

Quad Layout Generation and Symmetric Tilings of Closed Surfaces

*Beim Fachbereich Mathematik und Informatik der Freien Universität Berlin
eingereichte Dissertation von*

Faniry Harijaona Razafindrazaka

2015

Referees:

Prof. Dr. Konrad Polthier, Freie Universität Berlin, Germany

Prof. Dr. Pierre Alliez, Inria Sophia-Antipolis, France

Defended on February 9, 2016

Selbständigkeitserklärung

Gemäß §7 (4) der Promotionsordnung versichere ich hiermit, diese Arbeit selbständig verfasst zu haben. Ich habe alle bei der Erstellung dieser Arbeit benutzten Hilfsmittel und Hilfen angegeben. Diese Arbeit habe ich in keinem früheren Promotionsverfahren eingereicht.

ACKNOWLEDGEMENT

I thank God for giving me the strengths to achieve until the very end the development of this thesis. Prof. Konrad Polthier as my supervisor, I thank him for his valuable support. My full gratitude goes to Ulrich Reitebuch for all these years of mathematical discussions until the final realization of this thesis. Thank you Prof. Pierre Alliez for accepting without hesitation to review my thesis. I thank my family for their everyday assistance mentally with perseverance. To all members of the mathematical geometry processing group, I would have not make it this far without your assistance. Sandra Ramanantoanina, thank you for always supporting me with love and patience.

My research has been funded for 3 years by the Berlin Mathematical School and for 1 year by the SFB TRR109, Discretization in Geometry and Dynamics.

Contents

1	Introduction	1
1.1	Main Contributions	3
1.2	Previous Works	5
1.3	Structure	7
2	Parameterizations and Quad Layouts	8
2.1	Surfaces and Parameterizations	8
2.2	Base Complexes and Quad Layouts	13
3	Matching Theory	15
3.1	Classical Matching Problem	16
3.2	Disjunctive Constraints	16
3.3	Quad Layouts and Matchings	17
4	Graph Construction	19
4.1	Motorcycle Growing	20
4.2	Singularity-free Triangles	20
4.3	Edge Weights	22
4.4	Boundary	23
4.5	Matching	24
4.6	Illegal Crossings	24
5	Global Optimization	29
5.1	Binary Program Formulation	29

5.2	Complexity	32
6	Application I: Quad Layout on Triangle Meshes	35
6.1	Simplicial Surfaces	36
6.2	Function Spaces on Simplicial Manifolds	37
6.3	Field Curl on Simplicial Surfaces	37
6.4	Curl-free vs Non-curl-free	40
6.5	Injectivity	40
6.6	Curl Minimization: Poisson Parameterization	41
6.7	Graph Construction	44
6.8	Stopping Criterion	44
6.9	Results and Analysis	46
6.10	Comparison	48
7	Application II: Quad Mesh Structure Optimization	53
7.1	Problem	54
7.2	Preliminary	55
7.3	Algorithm Overview	55
7.4	Graph Properties	56
7.5	Results and Analysis	57
7.6	Comparison	60
8	Reparameterization	65
8.1	Problem Formulation	66
8.2	Mapping the Quads	67
8.3	Examples	67
9	Regular Maps	69
9.1	Background Notions	71
9.2	Generating Large Genus Surfaces	73
9.3	Identification Algorithm	76

9.4 Regular Surfaces	80
9.5 Group Structure	82
9.6 New Realizations	85
10 Extensions and Future Works	91
10.1 T-Layout to Quad Layout	92
10.2 Hexahedral Mesh Simplification	95
11 Conclusions	97
Appendix A Energy minimization	100
Appendix B Solver Comparison	103
B.1 Input Models	104
B.2 Statistics	104
Zusammenfassung	111

CHAPTER 1

INTRODUCTION

In recent years, the digital processing of real world objects has become central in geometry processing and computer graphics. The need of efficient discretizations of digitalized shapes has grown exponentially in order to perform accurate analyses as well as robust simulations. Traditionally, computer aided designers have been using primitives such as cubes, cylinders, or planes until the recent development of 3D scanners which allows a fast generation and a robust approximation of free-form surfaces represented as *triangle meshes* or in a more scientific context, *simplifical manifolds*. Triangle meshes which were for longtime used for finite element analysis have become the main foundation of modern geometry, giving birth to the field of *discrete differential geometry*. In other branch of computer graphics such as car design, computer animation or renderings, *quadrilateral meshes* or *quadrilateral patch layouts* are preferred due to their high level representations of the underlying surface. Several algorithms have been proposed to automatically convert dense triangle meshes into uniform quadrilateral meshes but very little is known about the global structure of the generated quad meshes. Mainly, the generation of coarse intrinsic partitions of the surface into all quadrilateral patches has not been addressed until very recently.

This thesis is about tiling of surfaces. The main part of the thesis treats problems related to quadrilateral patch layout generation while the second part proposes a solution of the problem of visualizing regular maps. Given a prescribed highly symmetric tiling, we model a closed surface on which this tiling preserves as much as possible its symmetry.

Quad Layouts Quadrilateral patch layouts or *quad layouts* are partitions of manifold surfaces into non-overlapping quadrilateral patches such that any two patches

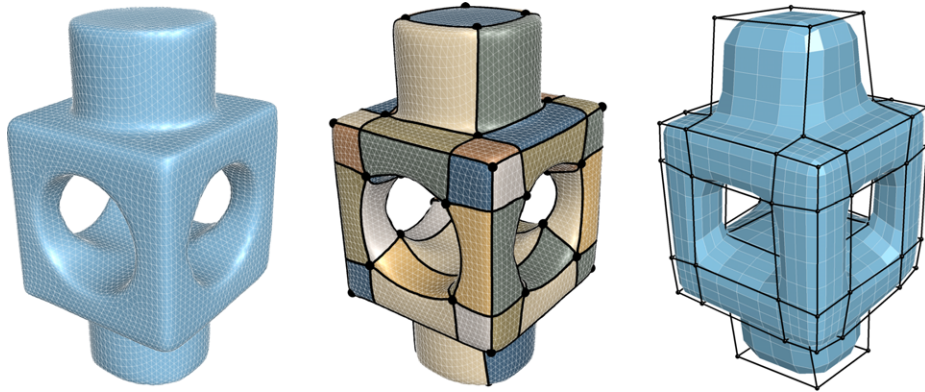


Figure 1.1: Triangle mesh, quadrilateral patch layout and a hierarchical quad mesh representation.

share an edge or a vertex or are connected by a chain of adjacent patches. Quad layouts are practically useful in contexts such as high order surface fitting, coarse base mesh for subdivision surfaces, hierarchy in finite element analysis, surface compression, architectural design, and many more. Coarseness and alignment to geometric features of the surface are very important for the generated patches. Finding a good balance between these two quantities is one of the achievements of this thesis.

The automatic generation of pure quadrilateral patch layouts on manifold meshes is a mixed topology and geometric problem. The geometry of the surface is captured by a guiding frame field derived by local features of the surface such as principal curvatures, while the topology is encoded in the singularities of this field. Finding a quad layout of a surface is equivalent to finding a graph of the singularities as nodes where patches bounded by intersecting edges are quads. Our approach considers the graph nature of the problem. It is based on a careful construction of a graph guided by a given input frame field. We derive a quadrilateral patch layout of the surface as a minimum weight perfect matching with disjunctive constraints of that graph. The resulting layout is optimal relative to a balance between coarseness and geometric feature alignment. The main advantage of the new method is its simplicity and its computation speed.

Regular Map A Regular map is an algebraic concept to describe most symmetric tilings of closed surfaces. All regular maps resp. symmetric tilings of surfaces up to genus 302 are algebraically known in the form of symmetry groups acting on their universal covering spaces. But still little is known about geometric realizations, i.e. finding most symmetric embedding of closed surfaces and a supported most

symmetric tiling. The Platonic solids are particular cases of regular maps with genus zero. The visualization of high genus regular maps is a challenging problem. In this thesis, we introduce the concept of regular surfaces which are high genus surfaces with well defined group structures. Regular surfaces are geometrically defined as space models of regular maps. In contrast to [vW09], they do not necessarily depend on a sequence of lower genus regular maps to exist. The construction can be directly made from the hyperbolic space.

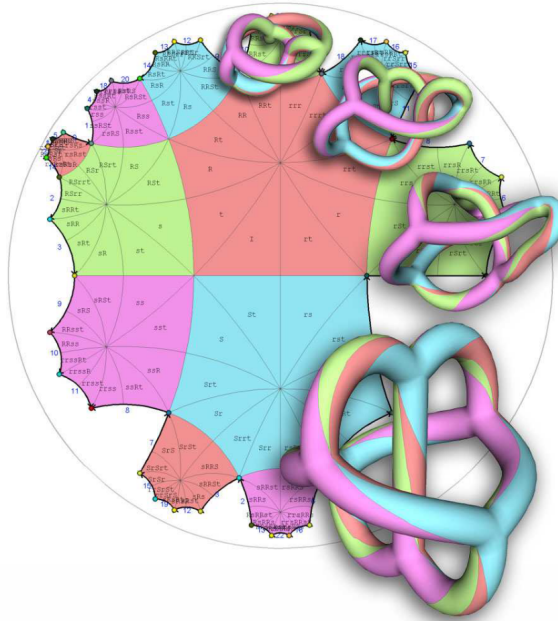


Figure 1.2: A realization of a hyperbolic regular map as a closed surface.

