|$. A few computation reveals that

$$[\mathbf{x}_u, \mathbf{x}_v] = [\overline{\mathbf{x}}_u, \overline{\mathbf{x}}_v] + \mathcal{O}(\nu). \tag{4.100}$$

Because of the continuity relations (4.77) and (4.81), ν tends to zero as the global difference

$$\mu := \max\{|\alpha_r - \overline{\alpha}_r|, |\beta_r - \overline{\beta}_r|, |\gamma_r - \overline{\gamma}_r|, |\delta_r - \overline{\delta}_r|\} \tag{4.101}$$

tends to zero. Therefore, $[\mathbf{x}_u, \mathbf{x}_v]$ must be strictly positive for all μ smaller than some μ_0 because σ is strictly positive.

Remark 16 As a result, the above description shows that if the deviations of the bounding curves from line segments are small enough, then the Gordon patch must be a diffeomorphism. That simple deduction was only done in order to guarantee that the algorithm terminates but the situations are much better in practice as seen in the next numerical results. That is especially true if the geometries come from mechanical objects. In this chapter, we have only treated *planar* Coons patches. *Spatial* trimmed surfaces are provided in IGES format in the following way. An initial parametric function ψ is given from a rectangular D parameter domain to the space. Afterward, a planar trimmed domain $\mathcal{D}$ is defined inside D. The eventual spatial trimmed surface is then the image of $\mathcal{D}$ by ψ. In general the parametric function ψ is supposed to be a diffeomorphism. Hence, the final diffeomorphism in case of spatial trimmed surface is the composition of a Gordon patch and the function ψ.

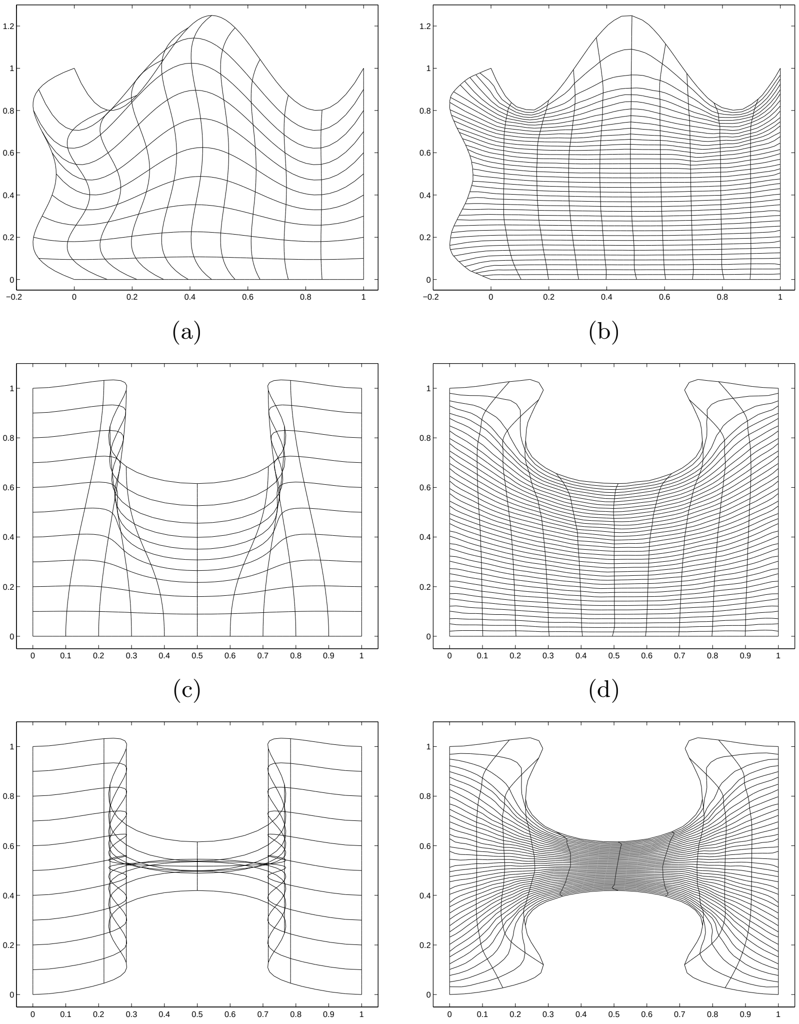
Tasks	CPU time
Searching for the gridpoints	4.306 sec
Finding the internal curves	0.080 sec
Evaluation at 100 positions	0.090 sec

Table 4.2: Runtimes for parameterization

4.10 Practical results

The first numerical test whose results can be found in Fig. 4.13 consists in finding the mappings for three different four-sided regions. In our investigation, we try to compare the Coons map and the results of the algorithm in section 4.9. We can see clearly that the Coons patches do not give diffeomorphisms in the three cases. We can observe terrible overspill phenomena especially in the third case where the internal domain becomes very tight. In the Gordon cases, the overspill phenomena could be completely eliminated.

As a second numerical test, we want to investigate the runtimes of the algorithm that we described for Gordon patches. Toward that end, we generate a Gordon patch for the domain in Fig. 4.11(b). In Table 4.2, we find the time needed to generate the gridpoints $\mathbf{x}_{ij}$, and the time for finding the internal curves $\{\mathbf{f}_j\}$ and $\{\mathbf{g}_i\}$. We are also interested in the computational cost of evaluating the resulting Gordon patch. In the last row of Table 4.2, we report the time needed to evaluate the Gordon patch at 100 parameter values $(u_k, v_k) \in [0,1] \times [0,1]$. One notes that determining the internal curves needs only to be done once and the values of the control points $\{\mathbf{g}_{ik}\}$ and $\{\mathbf{f}_{jl}\}$ in equations (4.79) and (4.80) can be stored for any future purpose. If we want to use the geometries for subsequent integral equation solvers, we can store those control points together with the information which describes the input geometric components.



(a) (b)

(c) (d)

(e) (f)

Figure 4.13: Nondiffeomorphic Coons patches (left side) and diffeomorphic Gordon patches (right side)

Chapter 5

MESH GENERATION

Abstract: We shall describe a mesh generation technique on a closed surface composed of a few parametric surfaces. The edge size function is a fundamental entity in order to be able to apply the process of generalized Delaunay triangulation with respect to the first fundamental form. Unfortunately, the edge size function is not known a-priori in general. We describe an approach which invokes the Laplace-Beltrami operator to determine it. Since the edge size function is required to be harmonic, the second objective of this chapter is to propose a method to determine piecewise linear approximations of the curved boundaries such that the variation of the edge lengths is smooth. We illustrate our approach by triangulating the surfaces of some CAD objects which come directly from IGES files. Numerical data demonstrate the quality of the resulting meshes with regard to harmonicity.

5.1 Introduction

Some numerical solvers [3, 61] of integral equations require that the input geometric information is represented as a mesh. As a consequence, we address the problem of creating a mesh [58, 41, 85] on a surface Γ which is composed of parametric surfaces $\mathbf{S}_i$ defined on parameter domains $\mathbf{D}_i \subset \mathbf{R}^2$. Additionally, we want the mesh to be composed of nicely shaped triangles. That is, the lengths of the three edges of each triangle should be proportional. For that matter, the ideal case would be the generation of a mesh of which all triangles are equilateral. In general, that ideal case is impossible to obtain because the discretizations of the boundary curves are not necessarily uniform. As a consequence, we aim at obtaining triangles which are as equilateral as possible. Thus, the variation of the edge lengths should be very smooth. That objective is obtained if we impose that the edge size is a harmonic function. In other words, we require that the edge size function (denoted by ρ throughout the chapter) vanishes by applying the Laplace-Beltrami operator. In order to generate the mesh, we choose the Delaunay technique for two reasons. First, it has a well-known nice property that generates a triangulation which maximizes the smallest angles. Additionally, the Delaunay triangulation can be used to control the size of the edges by splitting all edges which have lengths exceeding the ideal edge sizes.

The notion of equilateral triangles in the plane can be generalized in the context of parametric surfaces by using the first fundamental form. In fact, the usual length between two points $\mathbf{a}, \mathbf{b} \in \mathbf{D}_i$ and the angle at an apex $\mathbf{a}$ inside a triangle $[\mathbf{a}, \mathbf{b}, \mathbf{c}] \subset \mathbf{D}_i$ can be generalized by using a symmetric positive definite matrix M as follows

$$d_M(\mathbf{a}, \mathbf{b}) = \overrightarrow{\mathbf{ab}}^T M \, \overrightarrow{\mathbf{ab}}, \qquad \theta_M = \arccos \frac{\overrightarrow{\mathbf{ab}}^T M \, \overrightarrow{\mathbf{ac}}}{d_M(\mathbf{a}, \mathbf{b}) \cdot d_M(\mathbf{a}, \mathbf{c})}. \tag{5.1}$$

The problem of mesh generation in $\mathbf{S}_i$ amounts to applying the planar Delaunay triangulation on the parameter domain $\mathbf{D}_i$ by using the above generalized distance and angle with respect to the matrix which is provided by the first fundamental form. The generation of a mesh by means of the Delaunay technique with respect to the first fundamental form [13] requires the knowledge of the edge size function which is unfortunately unknown a-priori. The first purpose of the chapter is to describe a methodology to determine the edge size function ρ. Since the edge size function is known on the boundary, the treatment of the Laplace-Beltrami problem becomes therefore a boundary value problem which we propose to solve by means of the finite element method [15, 22].

In the next two sections, we will introduce various important definitions and we will outline our approach. In section 5.4, we will detail the meshing of a single parametric surface by using generalization of Delaunay triangulation [35]. Afterwards, we will describe the numerical resolution of the Laplace-Beltrami problem. We will apply it to the mesh generation of a composite trimmed parametric surface in section 5.9 where we will describe a method to approximate the curved boundaries by piecewise linear curves. For that matter, an approach invoking the use of a graph is utilized. Section 5.10 is written for those readers who are interested in theoretical background of the mesh generation approach. At the end of the chapter, we will report some benchmarks of CAD objects which are taken from IGES files. We will investigate numerically the quality of the resulting meshes with respect to their harmonicity.

5.2 Definitions and problem setting

Before describing our method of mesh generation, let us introduce a few definitions. Note that a thorough understanding of all the definitions is not an absolute necessity to understand the rest of the chapter.

A mesh $\mathbf{M}_h$ is a set of triangles $T_k \subset \mathbf{R}^d$ ($d = 2, 3$) such that for every two different triangles T_k, $T_l \in \mathbf{M}_h$ we have the following.

- Either $T_k \cap T_l = \emptyset$,
- or T_k and T_l share a node,
- or T_k and T_l share a complete edge.

A standard condition that is required in numerical solvers is that the smallest angle $\alpha_{\min}(T)$ inside each triangle $T \in \mathbf{M}_h$ satisfies

$$\alpha_{\min}(T) \geq \alpha_0, \tag{5.2}$$

where $\alpha_0 > 0$ is some prescribed threshold.

In our context, if d is 2 (resp. 3), then we will call $\mathbf{M}_h$ a 2D (resp. 3D) mesh. The above points imply in particular that there are no hanging nodes. That is, no node of $\mathbf{M}_h$ lies strictly in the interior of an edge. For a node A in a mesh $\mathbf{M}_h$, we will denote its valence by $\eta(A)$, i.e. the number of edges which are incident upon A. The set of nodes which are the endpoints of edges incident upon A, and which are different from A, will be denoted by $\nu(A)$.

Suppose we have a closed surface Γ that is composed of n parametric surfaces $\{\mathbf{S}_k\}_{k=1}^{n}$ such that each $\mathbf{S}_k$ is given as the image of a domain $\mathbf{D}_k \subset \mathbf{R}^2$ by the following function

$$\mathbf{x}_k : (u_1, u_2) \in \mathbf{R}^2 \longrightarrow (x_{k,1}(u_1,u_2), x_{k,2}(u_1,u_2), x_{k,3}(u_1,u_2)) \in \mathbf{R}^3 \tag{5.3}$$

which is supposed to be bijective and sufficiently smooth. Each domain $\mathbf{D}_k$ is a multiply connected region in $\mathbf{R}^2$ that is delineated by an external boundary and some internal boundaries which are composite curves having no double points. The surfaces $\mathbf{S}_k$ will be referred to as the patches of the whole surface Γ. Every patch of the surface Γ is bounded by a list of curves $\mathbf{C}_k$ which we will henceforth call '*curved*' edges. In the usual B-rep scheme, such curves are simply called edges. We use the expression curved edge in order to differentiate it from an edge of a mesh and from the edge of a graph which we will introduce subsequently. When no confusion is possible, we will drop the index k the sequel.

Let us introduce the following definitions:

$$g_{ij}(\mathbf{x}) := < \frac{\partial \mathbf{x}}{\partial u_i}, \frac{\partial \mathbf{x}}{\partial u_j} > \qquad i, j \in \{1, 2\}, \tag{5.4}$$

so that if we have $\mathbf{v} = a_1 \frac{\partial \mathbf{x}}{\partial u_1} + a_2 \frac{\partial \mathbf{x}}{\partial u_2}$ and $\mathbf{w} = b_1 \frac{\partial \mathbf{x}}{\partial u_1} + b_2 \frac{\partial \mathbf{x}}{\partial u_2}$ then

$$< \mathbf{v}, \mathbf{w} > = \sum_{i,j=1}^{2} a_i b_j g_{ij}(\mathbf{x}). \tag{5.5}$$

We will need the matrix $I(\mathbf{x}) := [g_{ij}(\mathbf{x})]$ which represents the first fundamental form [71, 87] in our mesh generation technique.

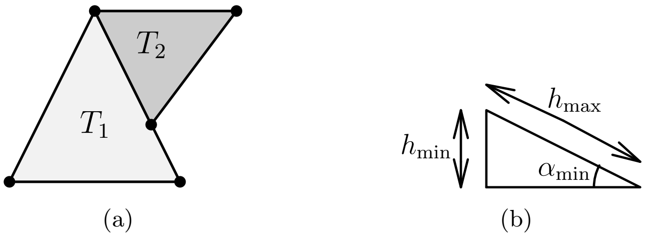


Figure 5.1: (a)Inadmissible mesh (b) Edge lengths

In this chapter, our objective is to generate a 3D mesh $\mathbf{M}_h$ which approximates Γ such that all nodes of $\mathbf{M}_h$ are located on the surface Γ. Additionally, the angular condition in (5.2) should be fulfilled. Note that if the lengths of the three edges in any triangle $T \in \mathbf{M}_h$ are proportional, then condition (5.2) follows. In other words, if we have

$$h_{\max}(T) \approx h_{\min}(T), \tag{5.6}$$

where $h_{\max}(T)$ and $h_{\min}(T)$ are the lengths of the longest and shortest edges of $T \in \mathbf{M}_h$ respectively then (5.2) holds as illustrated in Fig. 5.1(b). Beside shape quality, one of the most difficult things to achieve is to ensure that there are no hanging nodes at the interfaces of the surfaces $\mathbf{S}_k$.

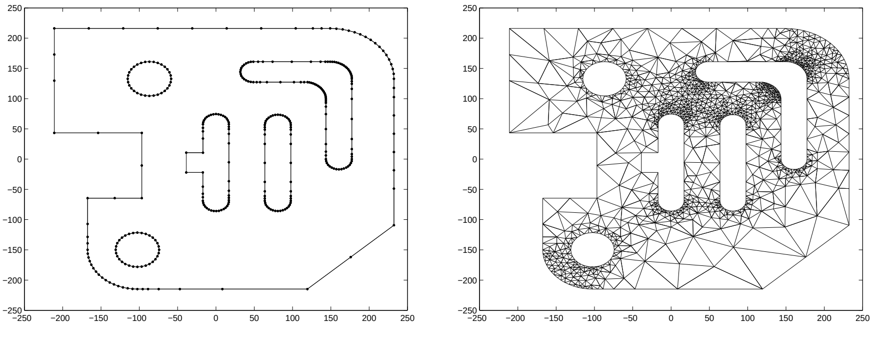
5.3 Motivation for the planar case

In this section, we want to treat briefly the mesh generation problem in the planar case (see Fig. 5.2) that should provide both motivation and intuitive ideas which facilitate the description of the general case of parametric surfaces. For that matter, we want to triangulate a planar multiply connected domain Ω_h with polygonal (external and possibly internal) boundaries P_h.

Observe that the edge sizes of the polygon P_h are usually nonuniform (see Fig. 5.2(a)). That is usually caused by adaptive discretization of some original curved boundaries P according to some error criteria.

The most straightforward method of obtaining (5.6) is to try to have a mesh such that all elements are equilateral triangles. That is not possible if the discretization of the bounding curves is nonuniform. As a result, we will try to obtain triangles which are as equilateral as we can do. Such triangles have in general very nice shapes [10, 11]. In order to obtain a few number of triangles while keeping their good quality shape, the edge sizes should have a small variation. That is, there are no two neighboring edges whose respective lengths are very disproportional. Let us introduce the edge size function

$$\rho : \Omega_h \subset \mathbf{R}^2 \longrightarrow \mathbf{R}\,, \tag{5.7}$$



(a) (b)

Figure 5.2: Planar case: (a)boundary nodes (b)internal nodes

which is supposed to take only positive values. If this function is explicitly known, then a way to obtain the mesh is to start from a very coarse mesh and to apply Delaunay node insertion (see [29, 13, 90]) in the middle of every edge $[\mathbf{a}, \mathbf{b}]$ whose length exceeds the value of ρ at the midnode of $[\mathbf{a}, \mathbf{b}]$. Unfortunately, the value of ρ is not known in practice. If the geometry of Ω_h is simple, then one can do some little computation to guess some reasonable values of ρ. That guessing approach could become difficult when the domain Ω_h becomes highly nonconvex. That usually happens in the following situations.

- The interior boundaries are placed in nonuniform positions,
- The difference between the lengths of the shortest and the longest boundary edges is very large,
- The initial curved boundary P has many constituents.

We face therefore the problem of determining the edge size function ρ which is only known at the boundaries $\partial\Omega_h = P_h$. We consider therefore the following boundary value problem:

$$\Delta\rho := \frac{\partial^2\rho}{\partial u_1^2} + \frac{\partial^2\rho}{\partial u_2^2} = 0 \qquad \text{in} \quad \Omega_h, \tag{5.8}$$

with the nonhomogeneous Dirichlet boundary condition given by the edge sizes at the boundary. That means the edge size function is required to be harmonic. A very good feature of a harmonic function ρ is that it satisfies the mean value property. That is, $\rho(\mathbf{a})$ is the same as the average of the values of ρ in a circle centered at $\mathbf{a} = (a_1, a_2)$:

$$\rho(a_1, a_2) = \frac{1}{2\pi}\int_0^{2\pi} \rho(a_1 + r\cos\theta, a_2 + r\sin\theta)d\theta. \tag{5.9}$$

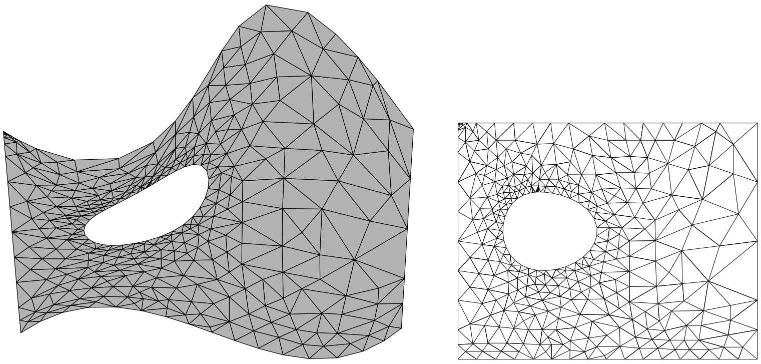

(a) (b)

Figure 5.3: (a)Mesh on a trimmed surface (b)the corresponding 2D mesh

An immediate consequence of this property is that the edge size function ρ has practically small variation. The Poisson problem (5.8) can be solved efficiently by means of the finite element method on a temporary mesh. In the following sections, we will first discuss how to mesh a single trimmed parametric surface. Before treating the parametric case, let us note the following generalization property. For two points **a** and **b** in the plane, we compute the distance with the help of

$$d(\mathbf{a},\mathbf{b}) = \overrightarrow{\mathbf{ab}}^T \overrightarrow{\mathbf{ab}}. \tag{5.10}$$

For a triangle $[\mathbf{a},\mathbf{b},\mathbf{c}] \subset \mathbf{R}^2$, the angle at the apex **a** can be expressed with

$$\theta = \arccos \frac{\overrightarrow{\mathbf{ab}}^T \overrightarrow{\mathbf{ac}}}{\|\mathbf{ab}\| \cdot \|\mathbf{ac}\|}. \tag{5.11}$$

For a given symmetric positive definite matrix M, those metric and angular relations can be generalized as

$$d_M(\mathbf{a},\mathbf{b}) = \overrightarrow{\mathbf{ab}}^T M \, \overrightarrow{\mathbf{ab}}, \qquad \theta_M = \arccos \frac{\overrightarrow{\mathbf{ab}}^T M \, \overrightarrow{\mathbf{ac}}}{d_M(\mathbf{a},\mathbf{b}) \cdot d_M(\mathbf{a},\mathbf{c})}. \tag{5.12}$$

Thus, the attempt to have well shaped triangles should be treated according to those generalized entities. We will utilize the Laplace-Beltrami operator which is the generalization of the Laplace operator for parametric surfaces. Afterwards, the mesh generation of a surface which is composed of multiple trimmed parametric surface will be described.

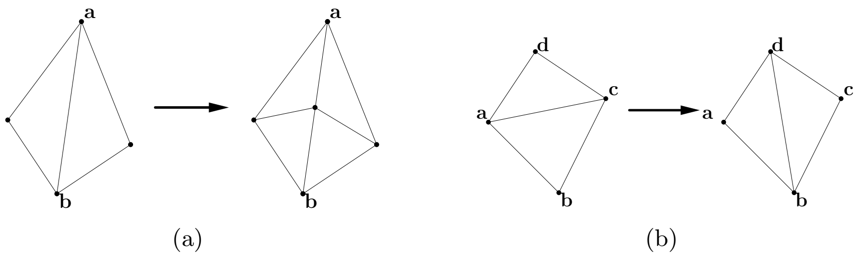


Figure 5.4: (a)Splitting (b)Flipping

5.4 Meshing using the first fundamental form

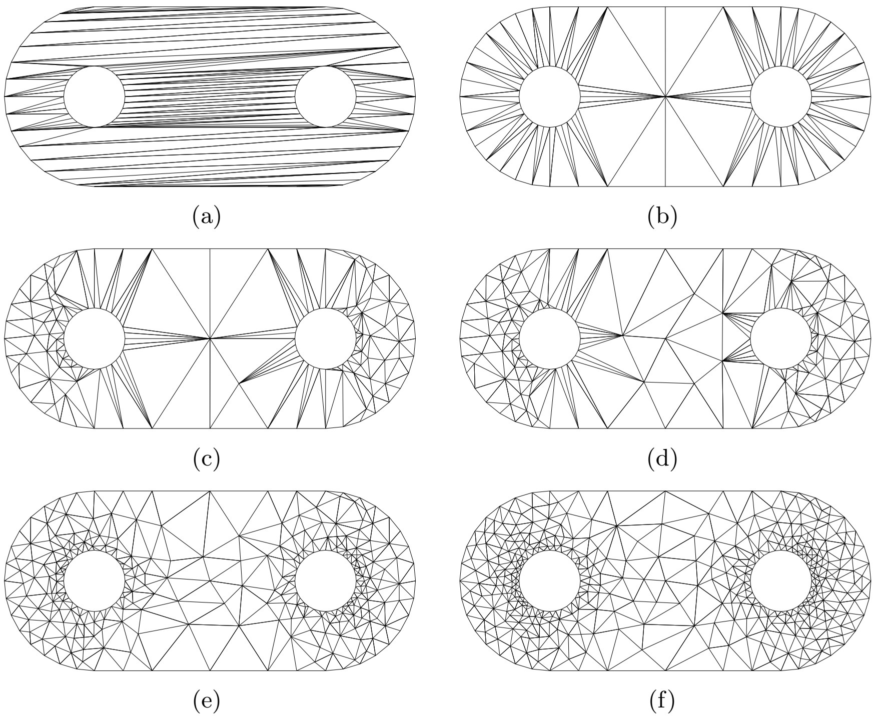
In this section, we suppose that a single parametric function $\mathbf{S}$ is provided by giving a smooth parametric function $\mathbf{x}$ defined on a parameter domain $\mathbf{D} \subset \mathbf{R}^2$. The approach in triangulating $\mathbf{S}$ is processed in two steps (see also [13, 51] for discussion about generation of anisotropic meshes). First, we generate a 2D mesh (see Fig. 5.3(b)) on the parameter domain $\mathbf{D}$ according to the first fundamental form. Afterwards, we lift the resulting 2D mesh to the parametric surface $\mathbf{S}$ by computing its image by $\mathbf{x}$ (see Fig. 5.3(a)). For that purpose, we start from a coarse 2D mesh of $\mathbf{D}$ and we use a generalized two dimensional Delaunay refinement which we want to describe briefly in this section. We will call an edge of a mesh in the parameter domain a *2D edge* and an edge in the lifted mesh a *3D edge*.

5.4.1 Splitting according to first fundamental form

Similarly to the planar case, we introduce an edge size function ρ which is defined now on the parametric surface $\rho : \mathbf{S} \longrightarrow \mathbf{R}^+$. By composing ρ with the parameterization $\mathbf{x}$ of $\mathbf{S}$,we have another function $\tilde{\rho} := \rho \circ \mathbf{x}$ which we will call henceforth "parameter edge size function" because it is defined for all $\mathbf{u} = (u, v)$ in the parameter domain. Let us consider a 2D edge $[\mathbf{a}, \mathbf{b}] \subset \mathbf{D}$ and let us denote the first fundamental forms at $\mathbf{a}$ and $\mathbf{b}$ by $I_\mathbf{a}$ and $I_\mathbf{b}$ respectively. Further, we introduce the following average distance between $\mathbf{a}$ and $\mathbf{b}$

$$d_{\text{Riem}}(\mathbf{a}, \mathbf{b}) := \sqrt{\overrightarrow{\mathbf{ab}}^T T \overrightarrow{\mathbf{ab}}} \qquad T := 0.5(I_\mathbf{a} + I_\mathbf{b}). \tag{5.13}$$

We split the 2D edge $[\mathbf{a}, \mathbf{b}]$ (see Fig. 5.4(a)) if this average distance exceeds the value of the parameter edge size function $\tilde{\rho}$ at the midnode of $[\mathbf{a}, \mathbf{b}]$. Note that only internal edges are allowed to be split. As a consequence, no new boundary nodes are introduced during the refinement process.



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

Figure 5.5: Selected steps in mesh recursive refinement

5.4.2 Flipping according to first fundamental form

Let us consider the situation where the 2D edge $[\mathbf{a},\mathbf{c}]$ is shared by two triangles (see Fig. 5.4(b)) which form a convex quadrilateral. Denote by $I_{\mathbf{a}}$, $I_{\mathbf{b}}$, $I_{\mathbf{c}}$ and $I_{\mathbf{d}}$ the values of the first fundamental form when applied to the nodes $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ respectively. By defining T to be the average of these $I_{\mathbf{a}}$, $I_{\mathbf{b}}$, $I_{\mathbf{c}}$ and $I_{\mathbf{d}}$, we perform the generalized Delaunay edge flipping if the following generalized Delaunay angle criterion is met

$$\|\overrightarrow{\mathbf{bc}} \times \overrightarrow{\mathbf{ba}}\|(\overrightarrow{\mathbf{da}}^T T \overrightarrow{\mathbf{dc}}) < \|\overrightarrow{\mathbf{da}} \times \overrightarrow{\mathbf{dc}}\|(\overrightarrow{\mathbf{cb}}^T T \overrightarrow{\mathbf{ba}}). \qquad (5.14)$$

That is, we replace the two triangles $[\mathbf{a},\mathbf{c},\mathbf{d}]$ and $[\mathbf{a},\mathbf{b},\mathbf{c}]$ by $[\mathbf{a},\mathbf{b},\mathbf{d}]$ and $[\mathbf{b},\mathbf{c},\mathbf{d}]$ if we have the above inequality.

5.5 Initial coarse mesh

Many mesh operations in Computer Graphics start from a very fine mesh [50] which is repeatedly coarsened. In the opposite, for our mesh generation purpose, we start from a very coarse triangulation and we refine it according to the ideal mesh size function ρ. An illustration of the refinement process can be seen in

Fig. 5.5. Suppose we have a 2D domain $\mathbf{P}$ which may contain some holes and which has polygonal boundaries. In this section, we would like to describe the procedure of splitting $\mathbf{P}$ into a coarse triangulation whose vertices are located at the boundary of $\mathbf{P}$. The approach consits in chopping off a triangle from the polygon repeatedly. It is summarized in the following algorithm.

step 1 : Split the initial polygon into a set of simply connected polygons $\mathbf{P} = \bigcup_{i=1}^{N} \mathbf{P}^{(i)}$.
step 2 : For every simply connected polygon $\mathbf{P}^{(i)}$, do the following:
step 2.1 : Initialize its set of triangles as empty set $\mathcal{T}_h^{(i)} = \emptyset$.
step 2.2 : Find a triangle T which can be chopped off from $\mathbf{P}^{(i)}$.
step 2.3 : Update $P^{(i)} := P^{(i)} \setminus T$ and $\mathcal{T}_h^{(i)} = \mathcal{T}_h^{(i)} \cup T$.
step 3 : The triangulation of $\mathbf{P}$ is the union of all triangulations: $\mathcal{T}_h := \cup_i \mathcal{T}_h^{(i)}$.

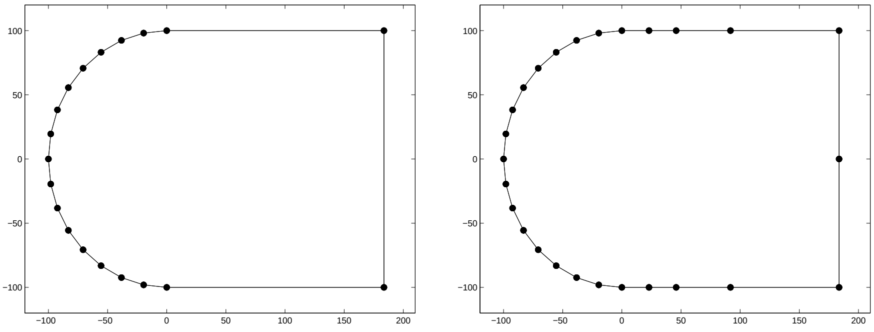
In section 5.10, we will give theoretical background which shows that this algorithm always give an acceptable triangulation.

Note that we can just as well use other triangulation methods such as advancing front techniques [52, 92] in order to obtain the coarse mesh. Another possible way of having a coarse triangulation of the domain is to consider a bounding box and inserting the vertices of the polygon one by one by applying Delaunay triangulation and we remove the triangles outside the domain afterwards [10]. Yet another procedure consists in splitting the polygon into monotone polygon [29] and then triangulating each monotone polygon by means of diagonal edge insertion [49].

5.6 Nonsmooth boundary approximation

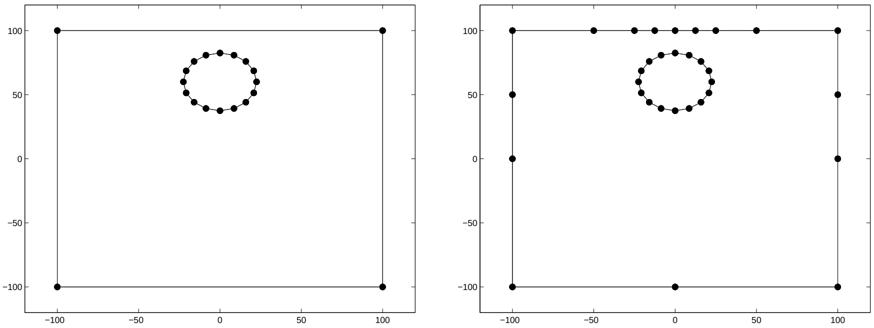
Let us suppose in this section that we have only one single trimmed patch $\mathbf{S}$. There are two usual ways of approximating the curved boundaries $\partial\mathbf{S}$ of $\mathbf{S}$ by piecewise linear curves. The first one is done by picking equi-spaced vertices on $\partial\mathbf{S}$. The second one consists in creating the piecewise linear curves so that very few vertices are needed to have a good approximation of the curved boundaries. The former method is not always desirable because it could yield a lot of vertices even in linear part of the boundary $\partial\mathbf{S}$. The drawback of the latter is that it could lead to nonsmooth edge size transition. In fact, we distinguish three kinds of nonsmoothness which we want to point out briefly now.

First kind: Two edges on the same curve which share one node have lengths which are not proportional at all (see Fig. 5.6(a)). This usually happens when the boundary $\partial\mathbf{S}$ is composed of linear and curved components. One needs many vertices to obtain an accurate approximation of the curved component whereas only very few vertices are required to have a very good approximation of the linear component as is observed in Fig. 5.6(a).



(a) (b)

Figure 5.6: (a)Nonsmoothness of first kind (b)After smoothing



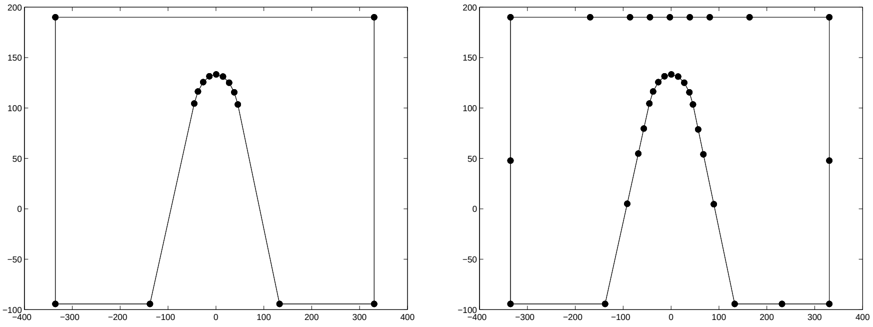
(a) (b)

Figure 5.7: (a)Nonsmoothness of second kind (b)After smoothing

Second kind: The second kind of nonsmoothness can only happen if the boundary $\partial\mathbf{S}$ is composed of several closed curves. In this case, one has one exterior boundary and one or several interior boundaries because the surface contains some holes. Nonsmoothness occurs if two curves are very close to one another but the respective edges have nonnegligible difference in length. For instance, that happens when one curve is circular and another one is rectangular as portrayed in Fig. 5.7(a).

Third kind: Nonsmoothness could also occur even if two nonproportional edges are not adjacent but they are spatially close to one another. That is exemplified in Fig. 5.8(a) where an edge on the curved boundary is close to the topmost edge but those two edges do not share any common node. In order to have smooth variation of the edge sizes, one has to insert more vertices in the linear top boundary as illustrated in Fig. 5.8(b).

Our method of discretization of curved boundaries is processed in two steps. First, we discretize each boundary components in which one uses as few nodes as possible to have a good accuracy. As a second step, one inserts additional nodes



(a) (b)

Figure 5.8: (a)Nonsmoothness of third kind (b)After smoothing

to have smooth edge length variation. We will describe this boundary smoothing process in the next sections where it will also be made clear that such a process should not be done patch by patch.

5.7 Determination of the edge size function

Let us consider a parametric surface $\mathbf{S}$ and a differentiable function $F : \mathbf{S} \longrightarrow \mathbf{R}$. The Laplace-Beltrami operator is defined by

$$\Delta_{\mathbf{S}} F = -\frac{1}{\sqrt{g}} \frac{\partial}{\partial u_j} \left(\sqrt{g} g_{ij} \frac{\partial F}{\partial u_i} \right) \tag{5.15}$$

in which we use Einstein notation in indexing and g is the determinant of I which we introduced in (5.4). The function F is said to be harmonic if

$$\Delta_{\mathbf{S}} F = 0. \tag{5.16}$$

It is known that this operator is independent of the chosen parameterization. To put it another way, the operator depends exclusively on the surface and not on the chosen local coordinates. In our approach the edge size function ρ which is defined on a parametric surface should be harmonic. We have therefore the following problem

$$\left\{ \begin{array}{rcll} -\Delta_S \rho & = & 0 & \text{in } \mathbf{S} \\ \rho & = & \rho_{\text{bound}} & \text{on } \partial\mathbf{S}\,. \end{array} \right. \tag{5.17}$$

We would like to sketch in this section how to numerically solve the boundary value problem (5.17) involving the Laplace-Beltrami equation. We approximate the function ρ by a function ρ_h by means of the finite element method [15, 22]. For that end, we take a temporary mesh $\mathbf{M}_h$ on $\mathbf{S}$ and we denote its boundary by $\partial\mathbf{M}_h$.

The values of ρ at the boundary which are denoted by ρ_{bound} are known because the piecewise linear boundary has already been determined from section 5.6. Since the function ρ represents the edge size, we can define its value at a boundary node A to be average of the lengths of the two incident boundary edges of A.

For a smooth function ϕ which takes value zero at the boundary we have

$$-\int_{\mathbf{S}} \Delta_{\mathbf{S}} \rho\phi = \int_{\mathbf{S}} < \nabla_{\mathbf{S}}\rho, \nabla_{\mathbf{S}}\phi >=: a(\rho,\phi)\,. \tag{5.18}$$

Let us define the following set of approximating linear space

$$V_h := \{f \in \mathbf{C}^0(\mathbf{M}_h)\ :\ f_{|T} \in \mathbf{P}_1\ \forall T \in \mathbf{M}_h\}\,, \tag{5.19}$$

where $\mathbf{C}^0(\mathbf{M}_h)$ denotes the space of functions which are globally continuous on $\mathbf{M}_h$ and $\mathbf{P}_1$ the space of linear polynomials. For a function g we define the set

$$V_h^g := \{f \in V_h :\ f = g \quad \text{on} \quad \partial\mathbf{M}_h\} \tag{5.20}$$

which is not in general a linear space (The sum of two functions in this set is not necessarily an element of this set if g in nonzero).

The approximated solution ρ_h will reside in the set $V_h^{\rho_{\mathrm{bound}}}$. In order to find ρ_h, we pick an arbitrary element $\tilde{\rho}$ of $V_h^{\rho_{\mathrm{bound}}}$ and define μ_h by setting

$$\rho_h = \tilde{\rho} + \mu_h\,. \tag{5.21}$$

The function ρ_h is therefore completely determined if we know the new unknown function μ_h which resides interestingly in V_h^0. Observe that V_h^0 is a linear space in which we choose a basis $\{\phi_i\}_{i\in I}$. As a consequence, the function μ_h is a linear combination of $\{\phi_i\}_{i\in I}$:

$$\mu_h = \sum_{i\in I} \mu_i\phi_i\,. \tag{5.22}$$

By introducing the following bilinear form $a_h(\cdot,\cdot)$ which approximates $a(\cdot,\cdot)$ of (5.18)

$$a_h(\psi,\phi) := \sum_{T\in\mathbf{M}_h} a_T(\psi,\phi) \qquad \text{with} \qquad a_T(\psi,\phi) :=< \nabla_T\psi, \nabla_T\phi >, \tag{5.23}$$

we have

$$a_h(\rho_h,\phi) = 0 \qquad\quad \forall\phi \in V_h^0 \tag{5.24}$$

or equivalently

$$a_h(\mu_h,\phi) = -a_h(\tilde{\rho},\phi) \qquad \forall\phi \in V_h^0\,. \tag{5.25}$$

Since ϕ_i builds a basis for V_h^0, this leads to a linear equation

$$\sum_{i\in I} a_h(\phi_i,\phi_j)\mu_i = -a_h(\tilde{\rho},\phi_j) \qquad \forall j \in I\,. \tag{5.26}$$

One can assemble the stiffness matrix $M_{ij} := a_h(\phi_i, \phi_j)$ and solve (5.26) for μ_i which yields the value of μ_h by using equation (5.22).

For every triangle T in $\mathbf{M}_h$ with internal angles α_1, α_2 and α_3, its contribution [9, 96] to the stiffness matrix M is

$$M_T = 0.5 \begin{bmatrix} \cot\alpha_2 + \cot\alpha_3 & -\cot\alpha_3 & -\cot\alpha_2 \\ -\cot\alpha_3 & \cot\alpha_1 + \cot\alpha_3 & -\cot\alpha_1 \\ -\cot\alpha_2 & -\cot\alpha_1 & \cot\alpha_1 + \cot\alpha_2 \end{bmatrix}. \qquad (5.27)$$

5.8 Meshing of a surface with multiple patches

In the previous sections, we have described the triangulation of a single trimmed surface. Now we are going to give some description of how to triangulate the surface Γ composed of the surfaces $\mathbf{S}_1, \cdots, \mathbf{S}_n$ (see Fig. 5.9). The different stages of the process are outlined in the following steps. First, we approximate the curved edges $\mathbf{C}_i$ by piecewise linear curves $\tilde{\mathbf{C}}_i$ in which we aim at both accuracy and smoothness (see Fig. 5.9(a)). Then, we map the 3D nodes of the relevant piecewise linear curves $\tilde{\mathbf{C}}_i$ back to the parameter domain $\mathbf{D}_k \subset \mathbf{R}^2$ for each patch $\mathbf{S}_k$. Afterwards, we generate a mesh $\mathbf{M}_k$ for each surface patch $\mathbf{S}_k$ by first generating a 2D-mesh on $\mathbf{D}_k$ according to some criteria and then by lifting the resulting 2D mesh to $\mathbf{S}_k$ as we have described in the former sections. Finally, we merge [51] the meshes $\mathbf{M}_1,\cdots$, $\mathbf{M}_n$ in order to have the final mesh. Note that nodes at the interface will surely align because we do not insert new boundary nodes in the triangulation of each $\mathbf{S}_k$. In the next section, we are going to describe how to accomplish the piecewise linear approximation of the curved edges in order to achieve smooth edge size variation.

5.9 Discretization of the curved boundaries

Let us first demonstrate that it is important to generate the piecewise linear approximation of the boundaries for the whole object (see Fig. 5.9). In other words, the inefficiency of performing that discretization by proceeding patch by patch can be explained by two reasons. First, the nonsmoothness that we have described before could also occur if two disproportional edges belong to two different patches. In order to illustrate that, let us consider again the object in Fig. 5.9 and let us observe the situation that happens in the interface between the cylindrical patch $\mathbf{P}_1$ and a planar patch $\mathbf{P}_2$ which is tangent to $\mathbf{P}_1$. One can readily notice that there are edges of $\mathbf{P}_2$ which do not belong to $\mathbf{P}_1$ (see Fig. 5.9(a)) but which must undergo some smoothing process in order that the resulting mesh (Fig. 5.9(b)) has smooth edge size transition in the neighborhood of the interface.

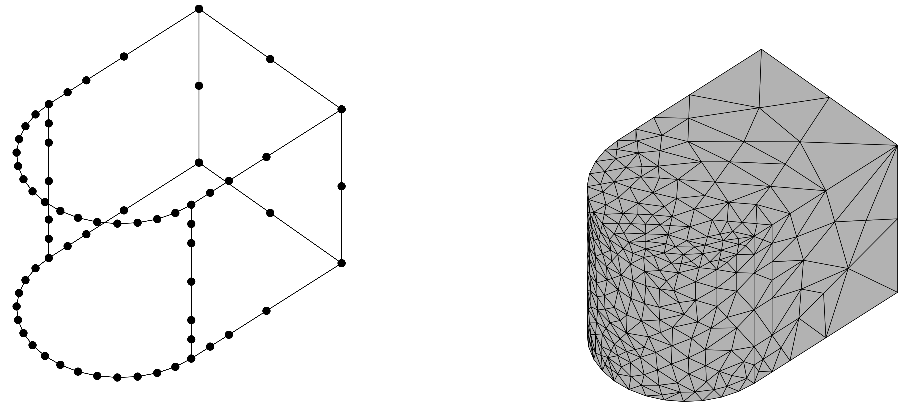

(a) (b)

Figure 5.9: (a)Boundary nodes (b)Surface mesh

The second reason for acting globally is about mesh merging. We generate a mesh $\mathbf{M}$ for the whole surface by first generating a mesh $\mathbf{M}_i$ for each patch and then by merging the meshes $\mathbf{M}_1,\cdots,\mathbf{M}_n$. In order that the merging process happens successfully, the nodes at the interface must align well. One approach of achieving that abutting of the boundary nodes is by using a postprocessing in which one inserts new nodes or one shifts some existing boundary nodes. We would like to avoid that approach because that could considerably damage the quality of the mesh. If we prescribe the boundary nodes by fixing their positions with the help of boundary discretization in advance then boundary nodes belonging to adjacent patches will surely align.

Now let us introduce our approach of achieving that global edge discretization with smooth variation.

Definition 16 A graph $\mathbf{G}$ is defined by giving a pair (V, E) in which V is the set of vertices and E is the set of graph edges. Each vertex $\mathbf{a} \in V$ has 3D coordinates and each graph edge $e \in E$ has two bounding vertices. We define the distance between two graph edges to be the Euclidean distance of their midpoints. For a positive number L, we define the L-neighborhood of a given graph edge $e \in E$ to be the set of graph edges $f \neq e$ of the graph $\mathbf{G}$ such that the distance between e and f is smaller than L. We will subsequently denote by $|e|$ the length of a graph edge e in V.

Remark 17 During the course of our algorithm, the coordinates of a vertex $\mathbf{a} \in E$ is the image of some curved edge $\mathbf{C}$, i.e.

$$\mathbf{a} = \mathbf{C}(t) \qquad \text{for some } t\,. \tag{5.28}$$

Definition 17 A smoothing factor is a number $\alpha \geq 2$ which is used to determine if an edge need to be refined. Let e be a graph edge in the graph $\mathbf{G}$ and let L be its length. We will say that e is nonsmooth if there is an edge f in the L-neighborhood of e such that

$$|e| \geq \alpha |f|. \tag{5.29}$$

Now we would like to describe briefly how to obtain the graph $\mathbf{G}$ of the boundaries of the surface $\Gamma = \mathbf{S}_1 \cup \cdots \cup \mathbf{S}_n$. The important steps are summarized in the following short description in which our objective is both accuracy of approximation and smoothness.

First, we initialize the graph $\mathbf{G}$ to have one graph-edge per curved edge of the surface $\mathbf{S}$. As a second step, we traverse the list of graph edges $e \in E$ and we suppose that e is bounded by two vertices $\mathbf{a}$ and $\mathbf{b}$ with $\mathbf{a} = \mathbf{C}(t_a)$ and $\mathbf{b} = \mathbf{C}(t_b)$. We compute the relative error according to the following formula:

$$err(e) = \frac{1}{\|\overrightarrow{\mathbf{ab}}\|} \left[\|\overrightarrow{\mathbf{ab}}\| - \mathbf{L}(\mathbf{a}, \mathbf{b}) \right] , \tag{5.30}$$

where $\mathbf{L}(\mathbf{a}, \mathbf{b})$ is the length of the curved edge $\mathbf{C}$ in the interval $[t_a, t_b]$. Afterwards, we split those graph edges e for which the error $err(e)$ is larger than some prescribed accuracy $\varepsilon > 0$. Then, we take a smoothing factor $\alpha > 0$ in order to find the list of nonsmooth edges $W \subset V$. Finally, we split every nonsmooth edge e of W by inserting a node between its endpoint.

5.10 Theoretical discussion

This section is devoted to those readers who are more interested in the theoretical aspect of the above description than in the practical numerical realization of the mesh generation from CAD data.

Basically, the mesh generation procedure consists in having an initial coarse mesh which is then refined recursively. Since splitting an edge by introducing an internal node and flipping an edge (Fig. 5.4) are two operations which are straightforward, the main critical points which we want to theoretically clarify are:

- In the initial coarse mesh generation of section 5.5, can every multiply connected polygon be triangulated by using only boundary nodes?
- Is the linear system in section 5.7 uniquely solvable?

In the next descriptions, we want to consider some theoretical background pertaining to those two items. First, let us show by the following theorem that every simply connected polygon has a triangulation having all nodes on the boundary.

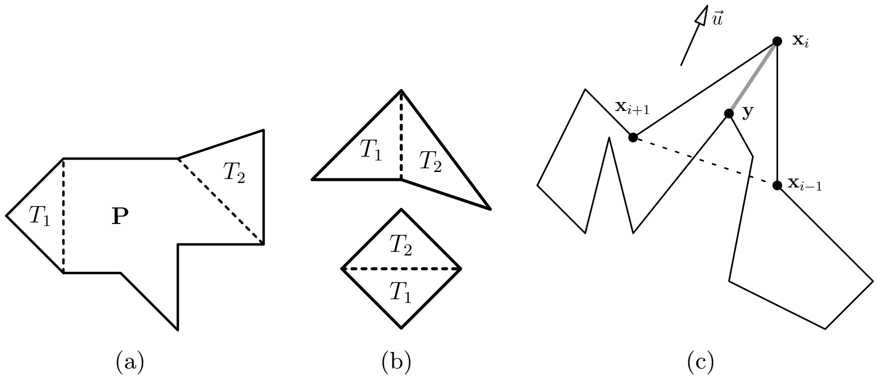


Figure 5.10: (a)Chopping two triangles off (b)Triangulating quadrilaterals (c)The triangle $T := [\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}]$ contains some nodes in its interior.