1.1 MAIN CONTRIBUTIONS

The contributions of this thesis are divided into two parts: automatic generation of high quality quad layouts for manifold meshes and symmetric tiling of closed surfaces or visualization of regular maps.

Quad Layouts

- We propose a new formulation of the quad layout problem into a perfect matching problem which is a global optimization problem.

- We introduce a novel singularity graph which is based on the concept of singularity-free triangles and whose perfect matching subgraphs are layouts of the surface.
- We find a simple linear inequality constraints to identify quad-only matching subgraphs of the singularity graph by analyzing the local crossings of the edges.
- We give a complete proof of the existence of an optimal quad layout for quadrilateral meshes and a weak proof for triangle meshes.
- We give a conceptual extension of the proposed algorithm to non-conforming meshes (e.g T-meshes) and hexahedral mesh simplification.

Regular Maps

- We introduce the concept of *Regular surfaces* which are high genus surfaces with well defined group structures.
- We propose new realizations of regular maps ranging from genus 9 to 85 taking advantage of the wide range of high genus surfaces using regular surfaces.
- By using a branch covering approach, we are able to design very symmetric space models of some regular maps inherited from the symmetry of the Platonic solids.

Publications Parts of this thesis appear in the following papers:

- “*Perfect Matching Quad Layouts for Manifold Meshes*”
Eurographics Symposium on Geometry Processing, Graz 2015 [RRP15]
- “*The 6-rings*”
In Proceedings of the Bridges Conference, Entschede 2013 [RP13]
- “*Regular Surfaces and Regular Maps*”
In Proceedings of the Bridges Conference, Seoul 2014 [RP14]
- “*Visualizations of Regular Maps of Large genus*”
Chapter submitted to the book Topological and Statistical Methods for Complex Data, 2015 [RP15]

Figure 9.1 in Chapter 9 appears on the cover of the proceeding of [RP14] and Figure 9.11 in Chapter 9 is chosen as the cover of the book [RP15].

1.2 PREVIOUS WORKS

The literature on field aligned global parameterizations in general and quad layout in particular is vast. A complete survey can be found in [BLP⁺12]. Our work is based on several previous results using cross fields and line tracing techniques on surfaces.

Field Aligned Global Parameterization One of the first field-aligned global parameterizations is the periodic global parameterization of Nicolas Ray et al. [RLL⁺06], which was later on improved by using a branch covering based approach [KNP07] or a mixed integer formulation [BZK09]. Since these methods rely heavily on good input frame fields, recent works [KCPS13, PPTSH14, ECBK14, DVPSH15a] focus on the generation of good frame fields. This class of parameterizations is not in general injective such that more advanced convex constraints or iterative optimization methods are required. Lipman [Lip12] introduces the bounded distortion mapping which explores the maximum deformation of a triangle during the parameterization. The trisector constraint proposed in [BCE⁺13] uses the Fermat point of a triangle to generate simple convex constraints per triangle. The non linear iterative optimization of [DVPSH15a] aims for curl free non symmetric poly-vectors which induces a flip-free parameterization.

Global parameterization methods are practically useful for quad meshing. Unfortunately, they do not always provide a high-level structure or coarse base mesh of the surface. Other recent works concentrate on the direct generation of coarse quad layouts for a given surface.

Quad Layouts In the past years, several methods have been introduced to generate high quality quad layouts on surface meshes. These include: direct algorithms, spectral methods, parameterization approaches, or field tracing techniques. Direct algorithms work directly on the triangulation of the surface without any high order geometric information. Matthias Eck [Eck96] uses triangle pairing techniques for the purpose of B-spline fitting; a Voronoi partition approach is used in [BMRJ04]; Daniels [DSC09] proposes to quadrangulate the surface by a one step Catmull-Clark subdivision, then uses quad mesh simplification techniques to get

a coarse base mesh. Spectral methods use a Morse-Smale complex of the Laplacian eigenfunctions [DBG⁺06]. Field-aligned parameterization can also generate coarse quad layouts by using large target edge lengths with a quadratic integer optimization [BCE⁺13]. Other methods generate initially a fine integer-grid map and build a T-mesh layout [MPKZ10] or optimize the base complex of the induced quad mesh [BLK11, TPP⁺11]. Recently, several works focus on direct field tracing. Campen et al. [CBK12] trace anisotropic loops to build from scratch the dual of the base complex, followed by a layout optimization [CK14]. Myles et al. [MPZ14] and Ray et al. [RS14] propose, independently, a robust polyline tracing of rotational symmetric fields. Their techniques produce a patch layout of the surface using the Motorcycle graph algorithm of Eppstein et al. [EGKT08].

Our approach differs from [MPZ14] in two aspects. First, we trace isolines of a parameterization induced from a curl free frame field which avoids the existence of limit cycles. Second, an isoline tracing does not stop when it intersects another one. Instead we evaluate the ratio of the arc lengths of the two isolines from their intersection points to their respective singularities. Accordingly, we construct a graph \mathcal{G} defining the space of possible layouts of the surface.

Regular Maps Up to now, there is no general method to visualize regular maps but a lot is already known about their symmetry group, see for example Conder [CD01]. The problem is two-fold, understanding the symmetry group of the regular map and finding a suitable space model for the realization of this group. Jack van Wijk [vW09], in his *SIGGRAPH* paper, suggested a generic approach which gives interesting visualization of some of the lower genus regular maps up to genus 29. He succeeds to handle about 50 cases by using a brute force computer search. However, his method has some limitations and cannot realize even some of the simplest cases. Séquin’s investigations [S07], [S10] are also a huge source of inspiration. He uses physical modeling techniques, including sketches, paper models, and Styrofoams to finally obtain a computer generated model. Some cases have been solved by his method from genus 2 to genus 5 but each regular map is handled separately. Sequin’s approach is useful for a better understanding of the structure of regular maps but too primitive to handle the large ones. In our early work [Raz12, RP13], we use the same approach as van Wijk, but we added a relaxation procedure to obtain more symmetrical and smooth tubular geometries. We use this relaxation scheme as a second step of the targetless tubification algorithm proposed in this thesis. We aim at surfaces having more than two junctions and with rich structures to accommodate regular maps. We are not interested in the hosohedral kind of surface. These are the surfaces obtained by taking the tubular

neighborhood of Hosohedra.

1.3 STRUCTURE

The thesis is subdivided in three parts. In the first part, containing Chapter 3, 4 and 5, we expose the theoretical foundation of our algorithm applied to general parameterized surfaces. These include, multi-chart parameterization, matching theory, and the formulation of the quad layout problem into minimum weight perfect matching problem. In the second part, we apply our algorithm to two examples of discrete surfaces, triangle meshes, and quadrilateral meshes. These are given in Chapter 6 and 7. The third part is our results on the symmetric embedding of regular maps. We introduce the concept of regular surfaces. We present in Chapter 9 the key ingredients of our construction. Chapter 10 proposes conceptual extensions of the perfect matching quad layout algorithm to other types of meshes such as T-meshes or hexahedral meshes.

CHAPTER 2

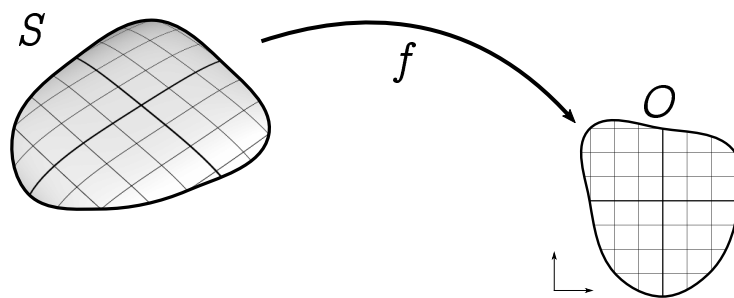
PARAMETERIZATIONS AND QUAD LAYOUTS

Parameterization is the process of mapping a surface to a 2D domain. All known concepts in 2D Euclidean geometry can then be transferred onto the surface by the inverse mapping. Quantities such as angles, lengths and areas could be measured on the 2D space instead of on the complicated 3D shape. In this Chapter, we give a mathematical definition of a parameterization together with the notion of quad layout.