Theorem 12 (Meister) From every simply connected polygon $\mathbf{P}$, one may remove two triangles T_1 and T_2 by introducing internal cuts (Fig. 5.10(a)).

Proof:

We proceed by induction with respect to the number n of vertices of $\mathbf{P}$.

If $n = 4$ (thus $\mathbf{P}$ is a quadrilateral), then $\mathbf{P}$ can be split into exactly two triangles T_1 and T_2 as in Fig. 5.10(b). We suppose now that the claim is true for n. Let us consider a polygon $\mathbf{P}$ having $(n+1)$ vertices. Pick a vertex $\mathbf{x}_i$ of $\mathbf{P}$ such that its interior angle is smaller than π. The following two cases may occur to the triangle $T := [\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}]$.

Case 1: If T does not contain any node, we remove it from the polygon $\mathbf{P}$. Let us denote by $\tilde{\mathbf{P}}$ the remaining polygon which has n vertices. Thus, the first triangle that one can chop from $\mathbf{P}$ is $T_1 := T$. On account of the hypothesis of induction, we can chop off two triangles $\tilde{T}_1$ and $\tilde{T}_2$ from the polygon $\tilde{\mathbf{P}}$. The second triangle that can be removed is $T_2 := \tilde{T}_1$ or $T_2 := \tilde{T}_2$ whichever does not have $[\mathbf{x}_{i-1}, \mathbf{x}_{i+1}]$ as edge.

Case 2: Suppose now that T contains some node of $\mathbf{P}$ in its interior as graphically illustrated in Fig. 5.10(c). Denote by $\vec{u}$ the unit normal to $[\mathbf{x}_{i+1}, \mathbf{x}_{i-1}]$ such that

$$\det(\overrightarrow{\mathbf{x}_{i+1}\mathbf{x}}_{i-1}, \vec{u}) > 0. \tag{5.31}$$

Let $\mathbf{y}$ be the node inside T which maximizes $< \vec{u}, \overrightarrow{\mathbf{x}_{i+1}\mathbf{y}} >$.

The segment $[\mathbf{x}_i, \mathbf{y}]$ splits the polygon $\mathbf{P}$ into two polygons $\mathbf{P}^a$ and $\mathbf{P}^b$. Let us consider two subcases depending on the number of vertices of the polygon $\mathbf{P}^a$.

Subcase 2.1: If $\mathbf{P}^a$ is a triangle, we simply define $T_1 := \mathbf{P}^a$. Then, we apply the hypothesis of induction to the polygon $\mathbf{P}^b$ which yields two triangles $\tilde{T}_1$ and $\tilde{T}_2$. At least one of the triangles $\tilde{T}_1$ and $\tilde{T}_2$ does not have the segment $[\mathbf{x}_i, \mathbf{y}]$ as edge. Therefore, we choose that triangle as T_2.

Subcase 2.2: If $\mathbf{P}^a$ has more than three vertices, we apply the hypothesis of induction to both polygons $\mathbf{P}^a$ and $\mathbf{P}^b$ in order to obtain the triangles T_1^a, T_2^a, and T_1^b, T_2^b. We choose T_1 and T_2 from among those four triangles whichever are not incident upon the segment $[\mathbf{x}_i, \mathbf{y}]$.

$\square$

Corollary 5 A simply connected polygon can be triangulated by using only boundary nodes.

Proof

Suppose the polygon has n vertices. Chop triangles off $(n-2)$ times by applying the above theorem.

$\square$

For triangulation of multiply connected polygons, we need to split them first into several simply connected polygons $\mathbf{P}_i$ by introducing internal cuts. Afterwards, we apply the above Corollary to every polygon $\mathbf{P}_i$. In chapter 3, we have met a lot of discussions and proofs pertaining to cut insertions.

Now we would like to show the solvability of the equation involving the Laplace-Beltrami operator with the help of the following theorem.

Theorem 13 The linear system from relation (5.26) is uniquely solvable.

Proof

We will show that the matrix is invertible by considering its expression in one element. At the same time, we will show how to derive the formula in (5.27).

Consider a triangle $T = [A, B, C]$ of the mesh $\mathbf{M}_h$. Let $\vec{N}_3$ be the unit normal vector of T. Generate two unit vector vectors $\vec{N}_1$ and $\vec{N}_2$ perpendicular to $\vec{N}_3$ such that $(\vec{N}_1, \vec{N}_2, \vec{N}_3)$ is an orthonormal system which can be centered at A. Since the triangle T is located in the plane spanned by $(\vec{N}_1, \vec{N}_2)$, every point $\mathbf{x} \in T$ can be identified by $(v_1, v_2) \in \mathbf{R}^2$ such that

$$\overrightarrow{A\mathbf{x}} = v_1 \vec{N}_1 + v_2 \vec{N}_2. \tag{5.32}$$

Consider the triangle $t := [(0,0), (1,0), (0,1)]$ and let φ be the (Fig. 5.11) parametrization which transforms t into T:

$$\begin{bmatrix} \varphi_1(u_1, u_2) \\ \varphi_2(u_1, u_2) \end{bmatrix} := \begin{bmatrix} V_1 & W_1 \\ V_2 & W_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{5.33}$$

where (V_1, V_2) and (W_1, W_2) are the components of $W := \overrightarrow{AB}$ and $V := \overrightarrow{AC}$ in $(\vec{N}_1, \vec{N}_2)$. Denote by M the above matrix and let θ be the inverse of φ:

$$\begin{bmatrix} \theta_1(v_1, v_2) \\ \theta_2(v_1, v_2) \end{bmatrix} = \frac{1}{\det M} \begin{bmatrix} W_2 & -W_1 \\ -V_2 & V_1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \tag{5.34}$$

Let ψ be the linear polynomial which transforms $\psi(0,0)=\mu(A)$, $\psi(1,0)=\mu(B)$, $\psi(0,1)=\mu(C)$. Its exact expression is

$$\psi(u_1,u_2)=[\mu(B)-\mu(A)]u_1+[\mu(C)-\mu(A)]u_2+\mu(A). \qquad (5.35)$$

We want to compute the expression of $a_T(\cdot,\cdot)$ with the help of the variables (u_1,u_2). By introducing $a_{ij}:=\partial_{v_j}\theta_i$, the integrand for $a_T(\cdot,\cdot)$ involves:

$$\begin{aligned} I(u_1,u_2) &:= (a_{11}\partial_{u_1}\psi+a_{21}\partial_{u_2}\psi)^2+(a_{12}\partial_{u_1}\psi+a_{22}\partial_{u_2}\psi)^2 && (5.36)\\ &= (a_{11}^2+a_{12}^2)(\partial_{u_1}\psi)^2+2(a_{11}a_{21}+a_{22}a_{12})(\partial_{u_1}\psi\partial_{u_2}\psi)+ && (5.37)\\ &\quad (a_{21}^2+a_{22}^2)(\partial_{u_2}\psi)^2. && (5.38)\end{aligned}$$

Because of equation (5.34), we have the relations by using $D=\det M$

$$a_{11}=W_2/D \qquad a_{12}=-W_1/D \qquad a_{21}=-V_2/D \qquad a_{22}=V_1/D. \qquad (5.39)$$

We have therefore

$$\begin{aligned} I(u_1,u_2) &= \tfrac{1}{D^2}\{(W_2^2+W_1^2)(\partial_{u_1}\psi)^2-2(W_2V_2+W_1V_1)\partial_{u_1}\psi\partial_{u_2}\psi+\\ &\quad (V_2^2+V_1^2)(\partial_{u_2}\psi)^2\}\\ &= \tfrac{1}{D^2}\{\|W\|^2(\partial_{u_1}\psi)^2-2<W,V>\partial_{u_1}\psi\partial_{u_2}\psi+\|V\|^2(\partial_{u_2}\psi)^2\}\\ &= \tfrac{1}{D^2}\{[\mu(B)-\mu(A)]^2<W,W-V>+[\mu(B)-\mu(C)]^2<V,W>\\ &\quad [\mu(C)-\mu(A)]^2<V,V-W>\}.\end{aligned}$$

By using the relations

$$\cos\alpha=\frac{<V,W>}{\|V\|\cdot\|W\|} \qquad \sin\alpha=\frac{\det M}{\|V\|\cdot\|W\|}, \qquad (5.40)$$

we obtain

$$I(u_1,u_2)=\frac{1}{D}\{[\mu(B)-\mu(C)]^2\cot\alpha+[\mu(C)-\mu(A)]^2\cot\beta+[\mu(B)-\mu(A)]^2\cot\gamma\} \qquad (5.41)$$

We have therefore

$$\begin{aligned} a_T(\mu,\mu) &= \int_t I(u_1,u_2)(\det M)du_1du_2 && (5.42)\\ &= 0.5\{[\mu(B)-\mu(C)]^2\cot\alpha+[\mu(C)-\mu(A)]^2\cot\beta+ && (5.43)\\ &\quad [\mu(B)-\mu(A)]^2\cot\gamma\}. && (5.44)\end{aligned}$$

By denoting $\Psi(\mu):=a_T(\mu,\mu)$, the system in (5.27) can be obtained by

$$a_T(\mu,\nu)=0.5[\Psi(\mu+\nu)-\Psi(\mu)-\Psi(\nu)]. \qquad (5.45)$$

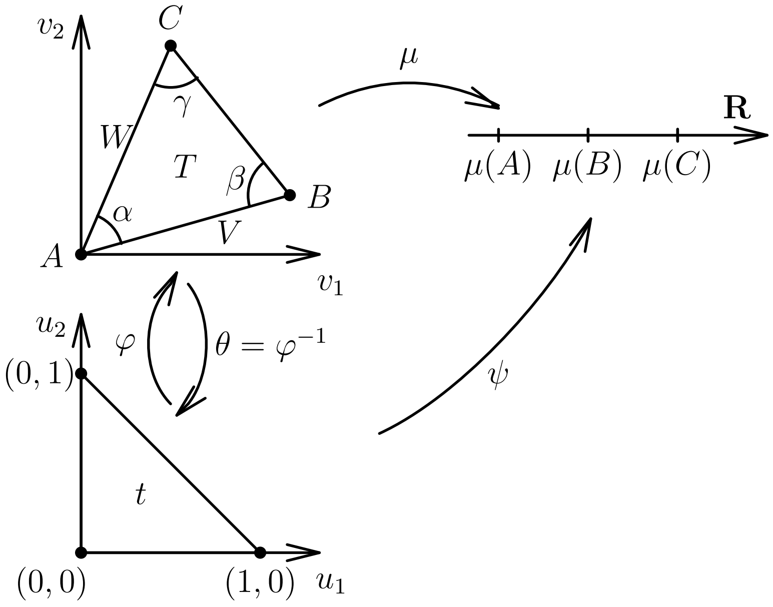


Figure 5.11: Computing the cotan formula

Denote by $\tilde{W} := \frac{1}{D}(\partial_{u_1}\psi)W$ and $\tilde{V} := \frac{1}{D}(\partial_{u_2}\psi)V$, we have

$$I(u_1,u_2) = \|\tilde{W}\|^2 - 2 < \tilde{W}, \tilde{V} > + \|\tilde{V}\|^2. \tag{5.46}$$

By the relation of Cauchy-Schwarz, $I(u_1,u_2) = 0$ iff $\tilde{W} = \lambda\tilde{V}$ for some $\lambda \in \mathbf{R}$. The fact that W and V are not parallel implies that $I(u_1,u_2) = 0$ if $\partial_{u_1}\psi = \partial_{u_2}\psi = 0$. That is to say $\psi = C^{te}$ which means $\mu(A) = \mu(B) = \mu(C)$. Since $\mu \in V_h^0$ is globally continuous and it takes zero values at the boundary as introduced in relation (5.20), we have $\mu = 0$. That amounts to saying that $a_h(\mu,\mu) > 0$ if the function $\mu \in V_h^0$ and $\mu \neq 0$. The form $a_h(\cdot,\cdot)$ is thus symmetric positive definite (s.p.d). Therefore, the linear system from relation (5.26) is invertible and it can be solved numerically with the help of numerical solvers for s.p.d systems like Conjugate Gradient.

□

5.11 Numerical results

We have implemented the approaches that we have described in this chapter in order to investigate their numerical performance. The program has been written in C/C++ and it uses OpenGL [66] and GLUT [76] to display graphical output which allows the user to interact with it. The CAD objects whose surfaces have to be triangulated are given as input. They are stored in IGES files [119] which have been generated by some CAD systems [124]. We have developed automatic routines to load the different parts of a given IGES file and to evaluate the constituting entities [39, 68]. The majority of the geometric patches in an IGES format are represented in form of NURBS surfaces with NURBS curve as trimming curves (see[95]).

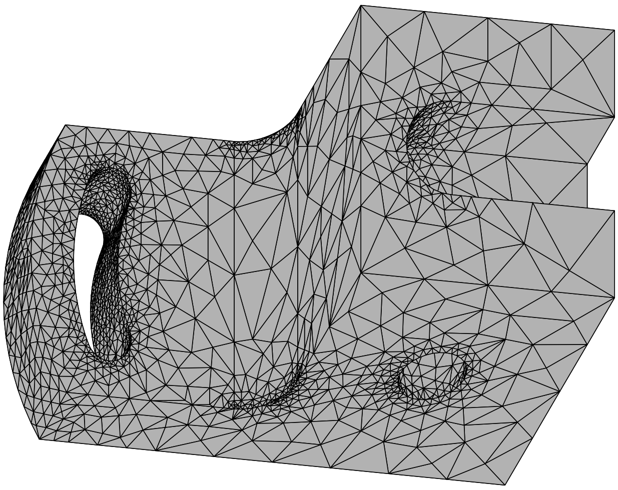


Figure 5.12: Mesh with 5424 elements and 2710 gridpoints

We consider in our numerical tests three CAD objects which have respectively 19, 38 and 22 patches. We used the aforementioned method to generate meshes on their surfaces. The resulting meshes, having respectively 5424, 32310, and 7314 elements, are portrayed in Fig. 5.12, 5.13 , 5.15. Before we describe those meshes, let us introduce the way we analyze the quality of our results with regard to our above-discussed approach of mesh generation. In our numerical tests, we would like to investigate the harmonicity of the meshes which we want to define now. For any considered node $A \in \mathbf{R}^3$ of a mesh $\mathbf{M}_h$, we define

$$\rho(A) := \frac{1}{\eta(A)} \sum_{B \in \nu(A)} \|\overrightarrow{AB}\| \tag{5.47}$$

to be the average edge length (see section 5.2 for the definition of the valence $\eta(A)$ and the set of incident nodes $\nu(A)$). Now we would like to introduce the discrete counterpart of the right hand side of the relation (5.9). For that end we define $r(A)$ to be the length of the shortest edge incident to a node A and we let s_i be the intersection of the i-th edge incident to A and the sphere centered at A with radius $r(A)$ (see Fig. 5.14 for an illustration). We define the mean value

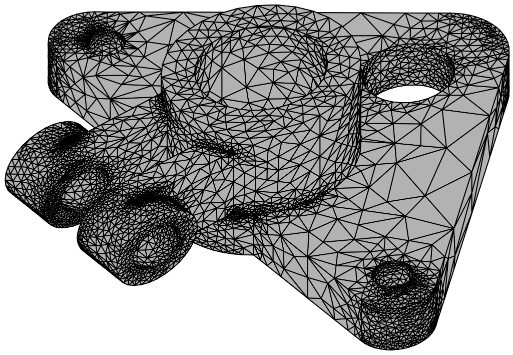


Figure 5.13: Mesh with 32310 elements and 16145 gridpoints

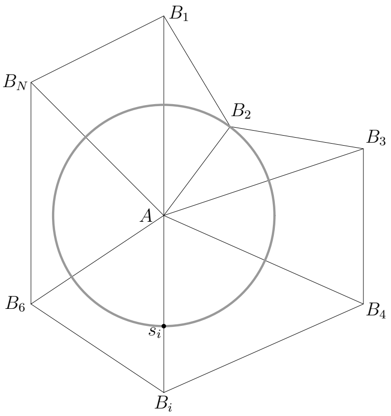


Figure 5.14: Edge length

$\rho_{\text{mean}}(A)$ to be

$$\rho_{\text{mean}}(A) := \frac{1}{\eta(A)} \sum_{B \in \nu(A)} \rho(s_i) \tag{5.48}$$

in which $\rho(s_i)$ is the following convex combination of $\rho(A)$ and $\rho(B_i)$

$$\rho(s_i) := \frac{\|\overrightarrow{As_i}\|}{\|\overrightarrow{AB}\|}\rho(B_i) + \left(1 + \frac{\|\overrightarrow{As_i}\|}{\|\overrightarrow{AB}\|}\right)\rho(A).$$

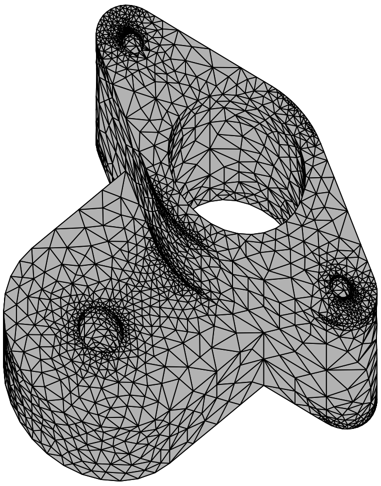


Figure 5.15: Mesh with 7314 elements and 3651 gridpoints

	Average harmonicity	Smallest harmonicity	Largest harmonicity
mesh1	.99727	.74156	1.25858
mesh2	.99496	.66758	1.43521
mesh3	.99627	.75712	1.29101

Table 5.1: Harmonicity of the three meshes

We have a discrete mean value property if

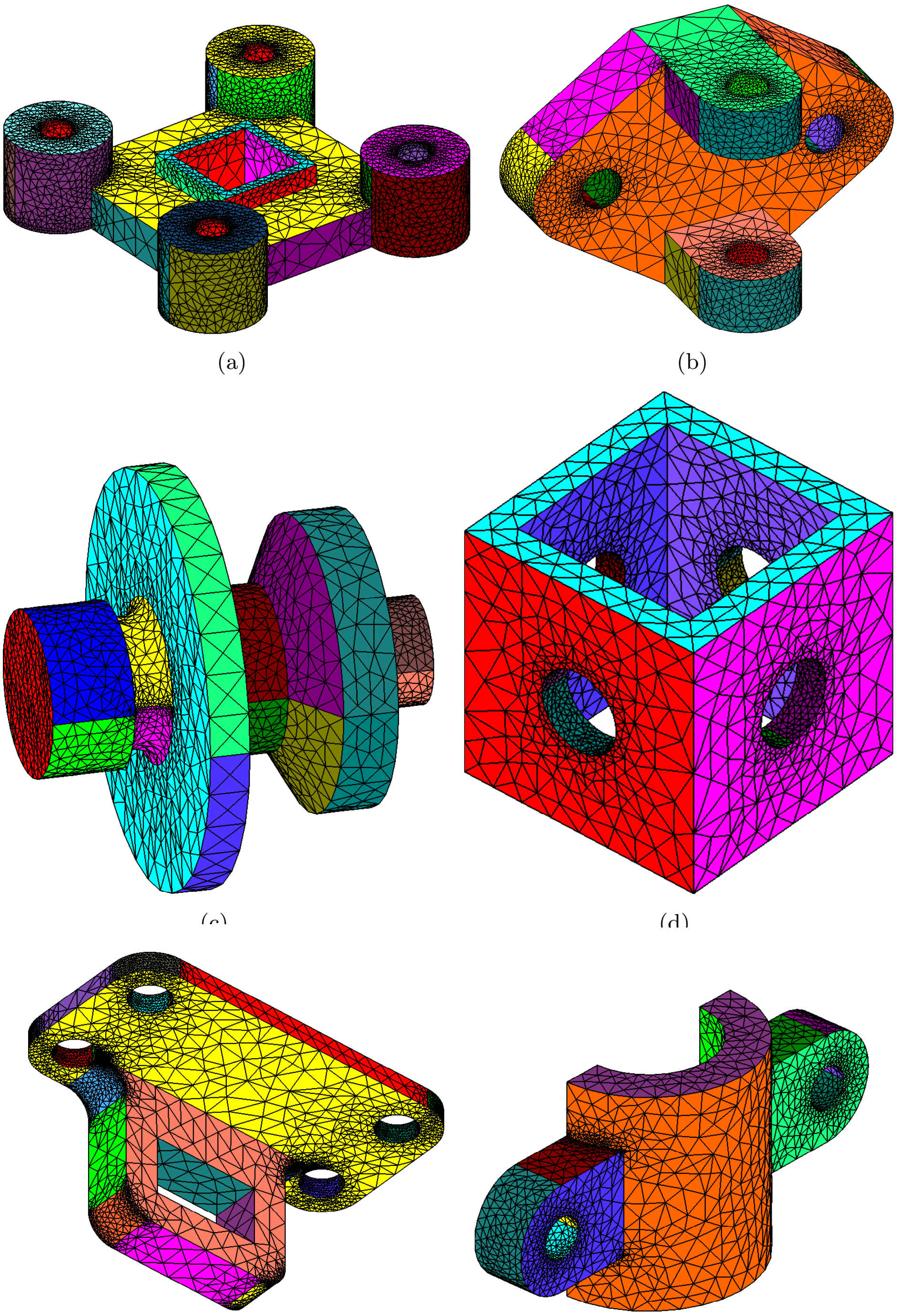
$$\rho(A) = \rho_{\text{mean}}(A). \tag{5.49}$$

We define the harmonicity of a node A to be the ratio

$$\xi(A) := \rho(A)/\rho_{\text{mean}}(A). \tag{5.50}$$

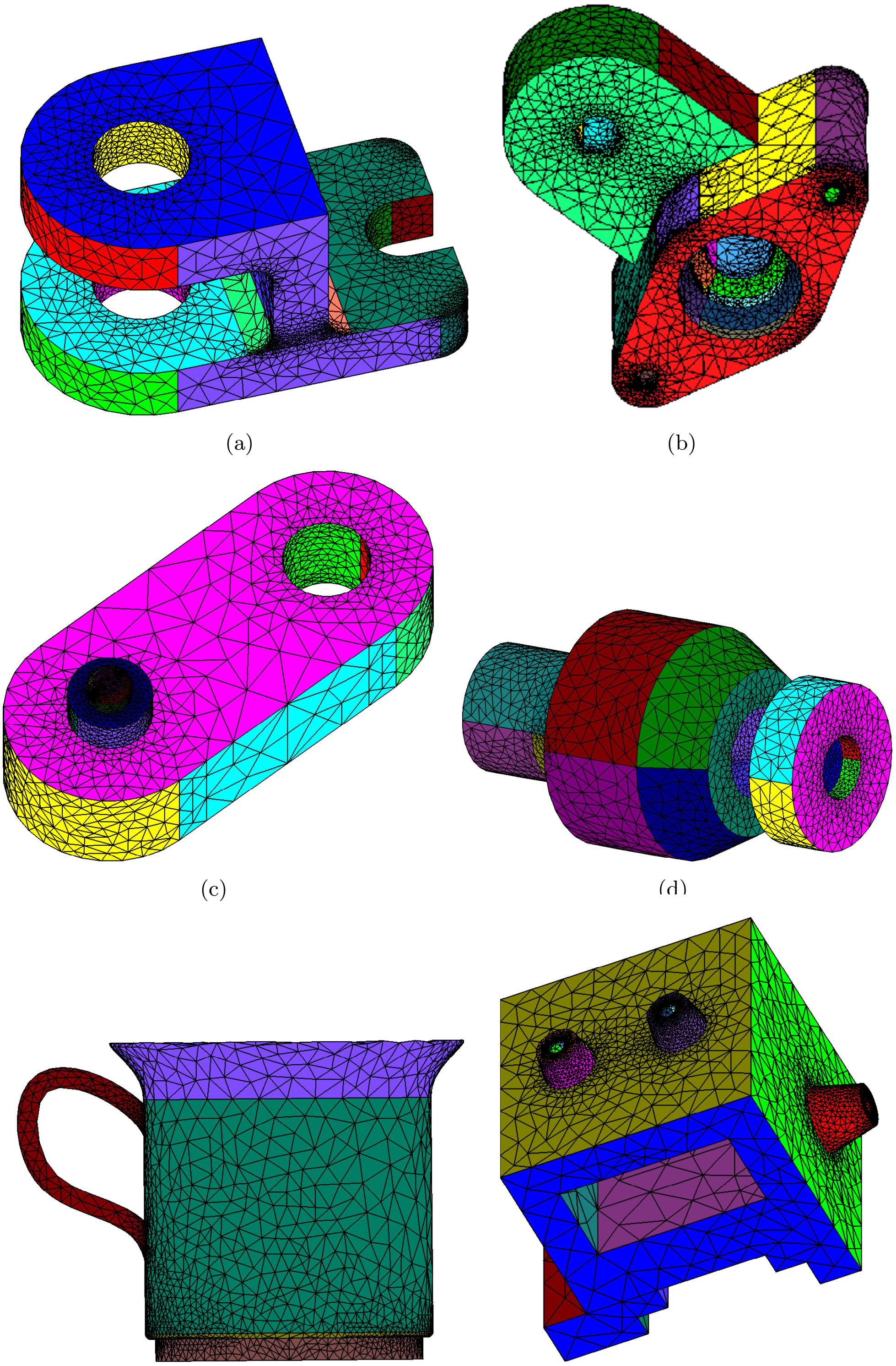
If the value of the harmonicity approaches the unity, the discrete mean value property is valid. We have computed the average harmonicity of the three meshes and the results can be found in the next table. As one can see, the mesh sizes in our tests nearly verify the discrete mean value property.

For more results where each trimmed surface is painted with its own color in order to facilitate the view of the mesh matching in the interfaces, we can see Fig. 5.16 and Fig. 5.17.



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

Figure 5.16: Results from CAD data in IGES files



(a) (b)

(c) (d)

(e) (f)

Figure 5.17: Results from CAD data in IGES files

Chapter 6

$\mathcal{C}^0$-PAVING OF MESHES BY QUADRILATERAL PATCHES

Abstract: We would like to decompose a closed orientable 2-manifold with arbitrary genus g into a set of four-sided tessellants. The input surface is a triangular surface mesh representing a free-form shape. The first process in the tessellation consists in finding a set of curves drawn on the surface mesh. The search for those curves is done in two steps: finding initial curves by means of algebraic or

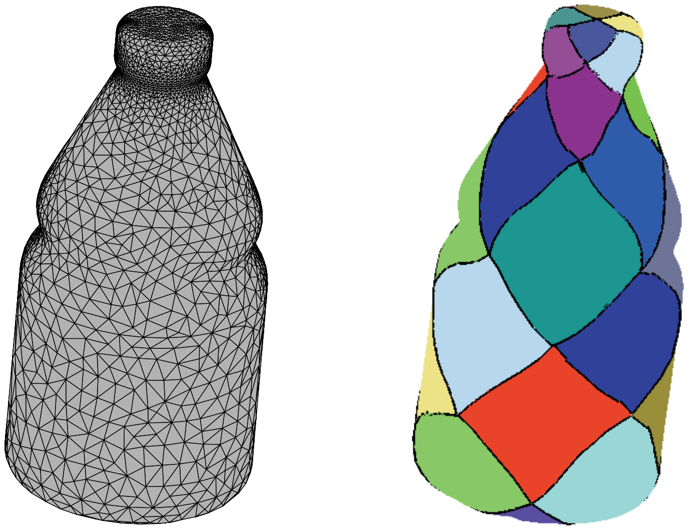

(a) (b)

Figure 6.1: Triangulation paving: (a) bottle in form of a triangular mesh (b)approximated surface in form of patches

geometric methods and improving them by using combinatorial optimization. Afterward, we perform the handle decomposition by slicing the surface along those curves. After flattening the surface on the plane by using the setting $(a_1 b_1 a_1^{-1} b_1^{-1}) \cdots (a_g b_g a_g^{-1} b_g^{-1})$, we use the resulting parametrization to split the surface into four-sided patches. We approximate those patches by B-spline surfaces in which we want to obtain global $\mathcal{C}^0$-joints. The proposed methods will be illustrated by several numerical examples.

6.1 Introduction

It often happens in practice that we have a numerical solver of integral equations which accepts mesh-free geometric data as input but the only geometry that we have at our disposal is a mesh. Treating every element of the mesh as a patch will surely explode the computational costs of the integral equation solver. Faced by such a conflicting situation, we have only one choice if we do not want to reject the input mesh: we try to generate a few patches from the mesh. That process amounts to approximating or interpolating the mesh by a set of practical surfaces such as B-spline patches (Fig. 6.1) and that is exactly the purpose of this chapter where we will treat only free-form surfaces. For algebraic surfaces such as spheres, planes, cylinders or combinations of them, we recommend [121, 120]. Apart from the fitting process, two main difficulties have to be considered. First, the partitioning of the large mesh into pieces which can be approximated by four-sided patches. Second, generation of parameter 2D meshes which allow us to perform the approximation by parametric surfaces. Before going any further, let us formulate our problem more specifically

6.2 Problem formulation

Let us suppose that we have a triangular mesh $\mathcal{M}$ which represents a surface embedded in the space. Additionally, we suppose that $\mathcal{M}$ is an orientable closed surface of arbitrary genus. We assume that the surface $\mathcal{M}$ is globally smooth in the sense that it can be approximated by a continuous surface which has no sharp edge anywhere. Those types of surfaces already cover a lot of interesting practical cases.

Our main purpose of this chapter is to present a methodology to approximate $\mathcal{M}$ into a set of surfaces $\{\mathcal{S}_r\}$. More precisely, our objective is twofold:

1. Decompose $\mathcal{M}$ into a set of four-sided submeshes $\{\mathcal{M}_r\}$.

2. Approximate every $\mathcal{M}_r$ by a B-spline patch S_r which is the image of a

parametric function

$$\mathbf{X}_r(u,v) = \sum_{i,j=0}^{n} \mathbf{d}_{ij}^r N_i^k(u) N_j^k(v).$$

Furthermore, we want the set of surfaces $\{S_r\}$ to be globally continuous. That is, for two neighboring submeshes $\mathcal{M}_r$ and $\mathcal{M}_\rho$, the surfaces $\mathcal{S}_r$ and $\mathcal{S}_\rho$ have $\mathcal{C}^0$-joint.

The realization of those goals will be done in a few steps. First, we have to find some curves such that if the surface $\mathcal{M}$ is split along those curves, we still have a single connected surface. The number of those curves depends on the genus of the surface $\mathcal{M}$. Second, we want to have a topological parametrization from a polygonal disk $P \subset \mathbf{R}^2$, which we still have to determine, to the mesh. After splitting the mesh into four-sided submeshes, we approximate the boundary of the submeshes $\mathcal{M}_r$ by B-spline curves $\mathcal{C}_q$. The last step consists in fitting the submeshes $\mathcal{M}_r$ by B-spline surfaces $\mathcal{S}_r$ where we interpolate the bounding B-spline curves $\mathcal{C}_q$. In the next sections, we would like to describe those steps one by one. For topological terms which are not clear in this chapter, refer to the appendices or [46, 82].

Remark 18 Since the above two objectives are already very difficult in their own, the problem about diffeomorphism is not analyzed in this chapter. Interested readers should consult works of Greiner [60] who has treated similar problems by developping a very long theory about removing wrinkles from a B-spline surface.

6.3 Handle decomposition

Consider a topological surface $\mathcal{M}$ which is orientable and of genus $g > 0$. We define a system of canonical curves to be a set of $2g$ closed curves $\mathcal{A}_1$, $\mathcal{B}_1$, $\cdots$, $\mathcal{A}_g$, $\mathcal{B}_g$ which fulfills the following criteria.

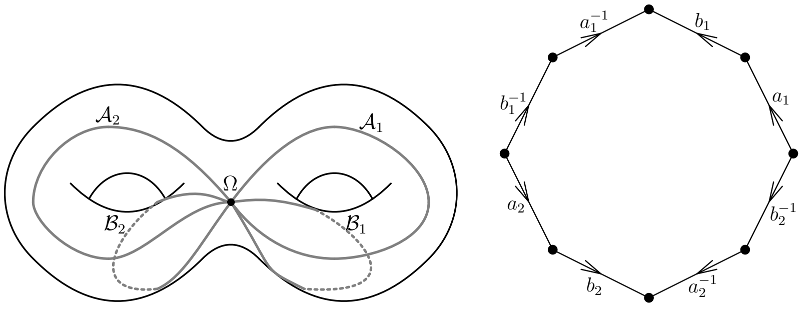
(C1) They reside on the surface $\mathcal{M}$: $\mathcal{A}_i \in \mathcal{M}$, $\mathcal{B}_i \in \mathcal{M}$ for all $i = 1, \cdots, g$.

(C2) They have one and only one intersection $\Omega \in \mathcal{M}$ known as *basepoint*.

(C3) If we cut the surface $\mathcal{M}$ along those curves then we still have a connected surface (Fig. 6.2) which can be flatten out to become a planar polygon [8, 36]

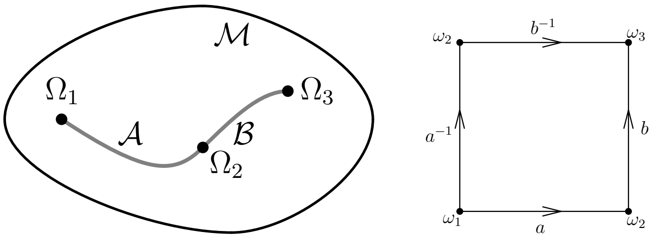
$$a_1 b_1 a_1^{-1} b_1^{-1} \cdots a_g b_g a_g^{-1} b_g^{-1}. \tag{6.1}$$

The planar edges a_i (resp. b_i) correspond to the canonical curves $\mathcal{A}_i$ (resp. $\mathcal{B}_i$). The exponents -1 in (6.1) specify that the corresponding planar edges are to be traversed in the opposite direction.



(a) (b)

Figure 6.2: (a) A surface with genus 2 and its four canonical curves $\mathcal{A}_1$, $\mathcal{B}_1$, $\mathcal{A}_2$, $\mathcal{B}_2$ (b)The flattened planar polygon $a_1b_1a_1^{-1}b_1^{-1}a_2b_2a_2^{-1}b_2^{-1}$



(a) (b)

Figure 6.3: Flattening: (a) Genus zero surface (b) Its flattened planar polygon $abb^{-1}a^{-1}$

The first step in our algorithm is to search for those canonical curves in which achieving criterion (C2) is the most difficult task. The process of cutting the surface $\mathcal{M}$ along those curves is better known as *handle decomposition.* After finding the canonical curves, we will describe an approach to define a parametrization from a planar polygon to the surface.

Remark 19 For surfaces of genus zero, the problem of splitting is much simpler because we need only to split the mesh along two abutting curves $\mathcal{A}$ and $\mathcal{B}$ as in Fig. 6.3(a). Those 3D-curves will be flattened out in the plane to obtain the boundary 2D-curves a, b, a^{-1}, b^{-1} as in Fig. 6.3(b).

6.4 Preliminary search for canonical curves

Since the following methods use standard techniques of finding canonical curves, we will only describe the approaches without long comments. Searching for the canonical curves is usually done in two stages. First, one finds some preliminary canonical curves $\overline{\mathcal{A}}_1$, $\overline{\mathcal{B}}_1$, $\cdots$, $\overline{\mathcal{A}}_g$, $\overline{\mathcal{B}}_g$ which fulfill the above criteria (C1), (C2), (C3) but which are undesirable in practice. The second stage consists in improving those preliminary curves by shifting them homotopically in order to find the final canonical curves $\mathcal{A}_1$, $\mathcal{B}_1$, $\cdots$, $\mathcal{A}_g$, $\mathcal{B}_g$. In this section, we would like to discuss how to achieve the first stage with two different methods. The first one is based on algebraic operations and the second one is a combinatorial approach.

6.4.1 Normalized canonical form

For a given mesh $\mathcal{M}$, let us denote by n_t, n_e, and n_v its number of triangles, edges and vertices respectively. We will denote the i-th triangle, the j-th edge and the k-th vertex of the mesh $\mathcal{M}$ by σ_i^2, σ_j^1, σ_k^0 respectively. We would like to describe briefly the way we compute the homology bases which represent some curves drawn on the surface (compare with Fig. 6.4(a)).

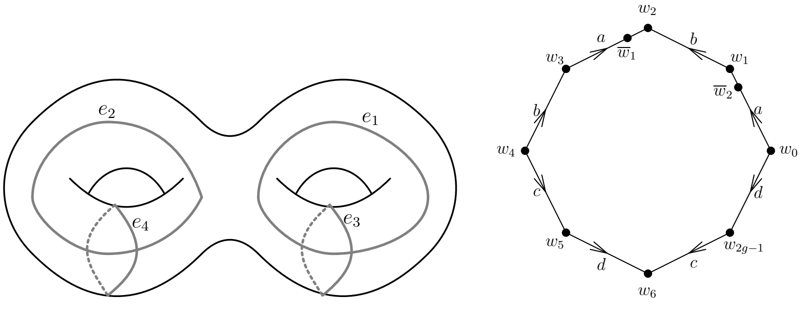
Consider the two incidence matrices [44] E_0 and E_1:

$$E_0 := \begin{bmatrix} [\sigma_1^1 : \sigma_1^0] & \cdots & [\sigma_{n_e}^1 : \sigma_1^0] \\ \cdots & \cdots & \cdots \\ [\sigma_1^1 : \sigma_{n_v}^0] & \cdots & [\sigma_{n_e}^1 : \sigma_{n_v}^0] \end{bmatrix} \quad E_1 := \begin{bmatrix} [\sigma_1^2 : \sigma_1^1] & \cdots & [\sigma_{n_f}^2 : \sigma_1^1] \\ \cdots & \cdots & \cdots \\ [\sigma_1^2 : \sigma_{n_e}^1] & \cdots & [\sigma_{n_f}^2 : \sigma_{n_e}^1] \end{bmatrix} \tag{6.2}$$

where $[\sigma_i^s : \sigma_j^{s-1}]$ denotes the incidence number of σ_i^s and σ_j^{s-1}.

Since those incidence numbers take integer values, we have to deal with integer computations. In particular, we may not perform any division operations. We have effectively the following relation pertaining to the operators E_0 and E_1:

$$\mathbf{Z}^{n_f} \xrightarrow{E_1} \mathbf{Z}^{n_e} \xrightarrow{E_0} \mathbf{Z}^{n_v}. \tag{6.3}$$



(a) (b)

Figure 6.4: (a)Homology bases e_1, e_2, e_3, e_4 (b)Coincidence of the images of different parameter nodes

The property of the boundary operator [115] implies in particular

$$E_0 \cdot E_1 = 0\,. \tag{6.4}$$

The set σ_i^k forms now a basis of the k-chains C_k. The reduction into normalized canonical form consists in searching for new bases of C_0, C_1, C_2 such that in those bases the above matrices take the form

$$\tilde{E}_0 = \begin{bmatrix} 0 & \vdots & \Lambda_0 \\ \cdots & . & \cdots \\ 0 & \vdots & 0 \end{bmatrix} \qquad \tilde{E}_1 = \begin{bmatrix} 0 & \vdots & \Lambda_1 \\ \cdots & . & \cdots \\ 0 & \vdots & 0 \end{bmatrix} \tag{6.5}$$

where Λ_0 and Λ_1 are square diagonal matrices. In other words, we are searching for three square matrices M_v, M_e, and M_t of size n_v, n_e, and n_f respectively such that

$$\tilde{E}_1 = M_e^{-1} E_1 M_t \qquad \tilde{E}_0 = M_v^{-1} E_0 M_e. \tag{6.6}$$

Therefore, the new bases of C_0, C_1 and C_2 can be expressed as linear combinations of the old bases $\{\sigma_i^k\}$ having coefficients from the columns of the matrices M_v, M_e, and M_t. Denote by γ_0 and γ_1 the dimensions of the square diagonal matrices Λ_0, Λ_1 and consider the new basis $\{e_1, \cdots, e_{n_e}\}$ of the 1-chains C_1. Because of the structure of $\tilde{E}_0$ and $\tilde{E}_1$ we have

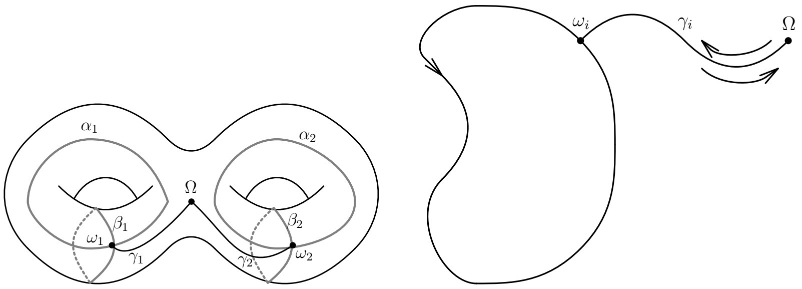
$$\operatorname{Ker} \tilde{E}_0 = \operatorname{span}\{e_1, \cdots, e_{n_e - \gamma_0}\}, \tag{6.7}$$

$$\operatorname{Im} \tilde{E}_1 = \operatorname{span}\{e_1, \cdots, e_{\gamma_1}\}. \tag{6.8}$$

The first homology group H_1 is spanned by the classes

$$[e_{\gamma_1+1}],\ [e_{\gamma_1+2}],\ \cdots,\ [e_{n_e-\gamma_0}] \tag{6.9}$$

in which $[f]$ is the class of a representant f.



(a) (b)

Figure 6.5: (a)Connect the homology bases with Ω by means of approach paths (b)Follow the approach path γ_i then the closed curve and finally traverse γ_i backward.

After that process, the classes from relation (6.9) provides us with a set of pairs of closed curves

$$\alpha_1, \beta_1, \cdots, \alpha_g, \beta_g \tag{6.10}$$

such that each pair (α_i, β_i) intersect only at a single point ω_i as illustrated in Fig. 6.5(a).

In order to have the preliminary canonical curves $\bar{\mathcal{A}}_1$, $\bar{\mathcal{B}}_1$,$\cdots$, $\bar{\mathcal{A}}_g$, $\bar{\mathcal{B}}_g$ from the curves α_1, β_1,$\cdots$, α_g, β_g, we use the following steps.

S1. Refine the triangular mesh $\mathcal{M}$ so that the edges of the curves (α_i, β_i) become edges of the mesh.

S2. Choose as basepoint any point Ω of the surface which does not reside on those curves.

S3. Use Dijkstra algorithm in order to find a path γ_i joining the basepoint Ω and the crossing point ω_i as illustrated in Fig. 6.5(a).

S4. At this point we have a set of loops $\mathcal{L}_i$. Each loop $\mathcal{L}_i$ is obtained by first traversing the approach path γ_i, then the closed curve $\mathcal{C}_i := \alpha_i$ or $\mathcal{C}_i := \beta_i$ and finally traversing γ_i backward as depicted in Fig. 6.5(b).

S5. Shift the loops $\mathcal{L}_i$ homotopically so that they traverse triangles internally as in Fig. 6.6(b). Use Reidemeister moves (see Appendix) so that the loops have only intersections at the basepoint Ω.

S6. The curves $\bar{\mathcal{A}}_i$, $\bar{\mathcal{B}}_i$ are then given by the loops $\mathcal{L}_k$.

6.4.2 Numerical realization

For the reduction of an integer matrix A into normalized canonical form as in relation (6.5), we distinguish three row operations:

(Op1) Swap the i-th row R_i and the j-th row R_j of A.

(Op2) Multiply the i-th row R_i by (-1).

(Op3) Replace the i-th row R_i by $R_i + qR_k$ where q is a given integer.

Note that all those three operations are invertible. Indeed, the first two are self-invertible while the inverse of the last one is $R_i - qR_k$.

Similar elementary operations can be done for the columns of A. The reduction into normalized form is processed in two steps. First, we reduce A into Smith normal form $\tilde{A} = GAF$ and then we perform some column swappings. That is, we search for $\tilde{A}$ such that

$$\tilde{A} = \begin{bmatrix} a_1 & & & & & \\ & \ddots & & & & \\ & & a_m & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}, \tag{6.11}$$

where $a_i \neq 0$ and a_i divides a_{i+1}. Let us show how to obtain (6.11) by using the above elementary operations. Suppose the matrix composed of the first c rows and the first c columns has already been reduced. The next steps are the following

1. Apply the above elementary operations so that $\alpha := A_{cc}$ divides the remaining entries in the c-th row and the c-th column. That is
$$\exists k_i, h_j \in \mathbf{Z} \quad \text{with} \quad A_{ic} = k_i\alpha \quad \text{and} \quad A_{cj} = h_j\alpha \quad \forall\, i, j > c\,. \tag{6.12}$$
2. Make all those entries zero by applying the third operation (Op3). That is, for all $i, j > c$, replace row R_i by $R_i - pR_c$ with $p = A_{ic}/\alpha$ and column C_j by $C_j - qC_c$ with $q = A_{cj}/\alpha$.

Since we are interested in the matrices of change of bases G^{-1} and F in (6.11), we should store the parameters for the elementary operations every time we apply them so that we do not need to explicitly compute the inverse of the integer matrices at the end of the process.

The difficulty of directly applying the above process to the incidence matrices E_0 and E_1 is that we have to do them simultaneously. Observe that we use the same matrix M_e of change of bases in equation (6.6) for both E_0 and E_1. For that end, we execute the following steps.

1. Reduce E_0 in normalized canonical form:
$$\overline{E}_0 = M_v^{-1} E_0 \overline{M}_e. \tag{6.13}$$
2. Apply the change of bases $\overline{M}_e$ to E_1:
$$\overline{E}_1 := \overline{M}_e^{-1} E_1. \tag{6.14}$$
3. Reduce $\overline{E}_1$ in normalized canonical form:
$$\tilde{E}_1 = \tilde{M}_e^{-1} \overline{E}_1 M_t = (\overline{M}_e \hat{M}_e)^{-1} E_1 M_t. \tag{6.15}$$
4. Define
$$\tilde{E}_0 := \overline{E}_0 \tilde{M}_e = M_v^{-1} E_0 (\overline{M}_e \tilde{M}_e). \tag{6.16}$$

By introducing $M_e := \overline{M}_e \tilde{M}_e$, we obtain the relation (6.6).

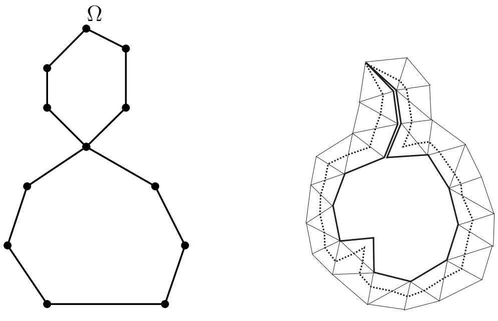
6.4.3 Bases of the fundamental group

Now, we would like to describe a second method of finding the temporary canonical curves $\bar{\mathcal{A}}_1$, $\bar{\mathcal{B}}_1$,$\cdots$,$\bar{\mathcal{A}}_g$, $\bar{\mathcal{B}}_g$ by using the fundamental group. In order to find the basis of the fundamental group $\pi_1(\mathcal{M})$ [45] of a two dimensional simplicial complex $\mathcal{M}$, we follow the next major steps.

- S1. Choose an arbitrary point Ω of $\mathcal{M}$ as a basepoint.
- S2. Find a list of loops (a sequence of edges starting and terminating at the basepoint Ω) $\mathcal{L}_e$, $e \in S$.
- S3. Shift the loops homotopically so that they traverse triangles as in Fig. 6.6(b). Use Reidemeister moves so that the loops have only intersections at the basepoint Ω.
- S4. The curves $\bar{\mathcal{A}}_i$, $\bar{\mathcal{B}}_i$ are then given by the loops $\mathcal{L}_e$.

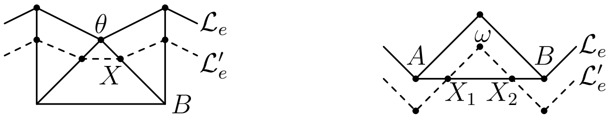
Let us describe the way of achieving step 2 by following the standard approach [45, 104, 114] requiring the use of a spanning tree. Let G be the edge-vertex graph which is generated from the mesh $\mathcal{M}$. We use the Breadth First Search (BFS) algorithm [2] to find a spanning tree T [73] of G which is rooted at the basepoint Ω. Now we would like to describe how to find the set of edges S of step 2. For that end, we will assemble recursively a subgraph Γ of G. As an initialization, we define S to be the empty set and Γ to be the spanning tree T. We perform the following updating process of Γ and S repeatedly.

Search for an edge $e = [v, w]$ from the set $G \setminus \Gamma$ such that there exists a triangle $[p, v, w]$ with $[p, v] \in \Gamma$ and $[p, w] \in \Gamma$. We distinguish now two cases depending



(a) (b)

Figure 6.6: (a)Impossible double nodes inside one individual loop (b)Initial loop $\mathcal{L}_e$ in bold line and homotopically shifted loop $\mathcal{L}'_e$ in dotted line.