2.1 SURFACES AND PARAMETERIZATIONS

Definition 2.1. A *parameterized surface element* is an injective map $f = (u, v)$ from a closed region S to a closed set $O \subset \mathbb{R}^2$.

$$f = (u, v) : p \in S \mapsto (u(p), v(p)) \in O \subset \mathbb{R}^2 \quad (2.1)$$



f is called a *parameterization*, O is called the *parameter domain* and $u, v : \mathbb{R}^2 \rightarrow \mathbb{R}$ are called the *parameter functions*. The 2D cartesian coordinates of O is pulled

back on S via f^{-1} . *Parameter lines* on O becomes *parameter curves* on S the same as straight lines on O becomes *geodesics* on S . In standard differential geometry textbooks, a parameterization is defined by f^{-1} . The choice is application dependent. In theory, f^{-1} is given as a multivariate function where in practice it is more suitable to look for f (e.g. in surface texture mappings). Also, f is not necessarily open such that $\partial O = f(\partial S)$. The use of closed region is more suitable in practice than open ones.

The *Jacobian* matrix of f at a point p is the gradient of the coordinate functions $(\nabla u|_p, \nabla v|_p)$ which defines a basis of the tangent plane at p . The set of all frames $\{(\nabla u|_p, \nabla v|_p)\}_{p \in S}$ defines a *frame field* on S .

The distortion of f is measured by comparing the frame directions at each point with the standard unit frame. For example,

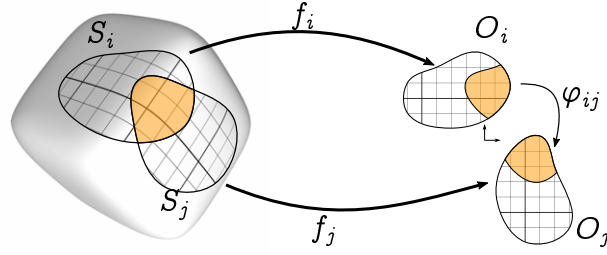
- f is *isometric* if $\langle \nabla u|_p, \nabla v|_p \rangle = 0$ and $\|\nabla u|_p\| = \|\nabla v|_p\|$ for all p , i.e. measuring lengths and angles are the same on S and O .
- f is *conformal* if $\langle \nabla u|_p, \nabla v|_p \rangle = 0$ for all p , i.e. measuring angles are the same on S and O but not the lengths.
- f is *area preserving* if $|\nabla u|_p \times \nabla v|_p| = 1$ for all p , i.e. measuring the area of patches are the same on S and O but not the lengths or angles.

Definition 2.2. A *parameterized surface* is a set \mathcal{M} with a family of parameterized surface elements $\{(S_i, f_i)\}_{i \in I}$ which satisfies the following conditions:

1. $\mathcal{M} = \cup_{i \in I} S_i$.
2. if $S_i \cap S_j \neq \emptyset$ and is simply connected or is a connected 1D curve, then there exists a transformation φ_{ij} such that $f_j(S_i \cap S_j) = \varphi_{ij} \circ f_i(S_i \cap S_j)$.

The set $\{f_i\}_{i \in I}$ defines a *global parameterization* of \mathcal{M} , $\{(S_i, f_i)\}_{i \in I}$ is called an *atlas*, the S_i 's are called *charts* and the φ_{ij} 's are called *transition functions*. Together, they define a *manifold* structure on \mathcal{M} which is not necessarily differentiable. In this definition, charts are allowed to intersect on boundary curves. Surfaces which are only "chartwise" smooth such as polyhedral surfaces are also included. To define a proper quadrangular parameterization, the φ_{ij} 's are combination of rotations, translations and scalings. More precisely,

$$\varphi_{ij}(p) = J^{r_{ij}}(p) + \mathbf{t}_{ij}, \quad \forall p \in f_i(S_i \cap S_j) \quad (2.2)$$



where J is a 90° rotation matrix, r_{ij} is called the *rotation index* (also called *matching*) which is an integer modulo 4 and $\mathbf{t}_{ij} \in \mathbb{R}^2$ is a translation vector from the set O_i to O_j . For the case of no rotations ($r^{ij} = 0$ for all i, j), the manifold is called *affine*. The classical parameterization of a torus is an affine manifold. In this definition, there is no assumption about the continuity of parameter lines in the charts' intersections. The value of \mathbf{t}_{ij} and the position of some special points of \mathcal{M} determine the property of the parameter lines in each chart. If $\mathbf{t}_{ij} \in \mathbb{Z}^2$ and the singularities (see Definition 2.3) are mapped onto 2D grid points (integer coordinates), then φ_{ij} is a *grid automorphism* which maps parameter grid lines to parameter grid lines. This class of parameterization is a *seamless continuous parameterization* that we call *closed parameterization*. It defines a quadrilateral grid of \mathcal{M} . For the general case, parameter grid lines are not necessarily mapped to parameter grid lines by some φ_{ij} . We call these parameterizations *open parameterization* or *seamless discontinuous parameterizations*. Some isolines of open parameterized surfaces have infinite lengths while they are either loops or parameter lines connecting two singularities on closed parameterized surfaces.

Remark 2.1. On a global parameterized surface, we can only define locally a uv -coordinate. If the rotation index r_{ij} is for example odd, then a u parameter line of S_i is mapped to a v parameter line in S_j .

Definition 2.3 (*Valence, Regular and Singular points*). On a parameterized surface, the *valence* of a point is the number of isolines intersecting at that point.

Consider a surface element $S_i \subset \mathcal{M}$, $p \in S_i$ a point of \mathcal{M} , and $q = f_i(p)$ the corresponding parameter point in O_i . We denote by α_q^i the inner angle of the vectors (left and right) tangent to the boundary curve of the parameter domain O_i at q . For all $i \in I_0 \subset I$ such that $q \in \partial O_i$ if

$$\sum_{i \in I_0} \alpha_q^i = 2\pi \text{ then } p \text{ is regular.}$$

$$\sum_{i \in I_0} \alpha_q^i = k\frac{\pi}{2} \text{ for } k \geq 1 \text{ and } k \neq 4, \text{ then } p \text{ is singular.}$$

If $p \in \tilde{S}_i$, i.e. strictly inside S_i , then p is always regular.

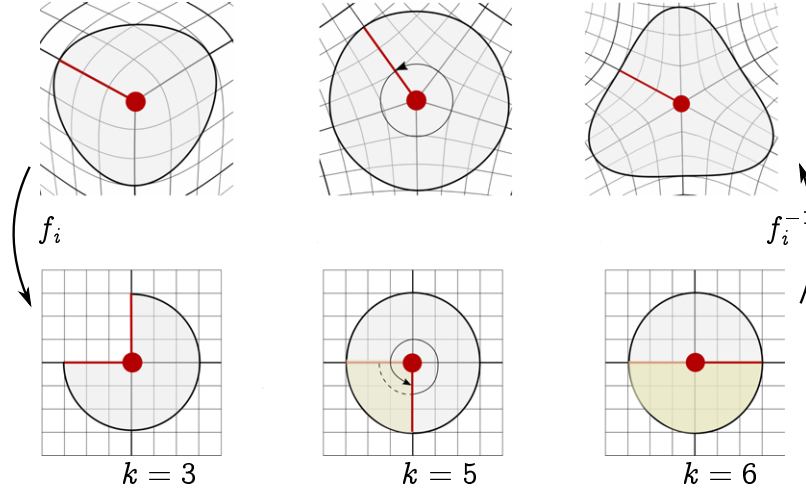


Figure 2.1: Examples of valence 3, 5 and 6 singular points.

Singular points of \mathcal{M} have valence $k \neq 4$. In Figure 2.1 are examples of singularity points with valence three, five and six. If the valence of a singularity is less than four, then a disc neighborhood of the point on \mathcal{M} is mapped to an incomplete disc in parameter domain taking transition functions between charts into account. If it is greater than four, then the disc neighborhood is mapped to a spiraling type disc in parameter domain. The gradient frame field is not defined at singular point but only in a small neighborhood. Singularities can be characterized with respect to the induced gradient field $\{(\nabla u|_p, \nabla v|_p)\}_p$ of the parameterization by constructing a small loop around each point and finding the *holonomy* angle of a frame along the loop. For singularities, this angle is a multiple of $\pi/2$.

Definition 2.4 (Ports). A *port* is a vector in the tangential plane of a singularity which is tangent to an isoline of \mathcal{M} . The number of ports is the same as the valence of the singularity.

We say that a curve is *aligned* to a port if the angle between the corresponding curve and the port is less than 45° in parameter domain. We denote by \mathbf{p}_a^i a port based at a singularity s_i , its next counter-clockwise port by \mathbf{p}_{a+1}^i and its next clockwise port by \mathbf{p}_{a-1}^i .

Definition 2.5 (Closed separatrix, Open separatrix). A *closed separatrix* is a curve on \mathcal{M} which connects two singularities (not necessarily distinct). For the

case of surfaces with boundary, one of the singularities is allowed to be a boundary point. If one of the endpoints is not a singular point and the curve is an isoline, then we call it an *open separatrix*.

We denote by $\gamma_{ab}^{ij} : [0, 1] \rightarrow \mathbb{R}^3$, the closed separatrix which connects the singularities s_i and s_j , tangent to the ports \mathbf{p}_a^i and \mathbf{p}_b^j , and by γ_{a^i} the open separatrix starting at the singularity s_i and aligned to \mathbf{p}_a^i . Open separatrices do not exist on closed parameterizations since all isolines starting at a singularity end at another, not necessarily a different, singularity.