(a) (b)

Figure 6.7: (a)First type of loop shifting (b)Second type of loop shifting

on the success of the search. If such an edge $[v, w]$ exists, we include it in the subgraph Γ:

$$\Gamma := \Gamma \cup [v, w]. \tag{6.17}$$

Otherwise, we pick any edge $[r, s]$ from $G \setminus \Gamma$ and then we include it in the list S. We terminate the above updating if the set of edges of $(\Gamma \cup S)$ is the same as that of G.

Now that we find a list of edges S, we want to show how to generate a loop $\mathcal{L}_e$ for every $e = [t, u] \in S$. Find a path p_t (resp. p_u) which belongs to the spanning tree T, which starts at t (resp. u) and which terminates at the basepoint Ω. We define the loop $\mathcal{L}_e$ corresponding to the edge e as

$$\mathcal{L}_e := p_t e p_u^{-1} \tag{6.18}$$

which is p_t followed by e and then by the inverse p_u. The loops $\mathcal{L}_e$ are also commonly termed '*generators*' of the fundamental group $\pi_1(\mathcal{M})$.

6.4.4 Homotopic transformation

In this section, we would like to show how to shift curves on a surface so that they do not have self-intersection and that they intersect only at the basepoint. The realization of the last steps of the algorithms in sections 6.4.1 and 6.4.3 will be found here. Before describing homotopic transformations, we would like to

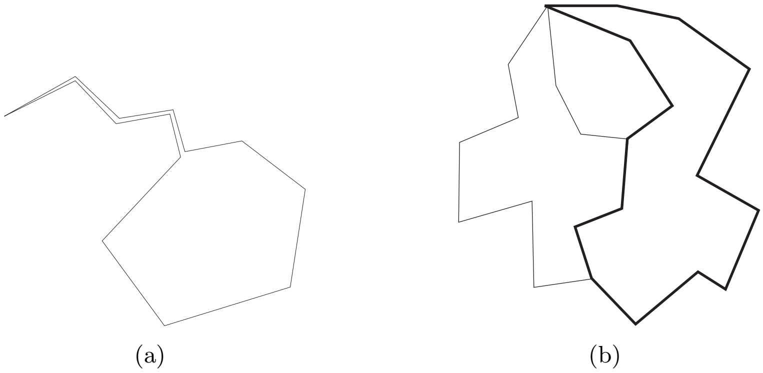


Figure 6.8: (a)A loop with double edges (b)Two loops with common edges.

mention a few features of the loops $\mathcal{L}_e$ which result from those steps. There are mainly two common problems occurring to those loops. The first one is a local one which happens for an individual loop without considering the other loops. The second problem is a global one in which two different loops $\mathcal{L}_e$ and $\mathcal{L}_f$ could interfere. They have intersections away from the basepoint. More accurately, there could exist

- Double edges: different edges which have the same metric information as in Fig. 6.8(a) or
- Double nodes which are not at the basepoint (Fig. 6.8(b)).

Now we would like to describe a method of homotopically repairing the loops $\mathcal{L}_e$ so that those problems are eliminated. For a given loop $\mathcal{L}_e$, our current objective is to generate another loop $\mathcal{L}'_e$ which starts and terminates at the basepoint Ω and which traverses the triangles on the left side of the original loop $\mathcal{L}_e$ as in Fig. 6.6(b). Since the surface $\mathcal{M}$ is orientable, it is not difficult to determine whether an emanating edge from the loop $\mathcal{L}_e$ is on its left or on its right side by simple strategy (i.e. without any metric information). That can be done by orienting all three local vertices of a triangle in anti-clockwise direction. Now let us denote by $\mathcal{R}_\theta$ the list of all edges $[\theta, B]$ which emanate from a node $\theta \neq \Omega$ of the loop $\mathcal{L}_e$ and which go to the left hand side such that $B \notin \mathcal{L}_e$.

In order to find $\mathcal{L}'_e$, we need to generate new nodes which we categorize in two types. A new node of first type X resides on an edge $e = [\theta, B]$ of $\mathcal{R}_\theta$ and its distance from the node θ is controlled by a prescribed positive parameter μ as $X = \mu B + (1 - \mu)\theta$. As an illustration, we find in Fig. 6.7(a) a dotted curve $\mathcal{L}'_e$ obtained by shifting a solid curve $\mathcal{L}_e$ using nodes of first type.

New nodes of second type are generated when two consecutive edges of a triangle τ of $\mathcal{M}$ belong to the loop $\mathcal{L}_e$. In such a case, a node ω inside the triangle τ is generated as in Fig. 6.7(b). Additionally, two nodes X_1, X_2 are generated on

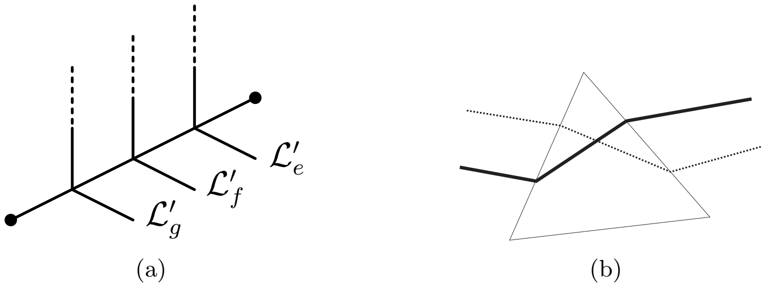


Figure 6.9: (a)Uniformly splitting an edge traversed by loops (b)Loop intersection inside a triangle.

the edge whose endpoints belong to the loop $\mathcal{L}_e$. That means, we have created three nodes of second type X_1, X_2, ω. The generation of the loop $\mathcal{L}'_e$ consists in tracing a curve which goes through those newly generated nodes (of first and second types) as shown in Fig. 6.6(b).

We note the obvious fact that this process generates a loop $\mathcal{L}'_e$ which is homotopically equivalent to the original loop $\mathcal{L}_e$. We would like to mention that this process solves the first problem of coinciding edges. That is due to the fact that the loop $\mathcal{L}_e$ can never be like the one depicted in Fig. 6.6(a). Note that the above homotopic shifts do not solve the global problem of interference between two different loops $\mathcal{L}'_e$ and $\mathcal{L}'_f$.

Let us now propose a remedy for that global problem. The first stage of the remedy consists in uniformization in which we search for the list of edges which are traversed by loops. We shift afterwards the nodes on the edges in such a way that they are equispaced as in Fig. 6.9(a). After this stage, no two different loops $\mathcal{L}'_e$ and $\mathcal{L}'_f$ have vertices of the same coordinates (other than the basepoint). But there could still exist the problem of intersection of two different loops which could only occur inside triangles as in Fig. 6.9(b). In order to eliminate such intersections, we apply a series of Reidemeister moves [79, 1, 31].

6.5 Improvements of the canonical curves

After a direct application of the formerly described algorithms, the resulting canonical curves could be too long or unnecessarily complicated. That could lead to undesirable splitting of the topological surface $\mathcal{M}$ as in Fig. 6.10(b). In order that we can apply the subsequent algorithm, we need to shorten the lengths of the canonical curves. In this section, we would like to describe an algorithm to improve the preliminary canonical curves $\overline{\mathcal{A}}_1$,$\overline{\mathcal{B}}_1$, $\cdots$,$\overline{\mathcal{A}}_g$, $\overline{\mathcal{B}}_g$ in order to obtain the final canonical curves $\mathcal{A}_1$, $\mathcal{B}_1$, $\cdots$, $\mathcal{A}_g$, $\mathcal{B}_g$. Before describing the improvement algorithm, let us introduce the notion of weighted split graph which can be obtained from the edge-vertex graph and the current canonical curves.

6.5.1 Weighted split graph

Suppose we have a few curves $\mathcal{S}_1$, $\mathcal{S}_2$, $\cdots$, $\mathcal{S}_{2g}$ on the surface mesh $\mathcal{M}$. That is, every curve $\mathcal{S}_i$ is composed of a sequence of consecutive edges of the mesh $\mathcal{M}$. For every index i from $\{1, \cdots, 2g\}$, let us describe [23] how a weighted graph H_i is defined by specifying its nodes, edges and weights. Consider the dual graph H of the edge-vertex graph G. That is to say, we insert an edge between the centers of gravity of two neighboring triangles of $\mathcal{M}$. At this first stage, the surface $\mathcal{M}$ is split into different faces F_k which are delineated by the edges of H. If we further split the mesh $\mathcal{M}$ by the edges of $\mathcal{S}_k$ $(k \neq i)$, we obtain a splitting of the topological surface $\mathcal{M}$ into several connected components C_1, C_2, $\cdots$, C_N. In Fig. 6.10(a) , we see an illustration of the graph H_i where thin segments represent the edge-vertex graph G while the dual graph H is depicted with dotted lines and the canonical curves are shown in bold faces. The highlighted region is an example of a connected component. For every connected components C_r, we generate a graph-node N_r of the split graph H_i.

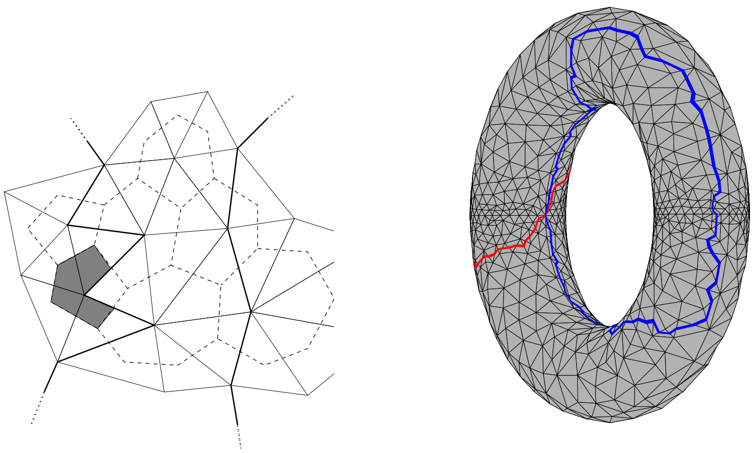
With the nodes of H_i in place, let us now define its edges. Consider two graph-nodes N_p and N_q whose corresponding connected components are C_p and C_q respectively. Consider the parent faces F_p and F_q of the connected components C_p and C_q. Let us denote by $\mathcal{T}_p$ (resp. $\mathcal{T}_q$) the set of triangles of F_p (resp. F_q) which are in the connected component C_p (resp. C_q). Between the two graph-nodes N_p and N_q of the graph H_i, we introduce a graph-edge e_{pq} of H_i if the sets $\mathcal{T}_p$ and $\mathcal{T}_q$ share a common triangle t_{pq}.

Now let us define the weight w_{pq} that is assigned to the graph-edge e_{pq} of H_i. Two nodes s_p and s_q of the triangle t_{pq} must be the centers of the parent faces F_p and F_q respectively. The weight w_{pq} assigned to the edges e_{pq} will then be the Euclidean distance between s_p and s_q.

6.5.2 Improvement algorithm

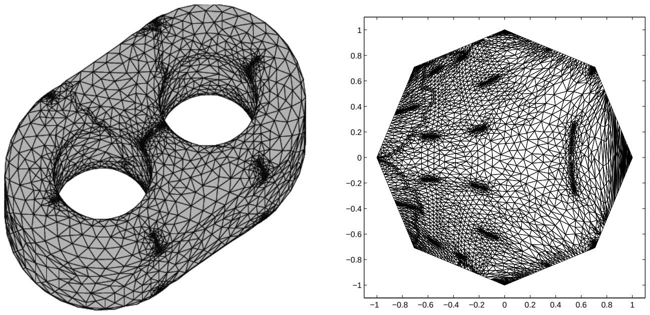
The improvement of the canonical curve is processed recursively as follow. Suppose the current canonical curves are $\mathcal{S}_1$, $\mathcal{S}_2$, $\cdots$, $\mathcal{S}_{2g}$ with g being the genus of the surface. For a given index $i = 1, \cdots, 2g$, we would now like to describe hot to find the improved canonical curve $\mathcal{S}'_i$. By using the weighted graph H_i, we can search for the shortest path from the starting and the terminating connected components of S_i with the help of the Dijkstra algorithm. The new path $\mathcal{S}'_i$ can then be deduced from the thus defined path.

For the canonical curves, the above improvement can be applied to the curves $\mathcal{S}_1 := \bar{\mathcal{A}}_1$, $\mathcal{S}_2 := \bar{\mathcal{B}}_1$, $\cdots$, $\mathcal{S}_{2g} := \bar{\mathcal{B}}_g$. The improved canonical curves are therefore $\mathcal{A}_1 := \mathcal{S}'_1$, $\mathcal{B}_1 := \mathcal{S}'_2$, $\cdots$, $\mathcal{B}_g := \mathcal{S}'_{2g}$.



(a) (b)

Figure 6.10: (a) Dual graph split by the canonical curves (b) Undesirable canonical curves.



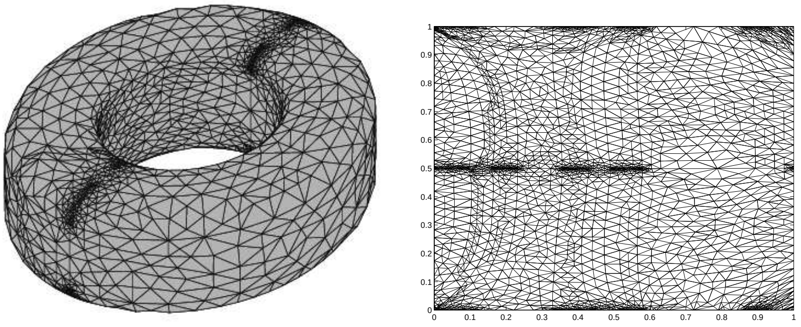
(a) (b)

Figure 6.11: (a)Surface with genus 2 (b)parametrization on $aba^{-1}b^{-1}cdc^{-1}d^{-1}$

6.6 Paving into quadrilateral patches

Suppose that we have an orientable surface $\mathcal{M}$ without boundary and a set of canonical curves. After slicing the surface $\mathcal{M}$ along the curves, we would like to have a single parametrization of the whole surface $\mathcal{M}$ from a planar parameter domain $\mathbf{P}$ as illustrated in Fig. 6.11 and Fig. 6.12.

For surfaces of higher genera $g \geq 1$, we proceed similarly but we need several parametrizations: the one for the whole surfaces, one in the neighborhood of each canonical curve.



(a) (b)

Figure 6.12: (a)Surface with genus 1 (b)parametrization on $aba^{-1}b^{-1}$

6.6.1 Mesh parametrization

The following parametrization technique can be applied to any surface mesh $\mathcal{R} \subset \mathbf{R}^3$ having a boundary consisting of m sides. We want to determine a function $\mathbf{q}$ from $\mathcal{R}$ to a planar convex polygon P having m sides. The mapping $\mathbf{q}$ is completely defined if we can define a 2D mesh μ on P such that the i-th node of μ maps to the i-th node $\mathbf{x}_i$ of $\mathcal{R}$ (and conversely). The first step in the determination of the mesh parametrization is to specify the image $\mathbf{q}(\mathbf{x}_i)$ of boundary vertices $\mathbf{x}_i \in \partial\mathcal{R}$. That can be done by chord length [39] approaches. Suppose now that $\mathbf{q}(\mathbf{x}_i) \in \partial P$ is known for every boundary node $\mathbf{x}_i \in \partial\mathcal{R}$. Let us denote by V_I and by V_B the set of indices of internal and boundary nodes of the mesh $\mathcal{R}$ respectively. For every vertex $\mathbf{x}_i$, let us denote by

$$\mathcal{N}_i := \{j \,:\, [\mathbf{x}_i, \mathbf{x}_j] \text{ is an edge of } \mathcal{R}\}. \tag{6.19}$$

For every point $\mathbf{x}_i$ with $i \in V_I$, its image $\mathbf{q}(\mathbf{x}_i)$ will be a convex combination of the images of its neighbors:

$$\mathbf{q}(\mathbf{x}_i) = \sum_{j \in \mathcal{N}_i} \lambda_{ij} \mathbf{q}(\mathbf{x}_j). \tag{6.20}$$

The coefficients λ_{ij}, known as weights, are some positive values which can be determined from the initial mesh $\mathcal{R}$ and which satisfy:

$$\sum_{j \in \mathcal{N}_i} \lambda_{ij} = 1 \qquad \forall\, i \in V_I. \tag{6.21}$$

Below we are describing how to find their values. From relation (6.20), we obtain

$$\mathbf{q}(\mathbf{x}_i) = \sum_{j \in \mathcal{N}_i \cap V_I} \lambda_{ij} \mathbf{q}(\mathbf{x}_j) + \sum_{j \in \mathcal{N}_i \cap V_B} \lambda_{ij} \mathbf{q}(\mathbf{x}_j) \tag{6.22}$$

or equivalently

$$\mathbf{q}(\mathbf{x}_i) - \sum_{j\in\mathcal{N}_i\cap V_I} \lambda_{ij}\mathbf{q}(\mathbf{x}_j) = \sum_{j\in\mathcal{N}_i\cap V_B} \lambda_{ij}\mathbf{q}(\mathbf{x}_j). \tag{6.23}$$

Since we know the values of $\mathbf{q}(\mathbf{x}_j)$ for all $j \in V_B$, the right hand side of the above equation is completely known. This leads to a linear system of order $\mathrm{card}(V_I)$

$$A\mathbf{X} = \mathbf{F} \tag{6.24}$$

where the unknown $\mathbf{X}$ consists of $\mathbf{q}(\mathbf{x}_j)$ with $j \in V_I$. By solving this linear system the values of the internal nodes $\mathbf{q}(\mathbf{x}_j)$ are completely known.

Now we would like to show how to choose the values of the weights λ_{ij}. In [42], there is a comprehensive study about various methods of specifying them. We show only one method based on convex combination.

Let us consider a vertex $\mathbf{x}_i \in \mathcal{R}$. For any $j \notin \mathcal{N}_i$, we set $\lambda_{ij} := 0$. The determination of λ_{ij} for every $j \in \mathcal{N}_i$ is done in two steps. The first step consists in flattening out the submesh $\mathcal{R}_i$ consisting of the triangles which are incident upon $\mathbf{x}_i$. That is, the nodes of $\mathcal{R}_i$ are composed of $\mathbf{x}_i$ and $\mathbf{x}_j$ with $j \in \mathcal{N}_i$. Flattening $\mathcal{R}_i$ consists in finding a planar mesh $\mathcal{P}_i$ having nodes $\mathbf{u}_r$ which map to $\mathbf{x}_r$ of $\mathcal{R}_i$. We do that flattening process in the following way. First, $\mathbf{u}_i$ is chosen to be any point in the plane. Then, we want to keep the distances unchanged, i.e.

$$\|\overrightarrow{\mathbf{u}_i\mathbf{u}_j}\|_{\mathbf{R}^2} = \|\overrightarrow{\mathbf{x}_i\mathbf{x}_j}\|_{\mathbf{R}^3} \qquad \forall\, j \in \mathcal{N}_i. \tag{6.25}$$

For every triangle $\tau_j = [\mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_i]$ incident upon $\mathbf{x}_i$, denote by α_j the angle made by $\overrightarrow{\mathbf{x}_i\mathbf{x}_j}$ and $\overrightarrow{\mathbf{x}_i\mathbf{x}_k}$. Similarly, we denote by β_j the angle made by $\overrightarrow{\mathbf{u}_i\mathbf{u}_j}$ and $\overrightarrow{\mathbf{u}_i\mathbf{u}_k}$. We want that

$$\alpha_i = \rho \cdot \beta_i,$$

where the coefficient ρ is chosen in such as way that the sum

$$\sum_{j\in\mathcal{N}_i} \beta_i = 2\pi.$$

Now that we have a planar mesh $\mathcal{P}_i$ having $\mathbf{u}_r$ as vertices, we want to proceed to the second step of determining λ_{ij}. For any $j \in \mathcal{N}_i$, find $\mathbf{u}_k$ and $\mathbf{u}_l$ such that the triangle $[\mathbf{u}_j, \mathbf{u}_k, \mathbf{u}_l]$ contains the point $\mathbf{u}_i$. Let $\tau_j^{i,j}$, $\tau_k^{i,k}$, $\tau_l^{i,l}$ be its barycentric coordinates:

$$\mathbf{u}_i = \tau_j^{i,j}\mathbf{u}_j + \tau_k^{i,j}\mathbf{u}_k + \tau_l^{i,j}\mathbf{u}_l.$$

By defining $\tau_r^{i,j} = 0$ for every $r \in \mathcal{N}_i$ such that $r \notin \{j, k, l\}$, we have

$$\mathbf{u}_i = \sum_{r\in\mathcal{N}_i} \tau_r^{i,j}\mathbf{u}_r.$$

The final coefficient is defined as

$$\lambda_{ij} := \sum_{s\in\mathcal{N}_i} \tau_s^{i,j}.$$

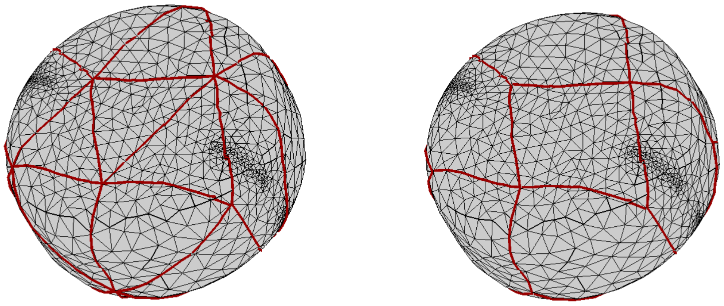

(a) (b)

Figure 6.13: (a) Triangular decomposition (b) Quadrilateral decomposition

6.6.2 Splitting into four-sided submeshes

In this section, we will describe a way to decompose a surface $\mathcal{M}$ into pieces of four-sided domains. In order to facilitate the presentation, we suppose that $\mathcal{M}$ is of genus zero and that we have a parametrization $\mathcal{P}$ from a rectangular domain. Our way of segmenting $\mathcal{M}$ into several pieces is done into two steps as explained in Fig. 6.13. The first one consists in splitting $\mathcal{M}$ into large curved triangles. Afterwards, we convert the resulting triangular decomposition $\mathcal{T}$ into four-sided decomposition $\mathcal{Q}$. There are two reasons for using a large triangular decomposition. First, the Delaunay approaches are better understood in the context of triangulations than quadrangulations and it is very simple to split a triangle without introducing any hanging node and without increasing the number of new elements significantly. Second, we have already a parametrization $\mathcal{P}$ in disposition and we can thus apply the idea of generalized Delaunay triangulation that we saw in chapter 5.

In order to convert a triangulation $\mathcal{T}$ to a quadrangulation $\mathcal{Q}$, we follow the algorithm in section 3.10 of chapter 3 where we try to merge two neighboring triangular patches to form a four-sided patch.

Now we would like to describe our decomposition algorithm to obtain the large triangulation $\mathcal{T}$. Since the method of generalized Delaunay triangulation was already presented in full detail in chapter 5, we need only to specify the criteria for splitting an edge (Fig. 6.14(a)) and for inserting a node inside a triangle (Fig. 6.14(b)). On that account, let us introduce the following quality assessment.

Definition 18 For a curve $\mathcal{C}$ defined on an interval $[a, b]$, we define its linear curve distortion $\epsilon(\mathcal{C})$ as

$$\epsilon(\mathcal{C}) := \frac{1}{\|AB\|} \sum_{i \in \mathcal{I}} \|\mathcal{C}(t_i) - \mathcal{L}(t_i)\|, \text{where} \quad A := \mathcal{C}(a), \quad B := \mathcal{C}(b) \quad \text{and} \tag{6.26}$$

$$\mathcal{L}(t_i) := \lambda_i A + (1 - \lambda_i) B \quad \text{with} \quad \lambda_i = (t_i - a)/(b - a) \quad \text{and} \tag{6.27}$$

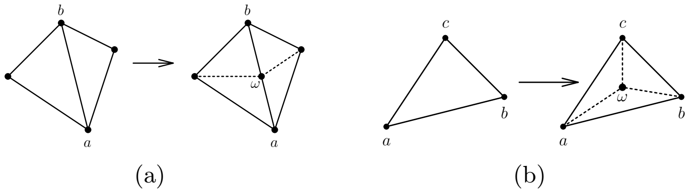


Figure 6.14: Insertion of a new node: (a) edge cutting (b) triangle subdivision

t_i are some values taken within $[a, b]$.

This definition can be extended to the case of linear surface distortion $\mu(S)$ of a surface S defined on a planar triangle $[a, b, c]$ where we need to replace $\mathcal{L}(t_i)$ by

$$\mathcal{L}(u_i, v_i) := \lambda_i^a A + \lambda_i^b B + \lambda_i^c C \quad \text{with} \tag{6.28}$$

λ_i^a, λ_i^b, λ_i^c being the barycentric coordinates if (u_i, v_i) within $[a, b, c]$.

The determination of the large triangulation $\mathcal{T}$ is done recursively by starting from an initial coarse triangulation $\mathcal{T}^{(0)}$. The large triangulation $\mathcal{T}^{(k+1)}$ is obtained by $\mathcal{T}^{(k)}$ by generalized Delaunay point insertion: we split an edge of $\mathcal{T}^{(k)}$ if its linear curve distortion exceeds some prescribed accuracy ϵ_0. We decompose a large triangle into three subtriangles if its linear surface distortion is larger than some prescribed μ_0.

The initial coarse large triangulation $\mathcal{T}^{(0)}$ can be done manually by picking a few vertices on the mesh $\mathcal{M}$. We recall that the quadrangulation conversion is only feasible if the number of triangles of $\mathcal{T}$ is even. If we start from a coarse triangulation $\mathcal{T}^{(0)}$ with an even number of triangles, then applying the mesh operations in Fig. 6.14 will keep the parity of the number of triangles even. In order to have the parametric representations, we use the method in section 6.6. For genus zero surfaces, we need two parametrizations $\mathcal{P}_1$ and $\mathcal{P}_2$. The first parametrization $\mathcal{P}_1$ consists of a split surface and parametrization $\mathcal{P}_2$ is a representation of the surface in the vicinity of the split curve $(\mathcal{A}, \mathcal{B})$ as in Fig. 6.3(a). For surfaces of higher genera $g \geq 1$, we proceed similarly but the parametrizations are now neighbors of the canonical curves and the basepoint.

6.7 Surface fitting with $\mathcal{C}^0$-joint

Let us suppose that we have K submeshes $\mathcal{M}_r$ which we would like to approximate by B-spline surfaces S_r having the following representation:

$$\mathbf{X}_r(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{n} \mathbf{d}_{ij}^r N_i^k(u) N_j^k(v) \qquad \forall\, r = 1, \cdots, K \ ; \tag{6.29}$$

where $N_i^k(\cdot)$ is the usual [39] B-spline basis function with respect to some knot sequence $\theta_0 = \cdots = \theta_{k-1} \leq \theta_k \leq \cdots \leq \theta_{n+1} = \cdots = \theta_{n+k}$:

$$N_i^k(t) = \frac{t-\theta_i}{\theta_{i+k-1}-\theta_i} N_i^{k-1}(t) + \frac{\theta_{i+k}-t}{\theta_{i+k}-\theta_{i+1}} N_{i+1}^{k-1}(t). \tag{6.30}$$

In the formula (6.29), we suppose that the parameters along the $u-$direction and $v-$direction are the same (Otherwise, one can use parameter elevation techniques [39]). The unknowns are the de-Boor points $\mathbf{d}_{ij}^r$ for all indices r. Let us denote by $\mathcal{D}_q$ $q = 1, \cdots, M$ the piecewise linear curves which bound the submeshes $\mathcal{M}_r$. That is, every submesh $\mathcal{M}_r$ is bounded exactly by four piecewise linear curves. Since we would like to have $\mathcal{C}^0$-joints, our first step is to approximate those curves $\mathcal{D}_q$ by B-spline curves $\mathcal{C}_q$

$$\mathcal{C}_q(t) = \sum_{i=0}^{n} \delta_i^q N_i^k(t). \tag{6.31}$$

Let us denote by $\{\mathbf{y}_s : s = 0, \cdots, m\}$ the vertices of the piecewise linear curve $\mathcal{D}_q$. We want that the endpoints of $\mathcal{D}_q$ are interpolated by the de-boor points of $\mathcal{C}_q$, i.e. $\delta_0^q := \mathbf{y}_0$ and $\delta_n^q := \mathbf{y}_m$. After using a chord length parametrization [39] which associates $\mathbf{y}_s$ to some t_s, we may approximate the curve $\mathcal{D}_q$ in a least square sense:

$$\sum_{s=0}^{m} \| \sum_{i=0}^{n} \delta_i^q N_i^k(t_s) - \mathbf{y}_s \|^2 \longrightarrow \min_{\delta_i^q}. \tag{6.32}$$

The sequence t_s belongs to some closed interval $[a, b]$. Without loss of generality, we may suppose $[a, b] = [\theta_0, \theta_{n+k}]$, $t_0 = \theta_0$ and $t_m = \theta_{n+k}$ (use shifting and scaling operations). Before describing resampling method, let us recall the following result [53, 110].

Theorem 14 (Schoenberg-Whitney) Suppose we have a sequence $(t_i)_i \subset \mathbf{R}$ which corresponds to the points $(\mathbf{y}_i)_i \subset \mathbf{R}^3$. There exists exactly one B-spline of order k with knot θ_j which solves the least-square approximation in (6.32) if $(t_i)_i$ has a subsequence $(t_{r(i)})_i$ which satisfies the Schoenberg-Whitney conditions of order k:

$$\theta_i \leq t_{r(i)} \leq \theta_{i+k} \qquad i = 0, \cdots, n. \tag{6.33}$$

Remark 20 (Resampling) It is possible that the formerly mentioned chord length parametrization $(t_s, \mathbf{y}_s)$ does not satisfy the Schoenberg-Whitney conditions. In that case, we have to perform a resampling process in the following way. We denote by ρ the piecewise linear function such that $\rho(t_s) = \mathbf{y}_s$ and we choose some appropriate number $p \geq 2$. For every $s = 0, \cdots, \bar{n} := (n - k + 2)$ and $\sigma = 0, \cdots, p - 1$, introduce:

$$\left\{ \begin{array}{lcl} \tilde{t}_{\sigma+s(p-1)} & := & \lambda_\sigma \theta_{(s+k-1)} + (1-\lambda_\sigma)\theta_{s+k} \\ \tilde{\mathbf{y}}_{\sigma+s(p-1)} & := & \rho(\tilde{t}_{\sigma+s(p-1)}). \end{array} \right. \tag{6.34}$$

where $\lambda_\sigma := \sigma/p \in [0,1[$ and

$$\left\{ \begin{array}{lll} \tilde{t}_{p(\bar{n}+1)} & := & \theta_{n+1} \\ \tilde{\mathbf{y}}_{p(\bar{n}+1)} & := & \rho(\tilde{t}_{p(\bar{n}+1)}). \end{array} \right. \tag{6.35}$$

We replace therefore the sample $(t_s, \mathbf{y}_s)$ by $(\tilde{t}_s, \tilde{\mathbf{y}}_s)$ which satisfy the Schoenberg-Whitney condition.

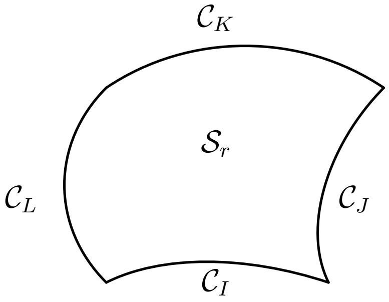


Figure 6.15: The B-spline curves $\mathcal{C}_I$, $\mathcal{C}_J$ $\mathcal{C}_K$, $\mathcal{C}_L$ are interpolated by $\mathcal{S}_r$

Now, we would like to fit a B-spline $\mathcal{S}_r$ to the submesh $\mathcal{M}_r$ which is bounded by four piecewise linear curves $\mathcal{D}_I$, $\mathcal{D}_J$, $\mathcal{D}_K$, $\mathcal{D}_L$. If we want the B-spline surface S_r to interpolate the B-spline curves $\mathcal{C}_I$, $\mathcal{C}_J$, $\mathcal{C}_K$, $\mathcal{C}_L$ which approximate those piecewise linear curves (Fig. 6.15), we specify the boundary de-Boor points by

$$\left\{ \begin{array}{lll} \mathbf{d}^r_{i0} := \delta^I_i & \mathbf{d}^r_{in} := \delta^K_i & \forall\, i = 0, \cdots, n \\ \mathbf{d}^r_{0j} := \delta^L_j & \mathbf{d}^r_{nj} := \delta^J_j & \forall\, j = 0, \cdots, n. \end{array} \right. \tag{6.36}$$

In the following discussion, we will skip the superscript and subscript r indicating the patch index when no confusion is possible. From the vertices of the submeshes $\mathcal{M}_r$, we can have some parameterized samples $(u_k, v_k) \in \mathbf{R}^2$, $\mathbf{P}_k \in \mathbf{R}^3$ from which we would like to determine the control points $\mathbf{d}_{ij}$.

By excluding the boundary de-Boor points $\mathbf{d}_{ij}$ with the help of (6.36), we have the remaining unknowns whose set of indices is given by

$$E := \{(i,j) : (i,j) \neq (i,0) \quad (i,j) \neq (i,n) \quad (i,j) \neq (0,j) \quad (i,j) \neq (n,j)\} \tag{6.37}$$

and whose number is $R := (N+1)^2 - 4N$.

The surface fitting is then reduced to a least-square approximation:

$$\sum_{l=0}^{R} \left\| \sum_{(i,j)\in E} \mathbf{d}_{ij} N^k_i(u_l) N^k_j(v_l) - \mathbf{P}_l \right\|^2 \longrightarrow \min_{\mathbf{d}_{ij}}. \tag{6.38}$$

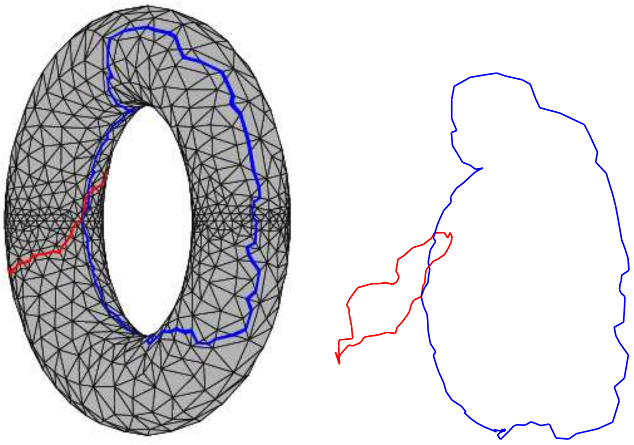

(a) (b)

Figure 6.16: Before Reidemeister moves

Let us note that the samples for the surface fitting might also violate the (tensor-product) Schoenberg-Whitney condition. In that case, we have to generalize the resampling method of remark 20.

Like in every least-square approximation, the problems in (6.32) and (6.38) can be transformed into linear normal equations

$$\mathbf{M}^T \mathbf{M} X = \mathbf{M}^T b. \tag{6.39}$$

6.8 Numerical results

In this section, we would like to describe some results of the former methods when applied to some meshes. First, we would like to show the developments of the canonical curves in three stages:

- Before applying the Reidemeister moves,
- After application of the Reidemeister moves which give us the preliminary canonical curves,
- The ultimate canonical curves after optimization.

We will describe our results with the help of two surfaces which have respectively genus 1 and 2 (torus and pretzel). In the next figures, we display on the left hand sides the canonical curves which are traced on the underlying surface meshes. We display on the right hand sides the canonical curves without surfaces in order to facilitate the visualization of the curve properties. In Fig. 6.16, we can see

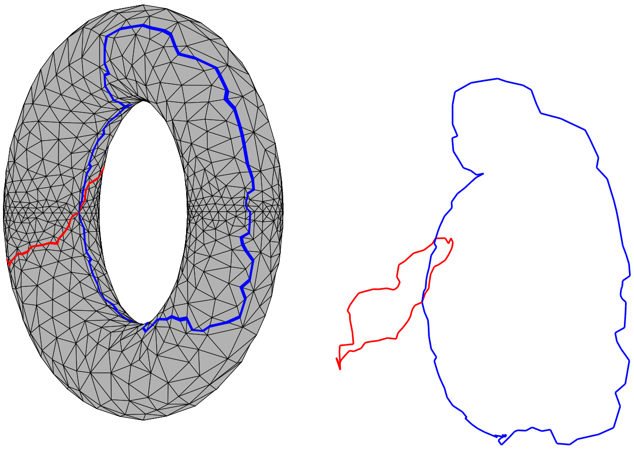
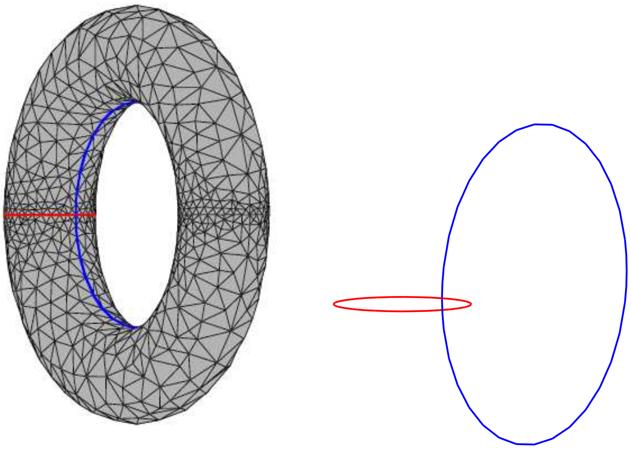

(a) (b)

Figure 6.17: After Reidemeister moves

that the canonical curves still have an intersection which is not at the basepoint. After application of the Reidemeister moves, there remains only one intersection at the basepoint but the curves are still very undesirable as seen in Fig. 6.17. By applying the optimization algorithm that we described in section 6.5, we obtain the ultimate canonical curves (Fig. 6.18) in which we have one single intersection point and the quality of the curves has been optimized homotopically.

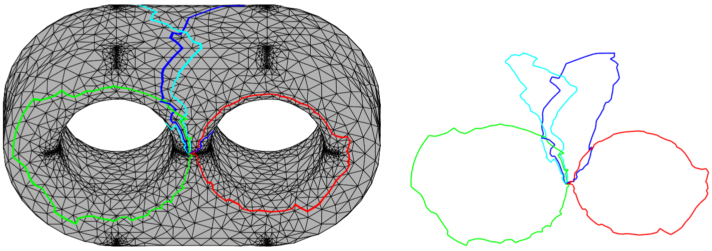
The same process has been successfully applied to a surface of genus two (see Fig. 6.19 through Fig. 6.22). We can observe in Fig. 6.21 a magnification of the results after Reidemeister moves in the neighborhood of the basepoint. As Fig. 6.21(a) clearly shows, there are still several curve crossings which need to be eliminated with the help of Reidemeister moves. In Fig. 6.21(b) we see only one position where we have curve crossings which are exactly at the basepoint.

We would like now to present the second result that consists in reconstructing surfaces from three meshes having respectively 8288, 11656 and 21872 triangles. We see the surface meshes together with the splittings in Figs 6.23(a), 6.24(a), 6.25(a), while the corresponding surface patches are located on the right figures.



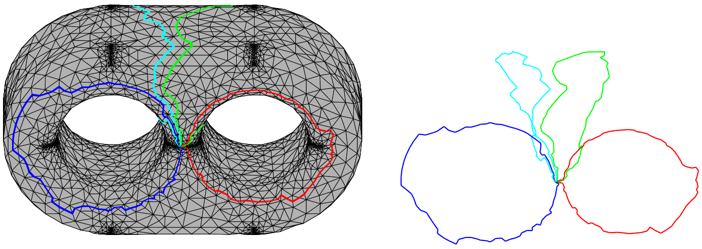
(a) (b)

Figure 6.18: After combinatorial optimization



(a) (b)

Figure 6.19: Before Reidemeister moves



(a) (b)

Figure 6.20: After Reidemeister moves

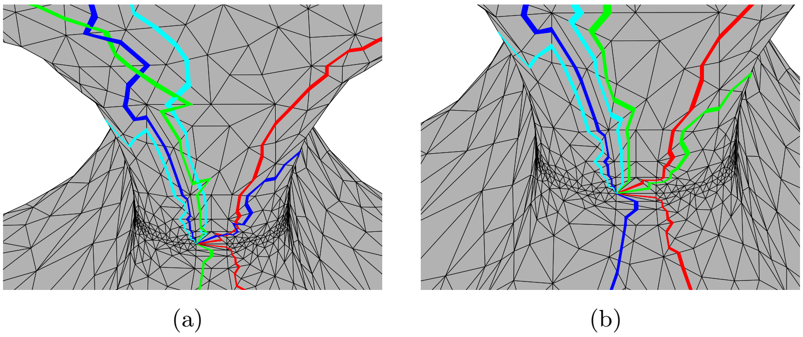

(a) (b)

Figure 6.21: Removing curve-crossings with the help of Reidemeister moves

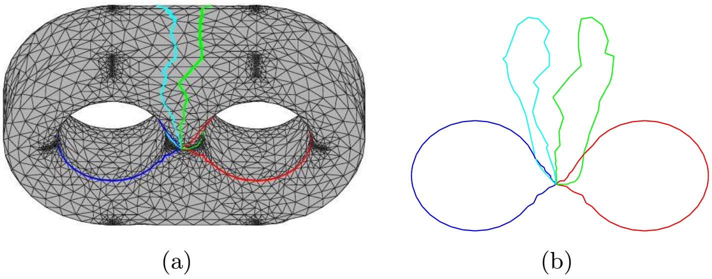

(a) (b)

Figure 6.22: After combinatorial optimization

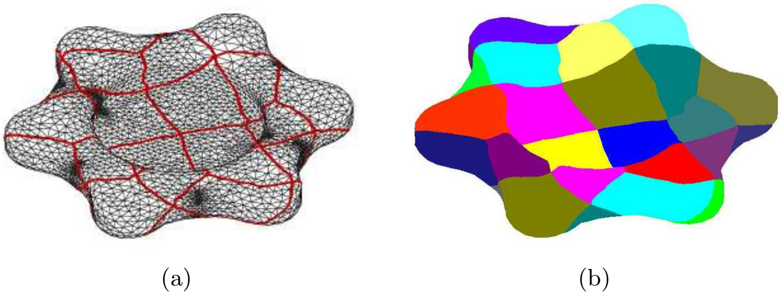

(a) (b)

Figure 6.23: (a)Mesh with 8288 triangles (b)Approximation with 40 patches

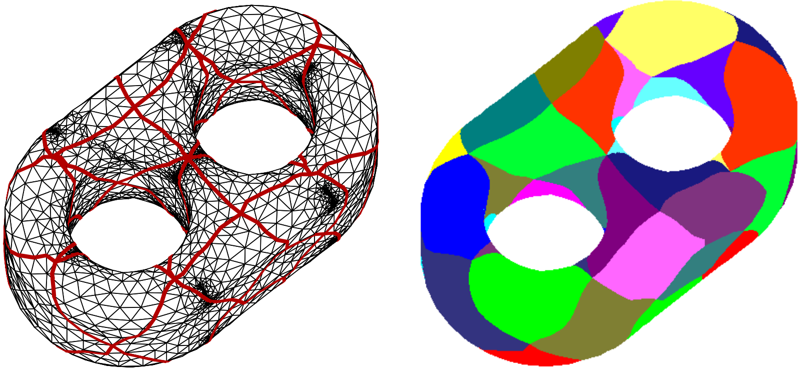

(a) (b)

Figure 6.24: (a)Mesh with 11656 triangles (b)Approximation with 32 patches

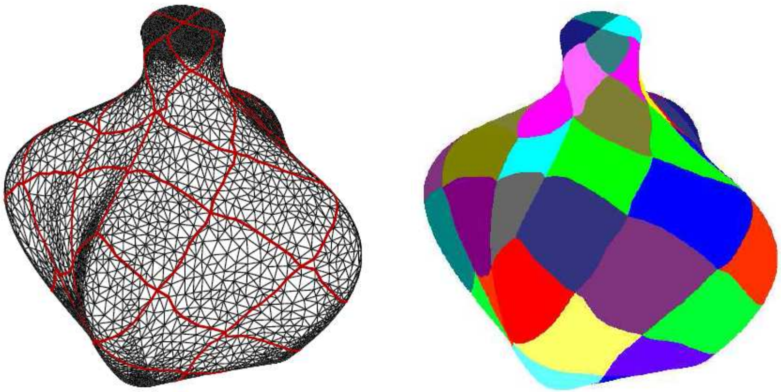

(a) (b)

Figure 6.25: (a)Mesh with 21872 triangles (b)Approximation with 52 patches

Appendix A

NUMERICAL TOPOLOGY

In this appendix, we would like to describe some notions that are usually met in computational topology. We mention only some topics which are closely related to our surface approximation approaches. For readers having no knowledge about numerical topology, the introductions of the notions here can be found in [44, 45] which contain materials which lead directly to applications. Interested readers who need an in-depth insight are urged to read [82, 115].

A.1 Simplicial complexes

The majority of the surfaces that are met in numerical topology are represented as simplicial complexes which we want to briefly introduce now. Let us consider some points P_0,...,P_k in $\mathbf{R}^{k+1}$ for which the family $\{\overrightarrow{0P_i}\}$ is supposed to be linearly independent. A k-dimensional simplex σ^k with vertices $\{P_i\}$ is defined to be the convex hull of $\{P_i\}$. For any point P of σ^k there must exist m_i, termed barycentric coordinates, whose sum is unity such that

$$\overrightarrow{0P} = \sum_{i=0}^{k} m_i \overrightarrow{0P_i}\,. \tag{A.1}$$

The i-th face σ_i^{k-1} of σ^k is the set of all points in σ^k for which the i-th barycentric coordinate m_i is zero. A simplex σ^k is said to be oriented if an ordering of its vertices P_i is prescribed. Two orderings of the simplex differing by an *even* permutation [44] are assumed to determine the same orientation. If the orders of vertices differ by an *odd* permutation, they determine opposite orientations. An oriented simplex is denoted by $+\sigma^k$ and a simplex with opposite orientation by $-\sigma^k$. Throughout this document, a simplex is always supposed to be oriented. We will drop the superscript k if no confusion is possible. A simplicial complex $\mathcal{M}$ is a set of simplices $\{\sigma_i\}_{i\in I}$ which fulfills the following conditions:

1. $\mathcal{M} \subset \cup_{i\in I}\sigma_i$

2. For every two different indices i and j such that $\sigma_i \cap \sigma_j$ is empty, either one of them is the face of the other or they have a common face which is the intersection of them.

The dimension of the simplicial complex $\mathcal{M}$ is the highest possible dimension of its elements σ_i.

A.2 Boundary operator and incidence matrix

For practical purpose, a triangular mesh $\mathcal{M}$ representing a surface embedded in the space can be viewed as a simplicial complex. We have to assign some orientations to the edges and the triangles of $\mathcal{M}$. The triangles (resp. edges, resp. vertices) of $\mathcal{M}$ can be considered as oriented 2-simplices (resp. 1-simplices, resp. 0-simplices). In this section, we are introducing the boundary operator ∂_k together with its matrix representation. We define the incidence number $[\sigma_j^{k+1} : \sigma_i^k]$ of the j-th $(k+1)$-dimensional simplex σ_j^{k+1} and the i-th k-dimensional simplex σ_i^k of $\mathcal{M}$ to be

- $[\sigma_j^{k+1} : \sigma_i^k] := 0$ if σ_j^{k+1} and σ_i^k are not incident,
- $[\sigma_j^{k+1} : \sigma_i^k] := +1$ if σ_j^{k+1} and σ_i^k are incident and their orientations agree,
- $[\sigma_j^{k+1} : \sigma_i^k] := -1$ if σ_j^{k+1} and σ_i^k are incident and their orientations are opposite.