Definition 2.6 (Geodesics). A *geodesic* curve γ connecting to points p, q on \mathcal{M} is a locally shortest path on \mathcal{M} in the metric induced by the parameterization.

$$\gamma = \min_{\delta \in H(p,q)} \int_{t_p}^{t_q} g_f(\delta'(t), \delta'(t)) dt$$

where, $H(p, q)$ is the set of all homotopic curves connecting p and q considering singularities as small holes. g_f denotes the metric induced by the parameterization f .

In standard differential geometry, geodesics are generalization of straight lines on surfaces using the metric induced by the embedding space. In this definition, they are geodesics not on the surface but in parameter domain whose homotopy class are governed by singularities. A *straight line* on \mathcal{M} is a line in parameter domain. Straight lines are always geodesics but geodesics are not necessarily straight lines. Consider two points $p, q \in \mathcal{M}$ and a starting curve γ_0 connecting them. The locally shortest geodesic curve connecting p and q homotopic to γ_0 is a curve γ_{\min} such that the surface bounded by γ_0 and γ_{\min} does not contain a singularity and γ_{\min} is the shortest in the homotopy class of γ_0 . As depicted in Figure 2.2, geodesic curves on \mathcal{M} are not necessarily straight but they are if there is not any singularity close by.

Definition 2.7 (Triangles). A *triangle* is a geodesic right triangular patch whose base and height are isolines of \mathcal{M} .

We consider the base of the triangle as the longer of the right-angled sides. If both lengths are the same, then we choose either of them as a base. A triangle is *singularity-free* if the geodesic homotopic to the combined base-height curve, which defines the hypotenuse of the triangle, is straight on \mathcal{M} . It is not otherwise and curves exactly at "inner" singularities. In our context, we consider only triangles whose hypotenuses are closed separatrices.

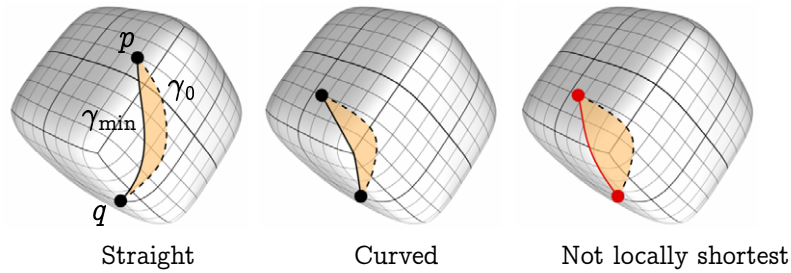


Figure 2.2: Geodesics on a parameterized surface. Locally shortest geodesics are the shortest within the homotopy class of curves not crossing a singularity.

2.2 BASE COMPLEXES AND QUAD LAYOUTS

Definition 2.8 (*Base Complex*). A *base complex* \mathcal{B} of a parameterized surface \mathcal{M} is a graph of closed separatrices where all surface elements bounded by intersecting separatrices are quadrilaterals.

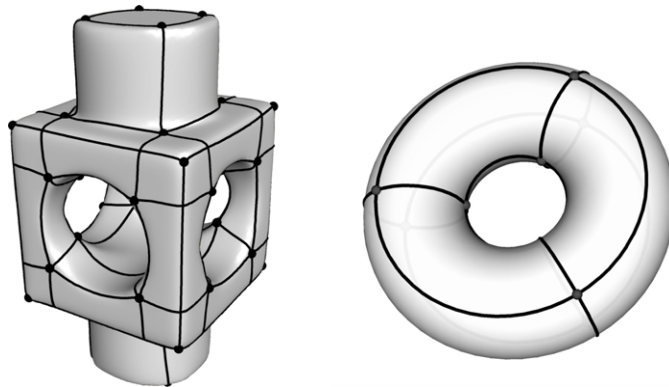


Figure 2.3: Examples of base complexes on high genus surfaces.

A closed parameterized surface admits a natural base complex obtained by the union of all parameter lines connecting singularities. Open parameterizations do not have a priori base complexes. If the parameterization does not have a singularity, then a regular vertex with the two intersecting isolines give a base complex of the surface. A base complex is not unique. It can be very coarse but it can as well be fine depending on the position of the singularities. In general, a base complex of a given surface (non-parameterized) does not need a parameterization to exist. Campen et al. [CBK12] for example construct a dual of the base complex using anisotropic

loops.

Base complexes are not only defined for quadrangular parameterization. Other type of parameterization such as triangular or hexagonal have as well base complexes. In this thesis, we only focus on quadrilateral base complexes for manifold surfaces.

Definition 2.9 (*Quad Layout*). A *quadrilateral patch layout* or *quad layout* is the union of all patches induced by a base complex of the surface where the patches are all quadrilaterals.

A quad layout induces a parameterization of the surface with coarser quad domains or *base domains*. The rectangular patches are the charts together with maps taking each chart to a 2D rectangular domain. The transition functions are composition of 90° rotations and translations which maps a chart boundary to another chart boundary. A good quad layout of a surface induces a low distortion parameterization

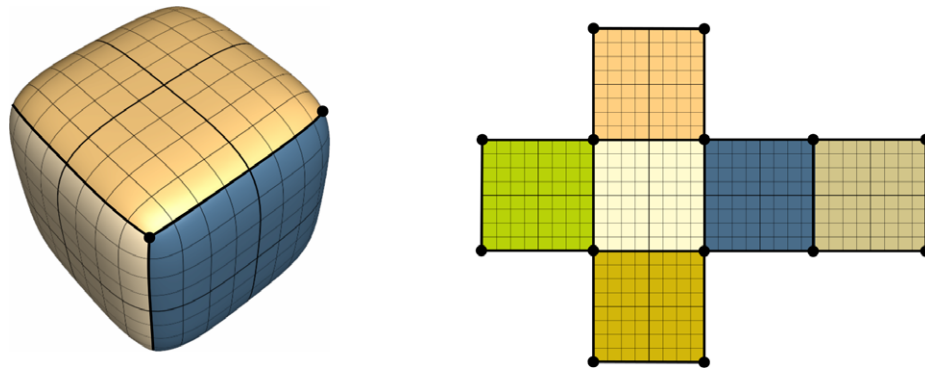


Figure 2.4: Quad layout induces a reparameterization of the surface whose charts are all rectangular patches.

of the surface. The patches are coarse and the separatrices are aligned to important features of the surface such as principal curvature directions. The individual charts have patch angles close to 90° . In practice, this ideal quad layout is hard to obtained such that a balance between coarseness and geometric feature alignment is als needed.

CHAPTER 3

MATCHING THEORY

A weighted graph \mathcal{G} is a set of vertices $V_{\mathcal{G}}$ connected by weighted edges forming a set $E_{\mathcal{G}}$ with edge weight w_e associated to an edge $e \in E_{\mathcal{G}}$. A *matching* \mathcal{Q} in a weighted graph $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}}, w_e)$, is a subgraph of \mathcal{G} such that no two edges share a vertex. A matching is *perfect* if all vertices of \mathcal{G} are in \mathcal{Q} . In other words, each vertex of \mathcal{G} is pairwise matched. A minimum (resp. maximum) weight perfect matching is a perfect matching with a minimal (resp. maximal) total edge weight. A typical example of a matching problem is the assignment problem. Suppose that we have a set of workers and a set of duties which has to be assigned to the workers. The number of duties is the same as the number of workers. For each duty, each worker has its own individual salary proposal. Some are high, some are reasonable and some are low. To maximize the production speed, each worker is assigned to exactly one duty but the total cost should also be minimized. The workers and the duties define a complete bipartite graph. The assignment problem is a minimum weight perfect matching problem.

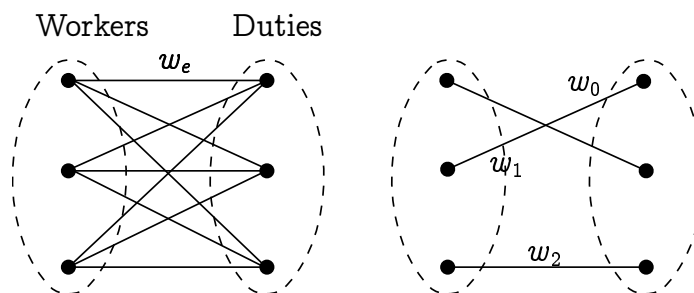


Figure 3.1: An example of a matching problem and a solution such that $w_0 + w_1 + w_2$ is minimum among all sums of edge weights.

The study of matching has been a long-time discussion in the field of graph theory. A complete overview can be found in [PL86]. Our algorithm uses the minimum

weight perfect matching problem with conflict pair constraints (MWPMPC) introduced in [DPSW11] and extended in [OZP13]. In our context, the conflict pair constraints are described in Section 4.6 as illegal crossing constraints which assure the quad topology of the resulting minimum weight perfect matching.