The k-th incidence matrix of a mesh $\mathcal{M}$ is the matrix E_k whose entries are

$$E_k(i,j) := [\sigma_j^{k+1} : \sigma_i^k]\,. \tag{A.2}$$

The boundary of an oriented simplex σ^k can be expressed in terms of its faces σ_i^{k-1} by

$$\partial_k \sigma^k := \sum_{i=0}^{k} (-1)^i \sigma_i^{k-1}. \tag{A.3}$$

If we denote by α_k the number of k-dimensional simplices in the mesh $\mathcal{M}$, then relation (A.3) allows us to formulate the expression of the boundary operator ∂_k in terms of the incidence coefficients:

$$\partial_k \sigma_j^{k+1} = \sum_{i=1}^{\alpha_k} [\sigma_j^{k+1} : \sigma_i^k]\, \sigma_i^k. \tag{A.4}$$

We introduce also the notion of k-chains which are linear combinations of k-simplices:

$$c = \sum_i a_i \sigma_i^k. \tag{A.5}$$

Roughly speaking, a 1-chain is a list of edges and a 2-chain is a set of triangles from the mesh $\mathcal{M}$. Further the definition of a boundary operator ∂_k can be easily extended to k-chains by linearity:

$$\partial_k c = \sum_i a_i \partial_k \sigma_i^k . \tag{A.6}$$

A.3 Homology group

By denoting the set of all k-chains of a given simplicial complex $\mathcal{M}$ by $\mathcal{C}_k$, the boundary operator ∂_k defines a linear mapping from $\mathcal{C}_k$ to $\mathcal{C}_{k-1}$. We have therefore the following diagram

$$\mathcal{C}_{k+1} \xrightarrow{\partial_{k+1}} \mathcal{C}_k \xrightarrow{\partial_k} \mathcal{C}_{k-1} \, . \tag{A.7}$$

The k-chains whose image by ∂_k are zero are called *cycles*. They represent the k-chains which do not have any boundary. The k-chains which are images of ∂_{k+1} are called *boundaries*. In other words, those two sets represent the kernel Z_k of ∂_k and the image B_k of ∂_{k+1}

A very important property [82] of the boundary operator from algebraic topology is that

$$\partial_k \circ \partial_{k+1} = 0 \, . \tag{A.8}$$

In terms of the incidence matrices, the above relation is equivalent to

$$E_k \cdot E_{k+1} = 0 \, . \tag{A.9}$$

Because of property (A.8), we have $B_k \subset Z_k$ and therefore we can define the k-th homology group to be the quotient group

$$H_k(\mathcal{M}) := Z_k / B_k \, . \tag{A.10}$$

The k-th Betti number β_k of the simplicial complex $\mathcal{M}$ is the dimension of the k-th homology group $H_k(\mathcal{M})$. Although this definition is valid for any k, we will be primarily interested in the first homology group $H_1(\mathcal{M})$. If we consider a surface $\mathcal{M}$ which is orientable, then the (first) Betti number is given by

$$\beta_1 = 2g + h - 1, \tag{A.11}$$

where g is the genus of the surface $\mathcal{M}$ and h is the number of boundary curves.

Appendix B

CURVES ON SURFACES

In this appendix, we would like to point out a brief excursus on the topological modeling of curves which are drawn on surfaces. That will prove particularly useful during the course of the determination of the canonical curves in chapter 6. For that end, let us first review the topological representations and classifications of surfaces.

B.1 Topological surfaces

Contrarily to surfaces that are usually met in differential geometry, local smoothness is not required in topological modeling. That fact makes the usefullness of topological surfaces very interesting in practice because it allows us to model corners, sharp edges while being able to keep track of the surface as a whole. In other words, it is not always necessary to model the surface as assembly of pieces of surfaces.

More precisely, a topological surface $\mathcal{M}$ is defined to be a set of points in $\mathbf{R}^3$ for which every point $\mathbf{x}$ has an open neighborhood $\mathbf{U}$ which is homeomorphic to the open unit disc $\mathbf{D} \subset \mathbf{R}^2$.

A very well-known and useful notion that is utilized both in theoretical derivations and in implementation is the notion of topological disks, which have also several other terminologies. In order for us to be able to introduce its definition, let us consider a finite discrete set $\mathcal{A}$ which we will call alphabet whose elements will be called letters. A polygonal disk $\mathbf{P}$ is a polygon with oriented sides labeled with letters from the alphabet $\mathcal{A}$. Each letter on the polygonal disk $\mathbf{P}$ appears at most twice. For each polygonal disk $\mathbf{P}$ we can assign a code ω in the following way. The code ω is initialized to be empty. We start from any vertex of the polygon $\mathbf{P}$ and follow the sides of the polygon in a counterclockwise orientation. If the direction of an edge labeled with a letter e has a clockwise orientation, then we append e in the code otherwise we append e^{-1}. An example of a code

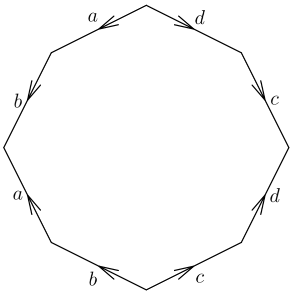
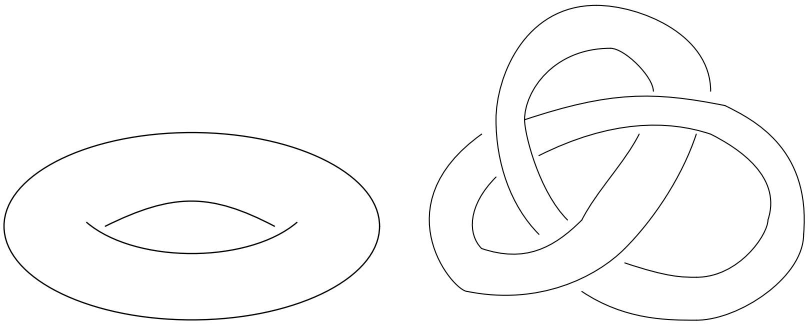


Figure B.1: A polygonal disk with code $aba^{-1}b^{-1}cdc^{-1}d^{-1}$

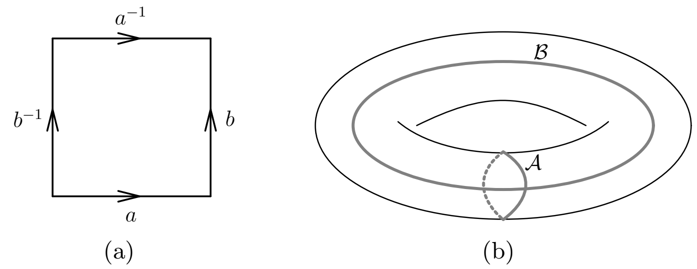


(a) (b)

Figure B.2: Two surfaces of genus unity

$\omega = aba^{-1}b^{-1}cdc^{-1}d^{-1}$ is illustrated in Fig. B.1 where we have started from its topmost vertex. A more rigorous definition involving abstract group theory could for example be found in [114].

Now we would like to describe how one can obtain a topological surface $\mathcal{M}$ from a polygonal disk **P**. Given a polygonal disk **P**, we can define a topological surface by gluing the edges having the same letter in **P** in which the edge orientations have to be respected. As an illustration, if we glue the appropriate sides in the topological disk with code $\omega = aba^{-1}b^{-1}$ of Fig. B.3(a), then we obtain the torus in Fig. B.3(b). The curves $\mathcal{A}$ and $\mathcal{B}$ of the torus correspond to the sides a and b from the code ω. A very well known property from geometric topology [8] states that for any given compact connected surface $\mathcal{M}$, there is a polygonal disk **P** such that $\mathcal{M}$ is obtained from **P** by gluing operations. We refer the reader to [8] for a more abstract definition of a gluing technique which involves quotient maps and identification spaces.



(a) (b)

Figure B.3: A topological disk and a corresponding topological surface

B.2 Classification of surfaces

Before introducing the sense of commutators, let us recall the notion of genus which is a very simple but useful notion that is used in many algorithms. The genus $g(\mathcal{M})$ of a surface $\mathcal{M}$ is the maximum number of pairwise disjoint simple closed curves along which we can cut the surface $\mathcal{M}$ while keeping it in a single piece. As an illustration, the genera of the surfaces in Fig. B.2 are both unity. Since we are interested in orientable triangular surface meshes, we want to point out that in that case, the genus is

$$g(\mathcal{M}) = \frac{1}{2}[1 - \chi(\mathcal{M})], \tag{B.1}$$

in which $\chi(\mathcal{M})$ is the Euler characteristic of the surface $\mathcal{M}$:

$$\chi(\mathcal{M}) = v - e + t, \tag{B.2}$$

where v, e, t are respectively the number of vertices, edges and triangles of the surface mesh $\mathcal{M}$.

In our application of geometric topology to surface splitting from chapter 6, we need a way to parameterize the whole surface from a single polygonal disk $\mathbf{P}$. On that account, we want a method to determine the polygonal disk $\mathbf{P}$ of a given surface $\mathcal{M}$ in function of its genus g. Before introducing that characterization, note that commutator $[a, b]$ is a code of the form

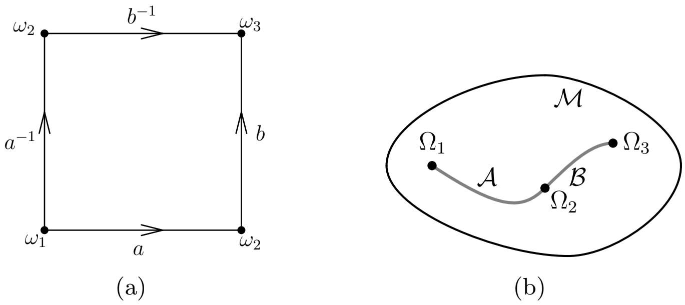
$$[a, b] := aba^{-1}b^{-1}. \tag{B.3}$$

Every compact orientable surface $\mathcal{M}$ without boundary with genus $g > 0$ can be obtained [45, 114] from the following code which is the multiplications of g commutators

$$\omega = [a_1, b_1][a_2, b_2]...[a_g, b_g]\,. \tag{B.4}$$

Furthermore, if the genus of the surface $\mathcal{M}$ is zero then the appropriate code is

$$\omega = abb^{-1}a^{-1}. \tag{B.5}$$



(a) (b)

Figure B.4: (a)Topological disk $\omega = abb^{-1}a^{-1}$ (b)Surface of genus zero

B.3 Fundamental group

Let us consider a topological surface $\mathcal{M}$ and a point $\Omega \in \mathcal{M}$ which is called basepoint. A loop is a curve lying on $\mathcal{M}$ starting from Ω and ending at Ω. More precisely, a loop is the image of a continuous map

$$\mathcal{L} : [0,1] \to \mathcal{M} \qquad \mathcal{L}(0) = \mathcal{L}(1) = \Omega. \tag{B.6}$$

We define the product of two loops $\mathcal{L}_0$ and $\mathcal{L}_1$ to be the loop

$$(\mathcal{L}_0 \cdot \mathcal{L}_1)(t) := \begin{cases} \mathcal{L}_0(2t) & \text{if} \quad t \in [0, 0.5] \\ \mathcal{L}_1(2t-1) & \text{if} \quad t \in [0.5, 1]\,. \end{cases} \tag{B.7}$$

Two loops $\mathcal{L}_0$ and $\mathcal{L}_1$ are equivalent if $\mathcal{L}_0$ can be continuously deformed to $\mathcal{L}_1$. In other words, they are equivalent if there is a continuous function

$$\phi : [0,1] \times [0,1] \to \mathcal{M} \tag{B.8}$$

such that

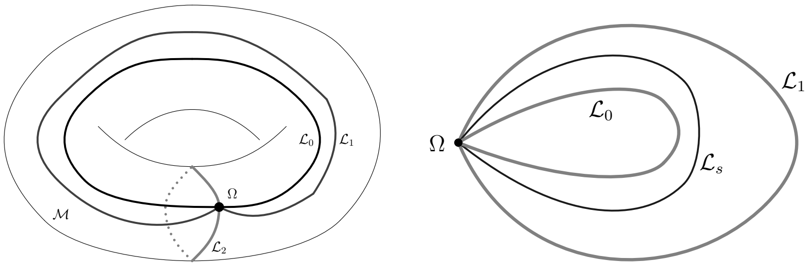
$$\phi(t,0) = \mathcal{L}_0(t) \qquad \phi(t,1) = \mathcal{L}_1(t) \qquad \mathcal{L}_r(t) := \phi(t,r). \tag{B.9}$$

As an illustration, in the torus of Fig. B.5(a) $\mathcal{L}_0$ and $\mathcal{L}_1$ are equivalent but $\mathcal{L}_0$ and $\mathcal{L}_2$ are not. It is to be noted that the loop equivalence form an equivalent relation in the set of loops of $\mathcal{M}$.

With this equivalence relation in place, we can define classes of equivalent loops whose set is denoted by $\pi_1(\mathcal{M}, \Omega)$. With the product operation that we defined in (B.7), the set $\pi_1(\mathcal{M}, \Omega)$ defines a group which is usually called (first) fundamental group.

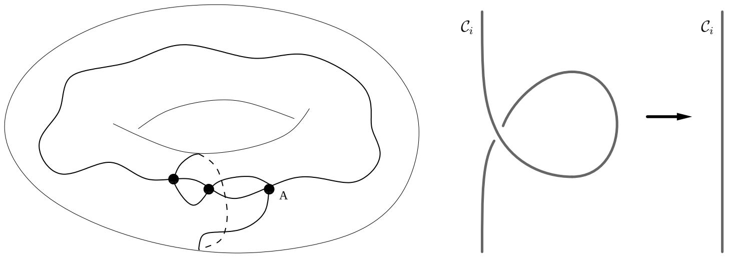
B.4 Reidemeister moves

For the reduction of the number of crossings of curves on a surface, we need to apply a series of Reidemeister moves [45] which we want to introduce now. They



(a) (b)

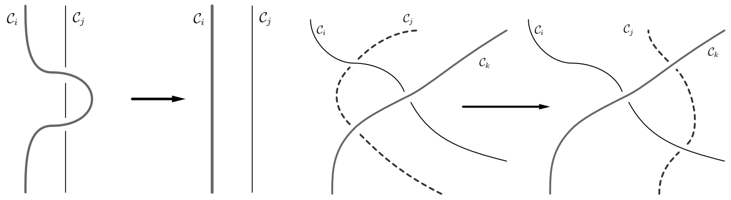
Figure B.5: (a)Equivalent and non-equivalent loops, (b) Loop $\mathcal{L}_0$ moves continuously into loop $\mathcal{L}_1$



(a) (b)

Figure B.6: (a) Unnecessary crossings (b) Twist: first Reidemeister move

are the principal moves that are often met in knot theory [1]. In our search for canonical curves from chapter 6, we are interested in curves which are drawn on a given surface $\mathcal{M}$. In order to be able to parameterize a surface $\mathcal{M}$ of genus g to a planar polygon efficiently as we do in chapter 6, we need that the curves $\mathcal{C}_1$,...,$\mathcal{C}_{2g}$ cross at one single point of the surface, namely at the basepoint. Unfortunately, during the course of searching a sequence of the curves $\mathcal{C}_i$, unnecessary crossings are often possible. That happens for example in Fig. B.6(a) where we have a surface $\mathcal{M}$ with genus unity, on which two curves have two crossings other than the basepoint. We need therefore to shift the curves $\mathcal{C}_i$ while keeping them on the surface. One distinguishes three types of Reidemeister moves whose planar projected diagrams can be observed in Figs. B.6(b), B.7(a), B.7(b). The first Reidemeister move or '*twist*' removes a crossing by a twisting operation. The second one or '*poke*' removes two crossings in which one curve overcrosses another as in Fig. B.7(a). The third move or '*slide*' is applied when we have a curve $\mathcal{C}_i$ on the left (resp. right) of a crossing. The third one consists therefore in shifting $\mathcal{C}_i$ so that it becomes on the right (resp. left of the crossing).



(a) (b)

Figure B.7: (a)Poke: second Reidemeister move (b)Slide: third Reidemeister move.

Bibliography

[1] C. Adams, The knot book, Freeman and Co., New York, 1994.

[2] S. Althoen, R. Bumcrot, Introduction to discrete mathematics, PWS-KENT publishing company, Boston, 1988.

[3] K. Atkinson, The numerical solution of integral equations of the second kind, Cambridge university press, Cambridge, 1997.

[4] DXF Reference guide, `www.autodesk.com/techpubs/autocad/dxf`, 2002.

[5] D. Barnette, A. Edelson, All orientable 2-manifolds have finitely many minimal triangulations, Israel J. Math. **62**, No. 1 (1988) 90–98.

[6] R. Barnhill, Computer aided surface representation and design, in: Proc. surfaces in CAGD, North-Holland, Amsterdam, 1986, pp. 1–24.

[7] H. Bast, K. Mehlhorn, G. Schäfer, H. Tamaki, A heuristic for Dijkstra's algorithm with many targets and its use in weighted matching algorithms, Algorithmica **36**, No. 1 (2003) 75–88.

[8] E. Bloch, A first course in geometric topology and differential geometry, Birkhäuser, Boston, 1997.

[9] A. Bobenko, B. Springborn, A discrete Laplace-Beltrami operator for simplicial surfaces, Preprint math. DG/0503219, 2005.

[10] H. Borouchaki, P. George, Aspects of 2-D Delaunay mesh generation, Int. J. Numer. Methods Eng. **40**, No. 11 (1997) 1957–1975.

[11] H. Borouchaki, P. Laug, P. George, Parametric surface meshing using a combined advancing-front generalized Delaunay approach, Int. J. Numer. Methods Eng. **49**, No. 1-2 (2000) 233–259.

[12] P. Bose, G. Toussaint, Characterizing and efficiently computing quadrangulations of planar point sets, Comput. Aided Geom. Des. **14**, No. 8 (1997) 763–785.

[13] F. Bossen, P. Heckbert, A pliant method for anisotropic mesh generation, in: Fifth international meshing roundtable, Saundia National Laboratories, 1996, pp. 63–76.

[14] M. Bousquet, J. Hester, 3D-Konstruktion und Präsentation mit AutoCAD, IWT-Verl., München, 1993.

[15] D. Bräss, Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie, Springer, Berlin, 1992.

[16] D. Bremner, F. Hurtado, S. Ramaswami, V. Sacristan, Small convex quadrangulations of point sets, in: Proc. 12th international symposium, ISAAC 2001, Christchurch, New Zealand, 2001, pp. 623–635.

[17] S. Brunnermeier, S. Martin, Interoperability cost analysis of the U.S. automotive supply chain, Research Triangle Institute Project Number 7007-03, March, 1999.

[18] M. Chaudhry, An extended Schwarz-Christoffel transformation for numerical mapping of polygons with curved segments, COMPEL **11**, No. 2 (1992) 277–293.

[19] M. Chaudhry, Numerical computation of the Schwarz-Christoffel transformation parameters for conformal mapping of arbitrarily shaped polygons with finite vertices, COMPEL **11**, No. 2 (1992) 263–275.

[20] H. Chen, H. Pottmann, Approximation by ruled surfaces, J. Comput. Appl. Math. **102** (1999) 143–153.

[21] L. P. Chew, Constrained Delaunay triangulations, Algorithmica **4**, No. 1 (1989) 97–108.

[22] P. Ciarlet, The finite element method for elliptic problems, North-Holland, Amsterdam, 1978.

[23] E. Colin de Verdière, Raccourcissement de courbes et décomposition de surfaces, Ph.D. thesis, Université Paris 7, 2003.

[24] J. Corney, 3D modeling with ACIS kernel and toolkit, John Wiley & Sons, Chichester, 1997.

[25] M. Costabel, J. Saranen, The spline collocation method for parabolic boundary integral equations on smooth curves, Numer. Math. **93**, No. 3 (2003) 549–562.

[26] W. Dahmen, Wavelet and multiscale methods for operator equations, Acta Numerica **6** (1997) 55–228.

[27] W. Dahmen, R. Schneider, Wavelets on manifolds I: Construction and domain decomposition, SIAM J. Math. Anal. **31**, No. 1 (1999) 184–230.

[28] W. Dahmen, A. Kunoth, K. Urban, Biorthogonal spline wavelets on the interval: stability and moment conditions, Appl. Comput. Harmon. Anal. **6**, No. 2 (1999) 132–196.

[29] M. De Berg, M. Van Kreveld, M. Overmars, O. Schwarzkopf, Computational geometry: algorithms and applications, Springer, Berlin, 2000.

[30] C. de Boor, A practical guide to splines, Springer, New York, 1978.

[31] M. De Graaf, A. Schrijver, Making curves minimally crossing by Reidemeister moves, J. Comb. Theory, Ser. B **70**, No. 1 (1997) 134–156.

[32] DIN 44302, Datenübertragung, Datenübermittlung, Feb 1987.

[33] DIN 66301, Rechnergestütztes Konstruiren, Jul 1986.

[34] V. Dolejsí, Anisotropic mesh adaptation technique for viscous flow simulation, East-West J. Numer. Math. **9**, No. 1 (2001) 1–24.

[35] H. Edelsbrunner, T. Tan, An upper bound for conforming Delaunay triangulations, Discrete Comput. Geom. **10** (1993) 197–213.

[36] R. Engelking, K. Sieklucki, Topology: a geometric approach **4**, Heldermann Verlag, Berlin, 1992.

[37] A. Evans, D. Miller, NASA IGES and NASA-IGES-NURBS-Only Standard, in: Handbook of grid generation, CRC Press, 1999, Chapter 31.

[38] H. Everett, W. Lenhart, M. Overmars, T. Shermer, J. Urrutia, Strictly convex quadrilateralizations of polygons, in: Proc. fourth Canadian conference on computational geometry, St. Johns, Newfoundland, 1992, pp. 77–82.

[39] G. Farin, Curves and surfaces for computer aided geometric design: a practical guide, Academic Press, Boston, 1997.

[40] G. Farin, Discrete Coons patches, Comput. Aided Geom. Des. **16**, No. 7 (1999) 691–700.

[41] M. Floater, E. Quak, Piecewise linear prewavelets on arbitrary triangulations, Numer. Math. **82**, No. 2 (1999) 221–252.

[42] M. Floater, K. Hormann, Parameterization of triangulations and unorganized points, in: European summer school lecture notes, Tutorials on multiresolution in geometric modelling, Munich Univ. of Technology, Germany, 2001, pp. 127–154.

[43] M. Floater, K. hormann, Surface parameterization: a tutorial and survey, in: Advances in multiresolution for geometric modelling, edts. N. Dodgson, M. Floater, M. Sabin, Springer, Berlin, 2005, pp. 157–186.

[44] A. Fomenko, Visual geometry and topology, Springer, Berlin, 1993.

[45] A. Fomenko, S. Matveev, Algorithmic and computer methods for three-manifolds, Kluwer Academic Publishers, Dordrecht, 1997.

[46] A. Fomenko, T. Kunii, Topological modeling for visualization, Springer, Tokyo, 1997.

[47] A. Forrest, On Coons and other methods for the representation of curved surfaces, Comput. Graph. Img. Process. **1** (1972) 341–359.

[48] P. Frey, H. Borouchaki, Surface mesh quality evaluation, Int. J. Numer. Methods Eng. **45**, No. 1 (1999) 101–118 .

[49] M. Garey, D. Johnson, F. Preparata, R. Tarjan, Triangulating a simple polygon, Inf. Process. Lett. **7** (1978) 175–179.

[50] M. Garland, P. Heckbert, Surface simplification using quadric error metrics, in: Proc. SIGGRAPH, 1997.

[51] P. L. George, H. Borouchaki, Delaunay triangulation and meshing. Application to finite elements, Hermes edition, Paris, 1998.

[52] P. L. George, Automatic mesh generation. Application to finite element methods, Chichester: John Wiley & Sons Ltd., Paris: Masson, 1991.

[53] R. Goldenthal, M. Bercovier, Spline curve approximation and design by optional control over the knots, Computing **72**, No. 1-2 (2004) 53–64.

[54] W. Gordon, C. Hall, Construction of curvilinear co-ordinate systems and applications to mesh generation, Int. J. Numer. Methods Eng. **7** (1973) 461–477.

[55] W. Gordon, C. Hall, Transfinite element methods: blending-function interpolation over arbitrary curved element domains, Numer. Math. **21** (1973) 109–129.

[56] W. Gordon, Sculptured surface interpolation via blending-function methods, Research Report, Department of Mathematics and Computer Science, Drexel University, Philadelphia, 1982.

[57] B. Goldstein, S. Kemmerer, C. Parks, A brief history of early product data exchange standard NISTIR 6221 , National Insitute of Standards and Technology, Sept 1998.

[58] I. Graham, W. Hackbusch, S. Sauter, Discrete boundary element methods on general meshes in 3D, Numer. Math. **86**, No. 1 (2000) 103–137.

[59] L. Greengard, V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimension, Acta Numerica **6** (1997) 229–269.

[60] G Greiner, Variational Design and Fairing of Spline Surfaces, Computer Graphics Forum 13(3), 1994, pp. 143–154.

[61] W. Hackbusch, Integral equations: theory and numerical treatment, International series of numerical mathematics 120, Basel: Birkhaüser, 1995.

[62] W. Hackbusch, C. Lage, S. Sauter, On the efficient realization of sparse matrix techniques for integral equations with focus on panel clustering, cubature and software design aspects, in: Proc. final conf. of priority research programme boundary element methods, Stuttgart, 1997, pp. 51–75.

[63] H. Harbrecht, R. Schneider, Biorthogonal wavelet bases for the boundary element method, Preprint SFB393/03-10, Technische Universität Chemnitz, Sonderforschungsbereich 393, 2003.

[64] H. Harbrecht, R. Schneider, Wavelets for the fast solution of boundary integral equations, Preprint SFB393/02-19, Technische Universität Chemnitz, Sonderforschungsbereich 393, 2002.

[65] P. Henrici, Applied and computational complex analysis, vol. 3, John Wiley & Sons, New York, 1986.

[66] F. Hill, Computer graphics using OpenGL, Prentice Hall, London, 2001.

[67] B. Holtz, The CAD rating guide, OnWord press, New Mexico, 1991.

[68] J. Hoschek, D. Lasser, Grundlagen der geometrischen Datenverarbeitung, Teubner, Stuttgart, 1989.

[69] B. Joe, Quadrilateral mesh generation in polygonal regions, Comput.-Aided Des. **27**, No. 3 (1995) 209–222.

[70] B. Joe, R. Simpson, Triangular meshes for regions of complicated shape, Int. J. Numer. Methods Eng. **23** (1986) 751–778.

[71] J. Jost, Riemannian geometry and geometric analysis, Springer, Berlin, 2000.

[72] M. Jungerman, G. Ringel, Minimal triangulations on orientable surfaces, Acta Math. **145**, No. 1-2 (1980) 121–154.

[73] D. Jungnickel, Graphs, networks and algorithms, Springer, Berlin, 1999.

[74] M. Kallmann, H. Bieri, D. Thalmann, Fully dynamic constrained Delaunay triangulations, in: Geometric modelling for scientific visualization, G. Brunnett, B. Hamann, H. Mueller, L. Linsen (Eds.), Springer-Verlag, Heidelberg, Germany, 2003, pp. 241–257.

[75] M. Keil, J. Snoeyink, On the time bound for convex decomposition of simple polygons, Int. J. Comput. Geom. Appl. **12**, No. 3 (2002) 181–192.

[76] M. Kilgard, The OpenGL Utility Toolkit (GLUT), programming interface API version 3, Silicon Graphics Inc., 1994.

[77] R. Knight, W. Valaski, AutoCAD-Referenz, IWT-Verl., München, 1993.

[78] C. Lee, S. Lo, A new scheme for the generation of a graded quadrilateral mesh, Comput. Struct. **52**, No. 5 (1994) 847–857.

[79] C. Livingston, Knot theory, The Carus mathematical monographs, The mathematical association of America, Washington, 1993.

[80] S. Lo, Generating quadrilateral elements on plane and over curved surfaces, Comput. Struct. **31**, No. 3 (1989) 421–426.

[81] D. Loffredo, Fundamentals of STEP implementation, STEP Tools, Rensselaer Technology Park, Troy, New York 12180, www.steptools.com.

[82] W. Massey, A basic course in algebraic topology, Springer, New-York, 1991.

[83] H. Mattson, Discrete mathematics with applications, John Wiley & Sons, New York, 1993.

[84] G. Meister, Polygons have ears, Amer. Math. Mon. **82** (1975) 648–651.

[85] R. Melosh, S. Utku, Estimating manipulation errors in finite element analysis II, Finite Elem. Anal. Des. **4**, No. 2 (1988) 163–173.

[86] A. Mezentsev, T. Woehler, Methods and algorithms of automated CAD repair for incremental surface meshing, in: Proc. eightth international meshing roundtable, South Lake Tahoe, California, 1999, pp. 299-309.

[87] F. Morgan, Riemannian geometry. A beginner's guide, Jones and Bartlett Publishers, Boston, 1993.

[88] M. Müller-Hannemann, K. Weihe, Minimum strictly convex quadrangulations of convex polygons, in: Proc. 13th ACM symposium on computational geometry, 193–200, 1997.

[89] J. Nocedal, S. Wright, Numerical optimization, Springer Series in Operations Research, New York, 1999.

[90] J. O'Rourke, Computational geometry in C, Cambridge Univ. Press, Cambridge, 1998.

[91] S. Owen, Non-simplicial unstructured mesh generation, Ph.D thesis, Carnegie Mellon University, 1999.

[92] J. Peraire, J. Peiro, K. Morgan, Advancing front grid generation, in: Handbook of grid generation, CRC Press, 1999, Chapter 17.

[93] L. Piegl, A menagerie of rational B-spline circles, IEEE Computer Graphics and Applications **9** (Sept. 1989) 48–56.

[94] L. Piegl, Infinite control points - a method for representing surfaces of revolution using boundary data, IEEE Computer Graphics and Applications **7** (March 1987) 45–55.

[95] L. Piegl, On NURBS: a survey, Computer Graphics & Application **11** , No. 1 (1991) 55–71.

[96] U. Pinkall, K. Polthier, Computing discrete minimal surfaces and their conjugates. Exp. Math. **2**, No. 1 (1993) 15–36.

[97] K. Polthier, Polyhedral surfaces of constant mean curvature, Habilitationsschrift, Technische Universität Berlin, 2002.

[98] H. Prautzsch, W. Boehm, M. Paluszny, Bézier and B-Spline techniques, Springer, Berlin, 2002.

[99] S. Ramaswami, M. Siqueira, T. Sundaram, J. Gallier, J. Gee, A new algorithm for generating quadrilateral meshes and its application to FE-based image registration, Proc. 12th International Meshing Roundtable, Sandia National Laboratories, 159–170, 2003.

[100] S. Ramaswami, P. Ramos, G. Toussaint, Converting triangulations to quadrangulations, Comput. Geom. **9**, No. 4 (1998) 257–276.

[101] M. Randrianarivony, G. Brunnett, Sufficient and necessary conditions for the regularity of planar Coons map, Sonderforschungsbereich 393, Preprint SFB393/04-07, 2004.

[102] A. Rathsfeld, Nyström's method and iterative solvers for the solution of double-layer potential equation over polyhedral boundaries, SIAM J. Numer. Anal. **32**, No. 3 (1995) 924–951.

[103] A. Rathsfeld, Iterative solution of linear systems arising from the Nyström method for the double-layer potential equation over curves with corners, Math. Methods Appl. Sci. **16**, No. 6 (1993) 443–455.

[104] S. Rees, L. Soicher, An algorithtmic approach to fundameental groups and covers of combinatorial cell complexes, J. Symb. Comput. **29**, No. 1 (2000) 59–77.

[105] W. Renz, VDAFS-a pragmatic interface for the exchange of sculptured surface data, in: Product data interfaces in CAD, CAM applications, edit. Encarnacao, Schuster, Vöge, Springer, Berlin, 1986, pp. 144–149.

[106] A. Riedel, The wavelet transform on Sobolev spaces and its approximation properties, Numer. Math. **58** (1991) 875–894.

[107] S. Saunders, T. Takaoka, Improved shortest path algorithms for nearly acyclic graphs, Theor. Comput. Sci. **293** , No. 3 (2003) 535–556.

[108] R. Schinzinger, P. Laura, Conformal mapping: methods and applications, Elsevier, Amsterdam, 1991.

[109] R. Schneider, Multiskalen- und Wavelet-Matrixkompression: Analysisbasierte Methoden zur Lösung grosser vollbesetzter Gleichungssysteme, Teubner, Stuttgart, 1998.

[110] L. Schoenberg, A. Whitney, On Polya frequency functions III, Trans. Amer. Math. Soc. **74** (1953) 246–259.

[111] G. Schulze, Blending-Function-Methoden im CAGD, Diplomarbeit, Universität Dortmund, 1986.

[112] H. Seidel, Computing B-spline control points using polar forms, Comput.-Aided Des. **23**, No. 9 (1991) 634–640.

[113] H. Seidel, Polar forms for geometrically continuous spline curves of arbitrary degree, ACM Trans. Graph. **12**, No. 1 (1993) 1–34.

[114] J. Stillwell, Classical topology and combinatorial group theory, Second edition, Springer, New York, 1993.

[115] R. Stöcker, H. Zieschang, Algebraische Topologie, Teubner, Stuttgart, 1988.

[116] B. Kehrer, G. Vatterott, CAD models and architectures, integration aspects of STEP and their expression in the CAD reference model, in: Modelling and graphics in science and technology, edts: J. Teixeira , J. Rix, 1996, pp. 1–20.

[117] L. Trefethen, Numerical computation of the Schwarz-Christoffel transformation, SIAM J. Sci. Stat. Comput. **1** (1980) 82–102.

[118] D. Trippner, Experience gained using the IGES interface for CAD/CAM data transfer, in: Product data interfaces in CAD, CAM applications, edit. Encarnacao, Schuster, Vöge. Berlin Springer, 1986, pp. 127–141.

[119] U. S. Product Data Association, Initial Graphics Exchange Specification. IGES 5.3, Trident Research Center, SC, 1996.

[120] M. Vanco, A direct approach for segmentation of unorganized points and recognition of simple algebraic surfaces, Ph. D. thesis, Technische Universität Chemnitz, 2003.

[121] T. Varady, P. Benko, G. Kos, Reverse engineering regular objects: simple segmentation and surface fitting procedures, Int. J. Shape Model. **4** (1998) 127–141.

[122] R. Verfürth, A Posteriori error estimators for the Stokes equations II: nonconforming discretizations, Numer. Math. **60** (1991) 235–249.

[123] B. Vidakovic, Statistical modeling by wavelets, John Wiley & Sons, New York, 1999.

[124] M. Vogel, P. Bunte, Pro/ENGINEER und Pro/MECHANICA, Carl Hanser Verlag, München, 2001.

[125] K. Weihe, T. Willhalm, Why CAD data repair requires discrete algorithmic techniques, in: Proc. WAE'98, Saarbrücken, Germany, 1998, pp. 1–3.

[126] U. Weissflog, Product data exchange; Design and implementation of IGES processors, in: Product data interfaces in CAD, CAM applications, edit. Encarnacao, Schuster, Vöge, Springer, Berlin, 1986, pp. 116–125.

[127] J. Wellington, Historical milestones, NIST, `www.eeel.nist.gov/iges/milestones.html`, 2003.

Index

Algorithm
    Adaptive Coons regularity, 104
    Boundary approximation, 135
    Breadth First Search, 155
    Coarse mesh generation, 129
    Convex polygon intersection, 81
    Diffeomorphism generation, 116
    Four-sided splitting, 61
    Meshing multiple patches, 133
    Prim or Kruskal, 60
    Quadrangulation, 38
    Single cut search, 53
Artificial vertex, 21

Bernstein polynomial, 98
Betti number, 175
Blending function, 94
Blossom, 102
    Symmetry, 102
Boolean sum, 93
Boundary approximation, 133
Boundary operator, 174
Boundary refinement, 69
Breadth First Search, 155

CAD interface
    CAD/CAM, 14
    Neutral format, 14
    Usage, 13
CAD repair, 24
Chain, 174
Cleanup, 56
Coarse mesh, 128
Collocation, 2
Commutator, 179
Conforming splitting, 32
Coons patch, 93
Cut search, 49

Data structure, 22
Degree elevation, 105
Double layer operator, 1
Double-edged polygon, 39

Edge flipping, 128
Edge size function, 131
Edge splitting, 127
Euler characteristic, 179

Fast multipole, 2
Fundamental group, 180

Galerkin, 2
Generalized Delaunay, 127
Genus, 179
Graph
    Breadth First Search, 155
    Canonical curve, 159
    Mimimum Spanning Tree, 60
    Sibling, 60
    Smooth boundary approximation, 133
    Spanning tree, 58, 155
    Weighted split graph, 159

Harmonic function, 125
    Laplace-Beltrami, 122
    Mean value property, 125
Harmonicity, 140
Homology group, 175

IGES
    Brief history, 14
    CAD flaw, 24
    Data structure, 22

  - Delimiter, 18
  - Entity, 18
  - IIF (Internal IGES file), 18
  - Implementation, 21
  - Section, 18
  - Shortcomings, 18
  - Structure, 15
- Incidence number, 174
- Integral equation, 1
  - Mesh free method, 3
  - Mesh-based method, 4

- Kernel, 33

- Loop, 180
  - Equivalent, 180
  - Product, 180

- Mean value property, 125
- Meister, 35, 135
- Mesh conversion, 58
- Mesh free method, 3
- Mesh generation, 121
  - Coarse mesh, 128
  - Edge size function, 122
  - Generalized Delaunay, 127
  - Laplace-Beltrami, 122
  - Nonsmoothness, 129
  - Planar case, 124
- Mesh merging, 133
- Mesh-based approach, 4
- Multiresolution, 3
  - Dual, 3
  - Primal, 3

- Nonsmoothness, 129
  - First kind, 129
  - Second kind, 129
  - Third kind, 130
- Nyström, 2

- Panel clustering, 2
- Partition of unity, 98
- Poisson problem, 125
- Polygon
  - Double-edged, 39
  - Intersection, 80
  - Kernel, 33
  - Vertex visibility, 76
  - Wedge, 33
- Polygonal approximation, 61
- Polygonal disk, 177

- Quadrangulation
  - Conversion into convex, 54
  - Convertion from triangulation, 58
  - Hexagon, 78
  - Multiply connected, 46
  - Non-convex quadrilateral, 76
  - Pentagon, 77
  - Simply connected, 38
- Quadrilateral removal, 35

- Region
  - Multiply connected, 61
  - Simply connected, 35
- Regularity, 94
  - Adaptivity, 104
  - Necessary condition, 100
  - Performance, 106
  - Sufficient condition, 95, 99
  - Uniform subdivision, 104
- Reidemeister moves, 153, 167, 181

- Schoenberg-Whitney, 165
- Segment separator, 21
- Segment-rectangle intersection, 52
- Simplex, 173
- Simplicial complex, 173
- Spanning tree, 60
- Splitting
  - Boundary refinement, 69
  - Conforming, 32
  - Simple n-gons, 76
  - Special splittings, 75
- Steiner point, 58
- Stiffness matrix, 133
- Subdivision, 101
- Surface structure, 24
  - Metric structure, 26

  - Topological structure, 24

- Topological surface, 177
  - Classification, 179
  - Genus, 179
  - Gluing, 178
- Transfinite interpolation, 92
  - Blending function, 93
  - Boolean sum, 93
  - Overspill phenomenon, 94
  - Regularity, 100
- Trimmed surface, 32
- Two-ear theorem
  - Application, 35
  - Double-edged polygon, 40
  - Statement, 35

- Vertex
  - G^1-vertex, 72
  - Reflex, 34

- Wavelet Galerkin scheme, 3
- Wedge, 33

THESES

Thesis 1: A quadrangulation can be generated by repeatedly removing quadrilaterals

Many interesting theoretical results related to quadrangulation hold for simply connected and multiply connected polygons. For simple polygons, either one can chop off a quadrilateral which is not necessarily convex by introducing a cut connecting two boundary vertices or one can remove a convex quadrilateral by inserting an internal Steiner point. The proof of that fact uses the famous 2-ear theorem. Since that theorem fails to hold for multiply connected polygons, the notion of double-edged polygons is introduced. For multiply connected polygons, a method of searching for cuts is required in order to obtain double-edged polygons. Thanks to the version of the 2-ear theorem for double-edged polygons, the results about simple polygons can be generalized for double-edged ones. Our quadrangulation method has the property that $\mathcal{O}(n)$ quadrilaterals are obtained from a polygon having n vertices. Since that method of generating a quadrangulation may fail to give convex quadrilaterals, a convertion from a nonconvex quadrangulation to a convex one is required. That is done by combining two neighboring quadrilaterals in order to form a hexagon. After quadrilating the hexagon by using only internal Steiner point, the two quadrilaterals are replaced by the quadrangulation from the hexagon. Cleanup operations are used to enhance the quality of a given quadrangulation. Our cleanup operations consist only in shifting a node or flipping an edge. That is, the numbers of nodes and edges remain unchanged when a cleanup operation is applied.

Thesis 2: The above quadrangulation yields an approach for decomposing a set of surfaces having curved boundaries into four-sided subdomains

The basic method of splitting a domain having curved boundaries into four-sided subregions consists in having first a polygonal approximation. After quadrilating the resulting polygon by using the above method, one replaces every boundary edge of the quadrangulation by the corresponding curve portion. In order to be able to apply polygonal approximations without hanging nodes to a set of surfaces, the adjacency information has to be taken into account. For a closed surface having multiple faces, the number of odd polygonal approximations is proved to be even. In order that a polygon can be quadrilated, the number of boundary vertices has to be even. Thanks to the adjacency graph, all approximating polygons can be converted into even ones. On the other hand, the process of replacing a straight boundary edge by a portion of curve might introduce intersections with straight internal edges of the quadrangulation. We present an approach to get rid of such boundary interference. Since the quadrangulation is only repaired in the neighborhood of the curve portion which intersects an internal edge, most part of the quadrangulation is kept unchanged. A very undesired situation is to have a four-sided domain which has a G^1 vertex. That is, the internal angle at a corner of a four-sided domain is π. A quadrangulation method guarantees that an edge emanates from every G^1 vertex.

Thesis 3: Sufficient conditions exist for checking diffeomorphism about planar Coons patches

For given four curves delineating a four-sided domain, a method is needed to test if the corresponding Coons map is a diffeomorphism. If the given curves are represented in Bézier form, two sufficient conditions can be introduced. The first condition which is related to the tangents at the boundary curves can be expressed with the control points of the bounding curves. It is very easy to check but it is not very robust in practice. With the help of an example of curves whose curvature can be controlled by some parameter, we show that the first condition fails to give any answer when the curves deviate significantly from straight lines. The second method requires the expression of the Jacobian in Bézier form with some degree elevation. Verifying the second method is more computationally expensive but it performs much better than the first method.

Thesis 4: An adaptive algorithm about checking diffeomorphism of Coons patches can be derived from blossoming techniques

For the verification of whether a planar Coons map is a diffeomorphism, a condition which is both efficient and robust is found. In fact, from the blossoming strategy, a condition which possesses those two objectives is proved. As opposed to the above two conditions, it turns out that this condition based on blossoming is not only sufficient but also necessary for identifying the diffeomorphism. From that theory, one can derive an adaptive algorithm by using subdivision techniques. It is not necessary to subdivide the domain $[0,1]^2$ everywhere. We need only to subdivide it at positions where the condition is difficult to check. Thanks to that property, the adaptive algorithm performs well even for domains where the bounding curves are very complicated. Apart from theoretical predictions, different numerical examples show the efficiency and robustness of the adaptive algorithm .

Thesis 5: Gordon patches can be used to remove overspill phenomena

The problem related to the direct application of the Coons patch is that the resulting function may have overspill phenomena. That is, there are intersections between isolines which give some wrinkles in the surface. In order to get rid of the overspill phenomena, Gordon patches are used. This has the advantage of being able to interpolate given internal points and curves. The choice of some internal points and curves are done by using Floater parametrization techniques and a method which minimizes the Dirichlet energy. It is possible that the bounding curves are so complicated that it is very difficult to find the internal curves. In such cases, we have to subdivide the four-sided domain. Additionally, the results about Coons patches can be generalized to Gordon patches, if the blending functions are well chosen.

Thesis 6: The IGES format is a suitable CAD interface for the implementation of geometric preprocessing for integral equations

The IGES format is the CAD interface that is used to store geometric information in our work. There are two types of integral equation solvers: mesh-free and mesh based approaches. For both approaches, the IGES format is suitable for the storage of CAD data. Implementation with IGES format is a tedious work which include the following tasks. First, routines for location of information related to the the stored geometry have to be assembled. Since the IGES structure varies from one IGES entity to another, one set of routines for the loading of the record related to each entity has to be constructed. Since it is very difficult to treat all entities of the IGES format, a few selected entities have been implemented. Those special entities can represent interesting mechanical CAD data. Furthermore, generation of a data structure for each IGES entities should exit. As a consequence, one needs a conversion from the IGES records to those data structure. Good data structures are required in order to have good coherence between the components of the stored geometry. Finally, an evaluation routine for each important data structure is needed in order that the data in IGES files can be used with our geometric algorithm. Apart from the withdrawal of geometric information from IGES files, the topological structure and the metric information have to be generated. Having them helps in geometric algorithms which needs adjacency information. Those algorithms include the polygonal approximation without hanging node and the conversion of odd faces into even ones within a closed surface.

Thesis 7: The generalized Delaunay triangulation can be used to generation of a mesh from CAD data

The preprocessing of CAD geometry for subsequent application in numerical method for mesh-based integral equation solvers requires the generation of surface meshes. In order to generate a mesh from surfaces stored in CAD data, the generalized Delaunay triangulation proves suitable. The methods consists in starting from a coarse mesh which is refined repeatedly by using Delaunay splitting and flipping methods. In order to be able to split an edge, the knowledge of the ideal edge size is required. Unfortunately, the ideal edge size is unknown a-priori. So as to determine it, the Laplace-Beltrami operator is used. For CAD data having multiple trimmed surfaces, that technique is applied to each one of them. In order that there exists no hanging node, the boundaries of the trimmed surfaces are discretized before the process of mesh generation.

Thesis 8: A three-step approach exists for the $\mathcal{C}^0$-paving of a triangular mesh into quadrilateral patches

In the thesis, approximation of an orientable closed surface mesh by a set of B-spline patches was investigated. The first step consists in finding curves which are drawn on the surface such that by slicing the mesh along those curves, a single connected piece of surface remains. First, a parametrization technique is applied in order to find a mapping from a planar polygonal domain to the sliced mesh. Afterwards, the mesh is split into several four-sided submeshes. Since the goal in the surface fitting is to obtain global continuity, we approximate the bounding piecewise linear curves first by B-spline curves. The boundary de-Boor points of the B-spline surfaces are determined by interpolating those bounding B-spline curves. The final step in the approximation consists in determining the internal de-Boor points in a least-square method.

LEBENSLAUF

Persönliche Daten:

Vor- und Zuname : Maharavo H. Randrianarivony
Geburtsdatum : 07. August 1976
Geburtsort : Befelatanana, Madagascar
Familienstand : ledig
Wohnort : Klarastrasse 30
09131 Chemnitz
E-Mail : maharavo@informatik.tu-chemnitz.de

Studium:

1999-2001 : Mathematik mit Nebenfach Informatik.
Technische Universität Chemnitz.
Abschluss mit "Master of Science".
1997-1999 : Reine und Angewandte Mathematik.
Université d'Antananarivo.
Abschluss mit "Attestation d'Études Approfondies".
1993-1997 : Reine und Angewandte Mathematik.
Université d'Antananarivo.
Abschluss mit "Maitrise en Mathématiques".

Schule:

1985-1993 : Lycée Paul Minault. Série scientifique.
Abschluss mit "Baccalaureat".
1980-1985 : Grundschule Ambohimandroso.

Berufstätigkeit:

1999-2001 : Wissenschaftlicher Mitarbeiter an der Fakultät für Mathematik der TU Chemnitz.
2001-Jetzt : Wissenschaftlicher Mitarbeiter an der Fakultät für Informatik der TU Chemnitz.

VERSICHERUNG

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegeben Hilfsmittlel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Weitere Personen waren an der geistigen Herstellung der vorliegende Arbeit nicht beteiligt. Insbesondere habe ich nicht die Hilfe eines Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistugen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

BIBLIOGRAPHISCHE BESCHREIBUNG UND REFERAT

Geometric processing of CAD data and meshes as input of integral equation solvers.

Technische Universität Chemnitz, Fakultät für Informatik,
Dissertation, 2006, 182 pages

In diesem Dokument interessieren wir uns für die Vorbereitungen von CAD oder geometrischen Daten damit die mit Löseren von Integralgleichungen passen. Zu der Nutzung von Geometrien für mesh-basierten Integralgleichungen verwenden wir eine Methode, die auf einer generalisierten Delaunay Triangulierung basiert ist, um ein Mesh von einer Oberfläche zu erzeugen. Damit wir so eine Triangulierungsmethode auf den Riemanntensor anwenden können, verwenden wir eine Annäherung von der Kantenlägenfunktion, die den Laplace-Beltrami Operator benutzt. Anderseits, wurden Verfahren entwickelt und implementiert um die Oberflächen in viereckigen Gebieten zu zerlegen. Wir zeigen Methoden um Polygonen in konvexen Vierecken aufzuteilen. Wir versuchen die Zerlegung des getrimmten Parametergebietes mit wenigen Teilgebieten zu erhalten. Wir berichten unsere praktischen Erfahrungen über die Implementierung mit Hilfe von der CAD Schnittstelle IGES. Um Integralgleichungen die auf Wavelet Verfahren basiert sind umsetzen zu können, braucht man Diffeomorphismus von dem Einheitsquadrat. In dem Zusammenhang, haben wir Methode entwickelt um Diffeomorphismus mit Hilfe von transfiniten Interpolationen zu behandeln. Begrenzungskurven in Bézierform sollen das verwendete Diffeomorphismuskriterium darstellen. Anschliessend zeigen wir Methode um ein Mesh mit Hilfe von stückweise B-spline patches zu approximieren.

Stichworte: CAD, IGES, integral equation, quadrangulation, surface approximation, mesh generation, transfinite interpolation, diffeomorphism, four-sided splitting.