3.1 CLASSICAL MATCHING PROBLEM

As a combinatorial optimization problem, the minimum weight perfect matching problem (MWPM) can be formulated as a binary program that we denote by \mathbf{P}_0 ,

$$\begin{aligned} & \text{minimize} && \sum_{e \in E_G} w_e x_e \\ & \text{subject to} && \sum_{e \in \text{Adj}(v)} x_e = 1, && \text{for all } v \in V_G \\ & && x_e \in \{0, 1\}, && \text{for all } e \in E_G \end{aligned}$$

where $\text{Adj}(v)$ is the set of edges incident to a vertex v .

Binary programs are in general NP-hard but it has been proven by Edmonds [Edm65b, Edm65a] that the MWPM problem can be solve in polynomial time. He introduces the famous Blossom algorithm based on the fact that the odd cycles which are problematic in the solving of \mathbf{P}_0 can be identified using combinations of blossom shrinking or expanding followed by the search of augmenting paths. An example of the use of a MWPM is the direct conversion of triangle meshes to quad meshes as done in [RLS⁺12]. The approach consists of finding a perfect matching of the dual mesh edge graph which then induces a pairing of the triangles. A quad mesh is derived by removing the common edge of each paired triangle. To avoid distortion, a tangential smoothing of the mesh is applied.

3.2 DISJUNCTIVE CONSTRAINTS

A *disjunctive constraint* or *conflict pair constraint* is a pair of edges which is not allowed to appear simultaneously in the matching. Such constraints are sometimes indispensable in context such as budget constraints in the assignment problem. Mathematically, Darmann et al. [DPSW11] classify this type of constraints in two categories. A negative constraint expresses a conflict between two edges such that at most one of the edges is allowed to appear in the solution. A positive constraint

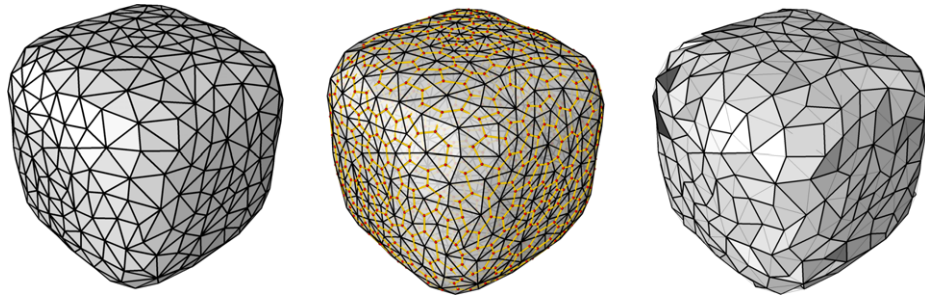


Figure 3.2: Perfect matching can be used to convert triangle meshes into non uniform quad meshes.

makes sure that at least one edge of the pair is in the solution. They can be translated into the following inequalities

$$x_{e_0} + x_{e_1} \leq 1 \quad (\text{negative}) \quad x_{e_0} + x_{e_1} \geq 1 \quad (\text{positive}) \quad (3.1)$$

if e_0 and e_1 are in conflict.

Surprisingly, the minimum weight perfect matching under disjunctive constraints (or conflict pair constraints MWPMPC) is strongly NP-hard as proven by Darmann et al..

3.3 QUAD LAYOUTS AND MATCHINGS

For a given parameterized surface \mathcal{M} , we would like to generate a quad layout of the surface. We formulate the problem as a minimum weight perfect matching problem with conflict pair constraints.

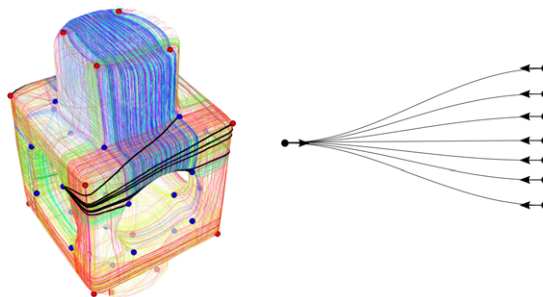


Figure 3.3: Finding a matching of a given port in the graph of separatrices.

In a quad layout, each singularity port is exactly matched to one other port. We build a graph \mathcal{G} whose vertices are the ports and whose edges are closed separatrices tangent to the ports. As stated in the previous Chapter, the number of ports at a singularity is the same as the valence of that singularity. Finding a quad layout of \mathcal{M} is then equivalent to finding a perfect matching of the ports.

Two edges are in conflict if the simultaneous appearance of both edges in the solution induces non-quad patches. Since one of the edge pairs do not need to appear in the solution, we are in the presence of a negative disjunctive constraints. As illustrated in Figure 3.4, there can be several solutions of the perfect matching problem on the graph of the ports. In this case, we allow the user to balance between coarseness and geometric feature alignment. This is one parameter added to the edge weights. In the next Chapter, we will see how to efficiently construct the graph of the ports in a consistent and robust manner.

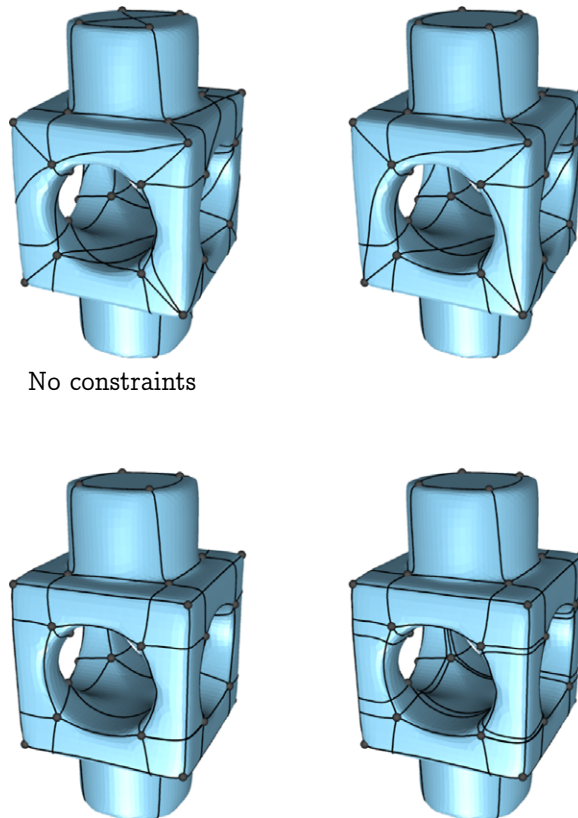


Figure 3.4: Example of perfect matchings on the Block model. There can be several perfect matching solutions. The number of patches versus geometric feature alignment can be controlled via a parameter.

CHAPTER 4

GRAPH CONSTRUCTION

In this Chapter, we construct a graph \mathcal{G} whose matching subgraphs are layout of \mathcal{M} . We then give a global formulation of the quad layout problem on closed as well as open parameterized surfaces whose solution is guaranteed to be optimal relative to a weight function.

In parameter space, isolines of \mathcal{M} are not necessarily connected. They are guided by the transition functions when they reach chart boundaries. To trace an isoline at a given point $p \in \mathcal{M}$ contained in a chart S_i in direction of a gradient vector \mathbf{v} , we start at $f_i(p)$ and follow the niveau line in direction of $Df_i(\mathbf{v})$ until the line hits the boundary of O_i at a point $f_i(\tilde{p})$. We then use the transition function φ_{ij} from O_i to O_j to find the image of $f_i(\tilde{p})$ in O_j together with the new direction $\varphi_{ij}(\mathbf{v})$. The same procedure is again applied to $\varphi_{ij} \circ f_i(\tilde{p})$ and $\varphi_{ij}(\mathbf{v})$ until the desired maximum tracing length is reached.

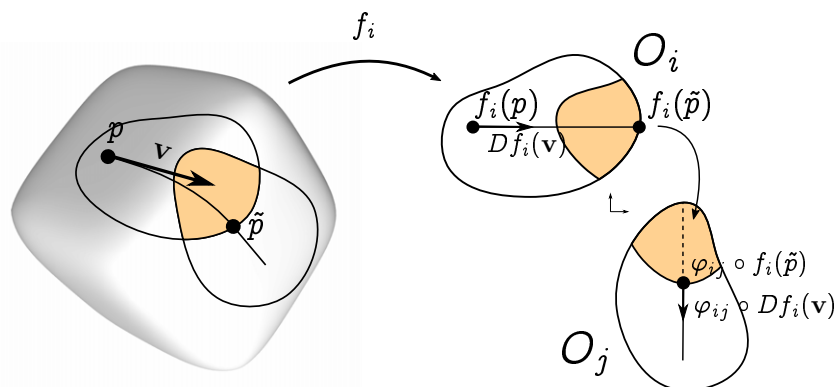


Figure 4.1: Tracing an isoline on a parameterized surface.

4.1 MOTORCYCLE GROWING

At each singularity, we grow simultaneously all isolines, at unit parametric length speed, in direction of the ports. These lines are open separatrices. If two of them intersect, we build a closed separatrix from the two end singularities using a ratio test. We add a new edge corresponding to that separatrix in \mathcal{G} . We define a function L_a^i which measures the geodesic distance of the singularity s_i to a point of the open separatrix γ_{a+}^i on \mathcal{M} . Mainly, given a point $p = \gamma_{a+}^i(t_p)$,

$$L_a^i(p) = \int_0^{t_p} g_f(\gamma_{a+}^i(t), \gamma_{a+}^i(t)) dt.$$

If $s_i \in S_k$ and $p \in S_l$, then $L_a^i(p)$ is the sum of the euclidean distances in each charts visited by constructing γ_{a+}^i as a straight line from $f_k(s_i)$ to $f_l(p)$ in parameter space as in Figure 4.1.

Now, consider two ports $\mathbf{p}_a^i, \mathbf{p}_b^j$ and the corresponding open separatrices $\gamma_{a+}^i, \gamma_{b+}^j$ intersecting at a point m of \mathcal{M} (see Figure 4.2). A new edge is added to \mathcal{G} according to the following conditions C1: if $\Delta s_i m s_j$ is a singularity-free triangle and

1. $L_b^j(m)/L_a^i(m) < 1$, then we add an edge connecting \mathbf{p}_a^i to \mathbf{p}_{b+1}^j in \mathcal{G} if s_j is on the right side of γ_{a+}^i , or \mathbf{p}_a^i to \mathbf{p}_{b-1}^j , if s_j is on the left side.
2. $L_b^j(m)/L_a^i(m) > 1$, then we add an edge connecting \mathbf{p}_b^j to \mathbf{p}_{a+1}^i in \mathcal{G} if s_i is on the right side of γ_{a+}^i , or \mathbf{p}_b^j to \mathbf{p}_{a-1}^i , if s_i is on the left side.
3. $L_b^j(m) = L_a^i(m)$, then we add one edge following Condition 1 and another edge following Condition 2, in \mathcal{G} . Both edges should not appear simultaneously in the final quad layout to avoid degeneration, we call these edges *diagonal edges* (see Section 4.6).

The new edge is a closed separatrix $\gamma_{a(b-1)}^{ij}$ defined geometrically as the parametric geodesic connecting s_i at the port \mathbf{p}_a^i and s_j at the port \mathbf{p}_{b-1}^j (not \mathbf{p}_b^j), homotopic to the curve $[\gamma_{a+}^i \circ \gamma_{b+}^j]$.

4.2 SINGULARITY-FREE TRIANGLES

On a parameterized surface \mathcal{M} , singularities behave like small holes. Geodesics can get stuck on these special points. In our construction, edges are geodesic

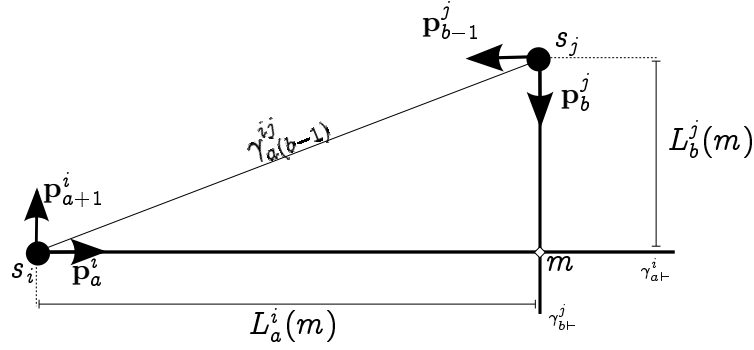


Figure 4.2: Creation of an edge by evaluating the ratio of two intersecting separatrices at a point m .

curves homotopic to the combined base and height curves of triangles. Allowing non straight edges will produce degenerate quadrilateral patches in the final layout (patches might have side length equal to zero). We require all triangles to be singularity-free during the isoline growing step.

For each port, we associate two propagation windows: left window and right window which initially have an angle of 45° with the port. We will describe our construction for the case of left window, the other case can be handled similarly. Consider an open separatrix $\gamma_{a_0+}^0$, starting at a singularity s_0 in direction of $\mathbf{p}_{a_0}^0$, as illustrated in Figure 4.3. Suppose that it intersects n other open separatrices $\gamma_{a_1+}^1, \gamma_{a_2+}^2, \dots, \gamma_{a_n+}^n$ at the points m_1, m_2, \dots, m_n . Additionally, suppose that all the corresponding base

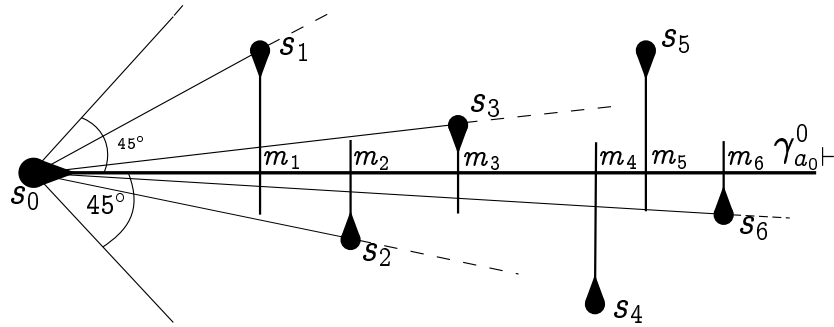


Figure 4.3: Singularities satisfying one of the ratio test along the open separatrix $\gamma_{a_0+}^0$. In this figure, only s_1, s_2, s_3 and s_6 agree with the singularity-free triangle constraint

singularities satisfy one of the ratio tests in condition C1 but not necessarily the singularity-free triangle condition.

Proposition 4.1. A triangle is singularity-free if the angle of its hypotenuse with

$\gamma_{a_0+}^0$ is less than the latest updated window size.

The angle at the hypotenuse defines the window size. If this angle is greater than the latest window size along $\gamma_{a_0+}^0$, then there is a singularity with smaller slope and hence lies inside the current triangle. Initially, the window size ψ is 45° . It is then updated according to the first singularity satisfying condition C1. $\Delta_{s_0 m_1 s_1}$ is always singularity-free since there is no other singularity preceding it within the starting window size. Depending on the position of s_1 (left or right of $\gamma_{a_0+}^0$), the corresponding new window size becomes $\psi = \arctan(L_{a_1}^1(m_1)/L_{a_0}^0(m_1))$. Along $\gamma_{a_0+}^0$, for $k \geq 2$, we collect singularities s_k satisfying

$$\arctan(L_{a_k}^k(m_k)/L_{a_0}^0(m_k)) < \psi,$$

and assigning the value of ψ to be the newly computed angle. This window reduction approach makes sure that no triangle contains a singularity and hence produces always straight geodesics on \mathcal{M} . In Figure 4.4 is an example of candidate singularities for a given port on the Icosahedron model.

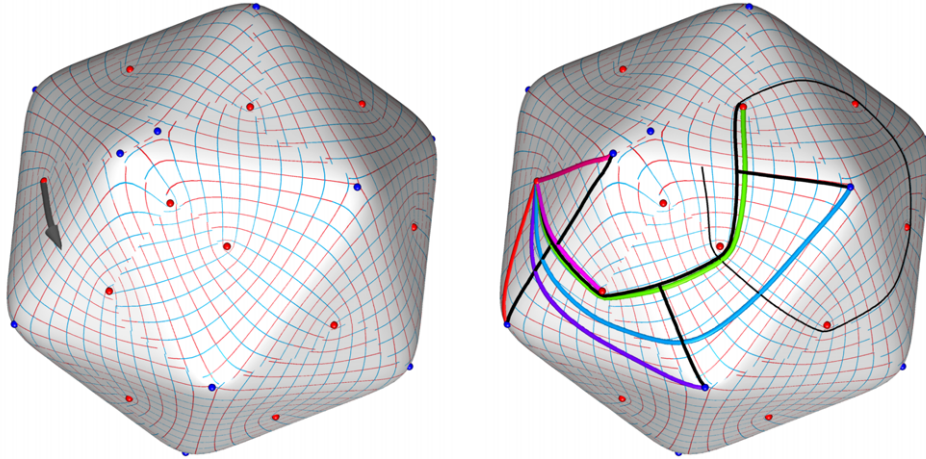


Figure 4.4: Example of candidate singularities satisfying the singularity-free triangle constraint on the Icosahedron model.

4.3 EDGE WEIGHTS

Recall that an edge $e \in \mathcal{G}$ connects two singularities s_i and s_j tangent to their respective ports. It is the hypotenuse of the triangle having $s_i m$ as base and $m s_j$

as height where m is the intersection point of the two separatrices $\gamma_{a^+}^i$ and $\gamma_{b^+}^j$. We define the weight of e as the linear combination of the base length and height length of the corresponding triangle. Mainly,

$$w_e = L_a^i(m) + \alpha L_b^j(m). \quad (4.1)$$

We can interpret α as a balance parameter which penalizes the deviation of a separatrix from the isolines of the parameterization. Taking $\alpha = 0$ ignores alignment to geometric features but prioritize short separatrices which in general gives the coarsest layout of the geometry. Choosing $\alpha = \infty$ increases the geometric feature alignment at the cost of increased number of patches. The same weighting is used in [TPP⁺11] as a separatrix's energy.

4.4 BOUNDARY

If an isoline ends at a boundary, we introduce a *virtual port* tangent to that separatrix pointing inside the surface and based at the boundary vertex. We then add a new vertex corresponding to the virtual port in \mathcal{G} together with an edge representing the separatrix, as shown in Figure 4.5. We do not collect singularity candidates from virtual ports, we only use them to have an uniform framework for surfaces with and without boundary. For singularities lying on boundary curves, we collect only candidates for the ports pointing inside the surface. The ports tangent to the boundary only grow but can also be used to collect candidates for other isolines. The concept of virtual port together with the previous consideration always maintain the quad topology of the boundary induced from \mathcal{M} . However, it requires that the boundary curves are isolines of \mathcal{M} . If not, it only assures pure quad for interior patches and arbitrary polygons for the boundary patches.

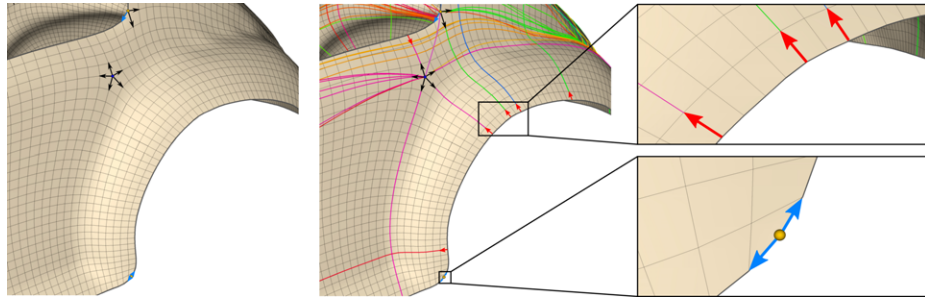


Figure 4.5: Tracing isolines for the case of surfaces with boundary.

4.5 MATCHING

After constructing \mathcal{G} , we would like to find a subgraph \mathcal{Q} of \mathcal{G} which forms a quad layout of our surface \mathcal{M} . The graph \mathcal{G} is huge and finding a proper quad layout subgraph needs a careful understanding of the edge crossings of \mathcal{G} on \mathcal{M} . By choosing exactly one edge per port (one or zero edges per virtual port), we obtain a perfect matching subgraph forming an arbitrary layout of \mathcal{M} which is not necessarily a pure quad layout. In Figure 4.6 (middle) is a perfect matching without constraints where the crossing red edges are for example in conflict. This is resolved in Figure 4.6 (right) by allowing at most one edge of each conflict pair to appear in the perfect matching.

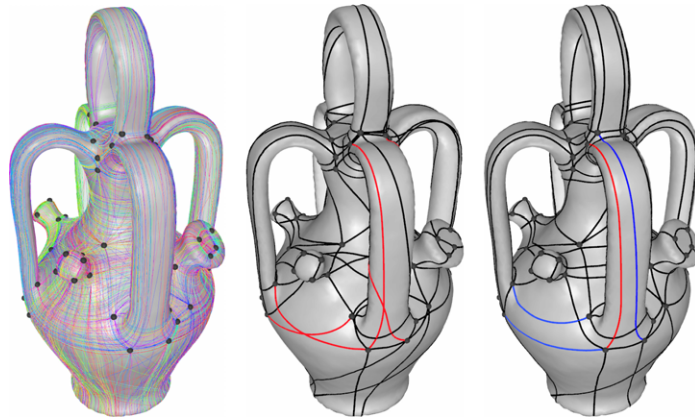


Figure 4.6: Examples of a perfect matching subgraph of \mathcal{G} on the Botijo model where the middle one is unconstrained and the right one constrained for $\alpha = 0$.

4.6 ILLEGAL CROSSINGS

The non-quad patches appearing in the unconstrained matching are caused by crossing edges which are locally aligned to the same parameter lines. We detect these edges and define a conflict set I in order to obtain a quad-only matching of the final layout. Two crossing edges in \mathcal{G} are aligned locally to the same parameter line on \mathcal{M} if the two triangles associated to the edges intersect and have parallel bases. We then say that the crossing is *illegal*, and it is *legal* otherwise. We denote by I the set of all illegal crossings of \mathcal{G} ,

$$I = \{(e_0, e_1) \in E_{\mathcal{G}}^2, e_0 \text{ crosses } e_1 \text{ illegally}\}. \quad (4.2)$$

We construct I as follows. Consider two open separatrices γ_{a+}^i and γ_{b+}^j intersecting at a point m_k (see inset diagram). We define an orthogonal frame $\{x, y\} = \{s_i m_k, m_k s_j\}$ at m_k in parametric space. Now, consider all edges whose triangles have a side on γ_{a+}^i or γ_{b+}^j . In this local coordinate system, we can do 2D triangle intersection checks without ambiguity.

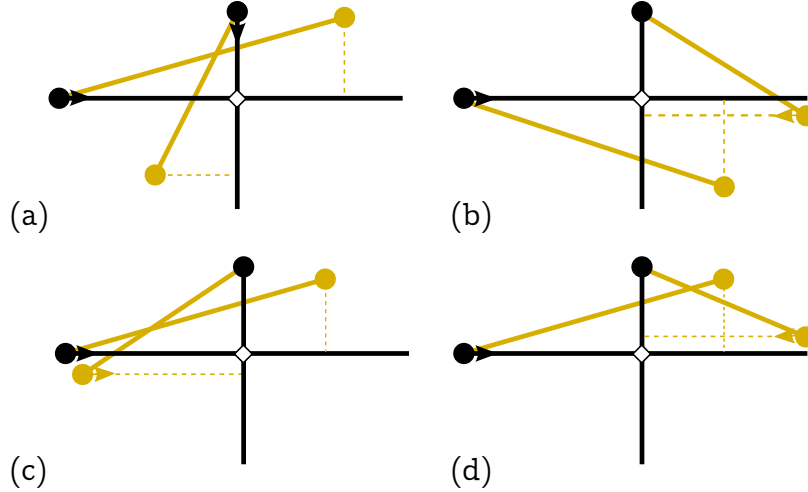
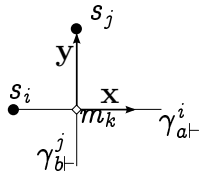


Figure 4.7: Example of edge crossings (yellow) represented in a common local coordinate system. In (a) and (b) are examples of legal crossings where in (c) and (d) are illegal crossings.

We say that an edge is x -aligned (resp. y -aligned) if the base of its associated triangle is parallel to x (resp. y). Two edges which cross and have different alignment have a legal crossing. On the other hand, if they cross and have the same alignment, they have an illegal crossing. It is important that they cross locally because it can happen that two edges with the same alignment do not cross although their triangles intersect, as illustrated in Figure 4.7 (b). We also add the pairs of edges generated by the case $L_a^i(m) = L_b^j(m)$ in the graph construction to I , to avoid degenerate patches in the final quad layout.

Remark 4.1 (Special cases). There are special cases where the local coordinate system cannot determine a priori the crossings of two edges. Those cases appear when one or more separatrices self-intersect. In Figure 4.8 is an example of such case. The two open separatrices have two intersection points m_0 and m_1 . The crossings are legal according to their local frames. However, we can see from m_2 which is another local frame that the crossing is illegal. In this case, illegality is

always stronger than legality. Crossings might be legal at some frames but illegal at other frames.

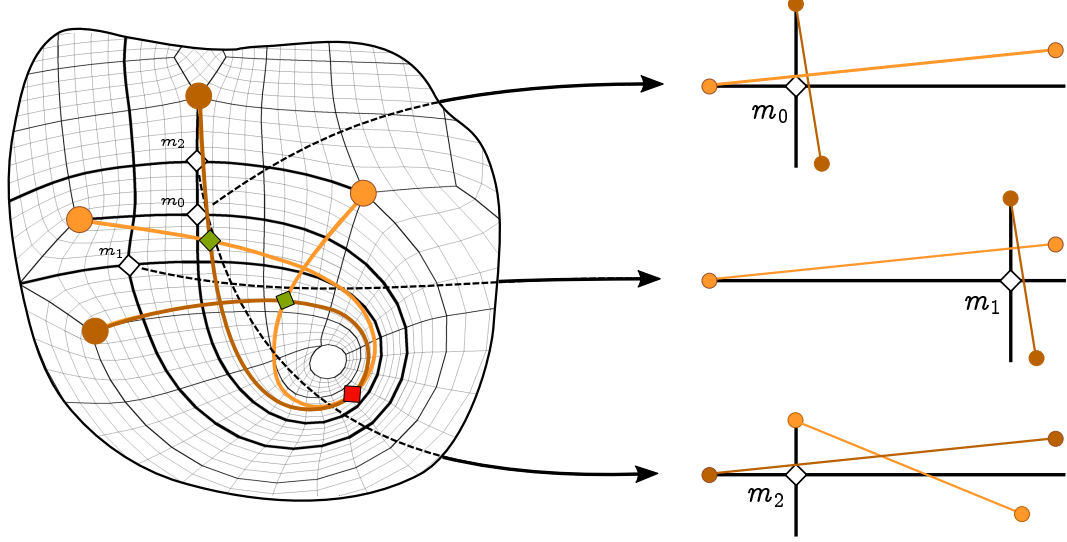


Figure 4.8: Special cases where the property of a crossing cannot be decided at one local coordinate system. In this configuration, local frames at m_0 and m_1 make the crossing of the two edges legal where it is illegal viewed from m_2 .

In the previous example, we can only decide about the property of the crossings at m_2 . But if m_2 does not exist then we need to consider the local frame defined at the point where the separatrix self-intersect. An example of this situation is given in Figure 4.9. Let us denote this separatrix by γ_{a+}^i . In this Figure, m_2 is considered to be the point of the self intersection of γ_{a+}^i . At this point, the edges whose triangles have their bases on γ_{a+}^i and contain m_2 have two alignment x and y . They have then two representations. Hence, to include this special case in our framework, we also allow $\gamma_{a+}^i = \gamma_{b+}^i$ in the local frame construction.

Remark 4.2 ($L_a^i(\mathbf{m}) = L_b^j(\mathbf{m})$). In the growing procedure, it can happen that the ratio of length of two intersecting open separatrices is equal to one. In this case we add two edges in \mathcal{G} , assign them a different alignment and add the pairs to the set of illegal crossing edges.

To understand the main problem, consider two open separatrices γ_{a+}^i and γ_{b+}^j which intersect each other at a point m and $L_a^i(m) = L_b^j(m)$. Without loss of generality, we suppose that s_j is on the left of γ_{a+}^i . The two edges $e_0 = \mathbf{p}_a^i \mathbf{p}_{b-1}^j$ and $e_1 = \mathbf{p}_b^j \mathbf{p}_{a+1}^i$ bound a surface of zero area on \mathcal{M} (as parametric geodesics). If both edges appear

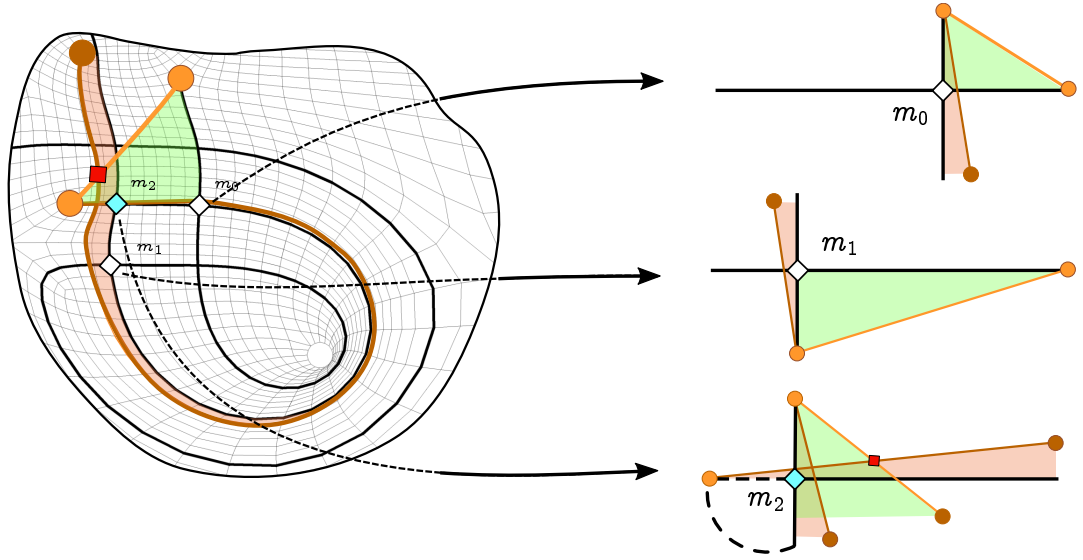


Figure 4.9: Special cases where the property of a crossing can only be decided at the self intersection point of a separatrix.

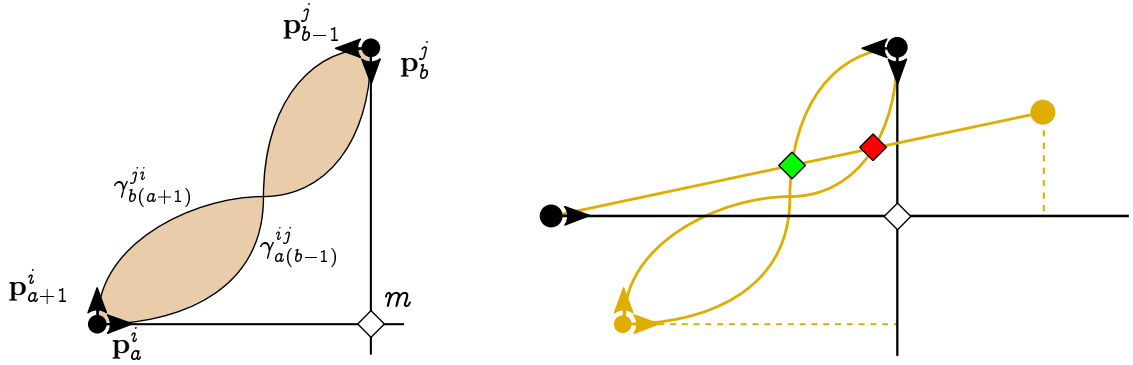


Figure 4.10: Case where the ratio of lengths of two separatrices is equal to one. Two edges are added in \mathcal{G} , one of them is x -aligned and the other one is y -aligned.

in the perfect matching graph, then we will get a degenerate quad layout. Hence, we extend I to $I \cup Z$, where Z is the set of pairs of edges spanning a zero surface area on \mathcal{M} . If an edge e_2 intersects one of e_0 and e_1 , then $(e_2, e_0) \in I$, i.e, they have an illegal crossing, and $(e_2, e_1) \notin I$ or vice versa. This case happens often for closed parameterized surface but rarely, or even non-existent, for open ones.

Algorithm We summarize the construction of I in Algorithm 1. The set K contains all intersection points of the open separatrices satisfying condition C1. The set I can be empty if \mathcal{G} is already a quad layout matching of \mathcal{M} .

Algorithm 1 Construction of the set of illegal crossings I

Require: Set of open separatrices intersection points K , set of pairs of edges Z spanning a zero surface area

```

1: Put  $I = \emptyset$ 
2: for  $m_k \in K$  do
3:   Take the two separatrices  $\gamma_{a^+}^i$  and  $\gamma_{b^+}^j$  intersecting at  $m_k$ . /* Not necessarily
   distinct */
4:    $\{x, y\} = \{s_i m_k, m_k s_j\}$ 
5:    $\{e_l\}_{l=0, \dots, n}$  the set of edges whose triangles have a side on  $\gamma_{a^+}^i$  or  $\gamma_{b^+}^j$  .
6:   for  $l \leftarrow 0$  to  $n$  do
7:     for  $r \leftarrow l + 1$  to  $n$  do
8:       if  $e_l$  and  $e_r$  are both x- or y-aligned then
9:         if  $e_l$  and  $e_r$  intersect then
10:           $I = I \cup \{(e_l, e_r)\}$ 
11:        end if
12:      end if
13:    end for
14:  end for
15: end for
16:  $I = I \cup Z$ 

```

CHAPTER 5

GLOBAL OPTIMIZATION

So far, we have constructed a graph of the ports with a set of edge pairs which are disjunctive constraints. Finding a proper subgraph of \mathcal{G} which forms a quad layout of \mathcal{M} is equivalent to solving a perfect matching problem with disjunctive constraints. We show in this Chapter a proof of this statement and a simple geometrical approximation of the solution which does not involve advanced integer program solvers.

5.1 BINARY PROGRAM FORMULATION

We derive a quad layout of \mathcal{M} as a subgraph matching of \mathcal{G} . Recall that the vertices of \mathcal{G} are the ports and the edges are closed separatrices. Assigning exactly one edge per ports and making sure that no illegal pairs of edges are picked will always produce a quad layout. Finding a quad layout of \mathcal{M} is then equivalent to solving a MWPMPC in \mathcal{G} . This is equivalent to solving the following binary program \mathbf{P}_1 .

$$\text{minimize} \quad \sum_{e \in E_{\mathcal{G}}} w_e x_e \quad (5.1)$$

$$\text{s.t.} \quad \sum_{e \in \text{Adj}(v)} x_e = 1, \quad \text{for all not virtual } v \in V_{\mathcal{G}} \quad (5.2)$$

$$x_{e_1} + x_{e_2} \leq 1, \quad \text{for all } (e_1, e_2) \in I \quad (5.3)$$

$$x_e \in \{0, 1\}, \quad \text{for all } e \in E_{\mathcal{G}} \quad (5.4)$$

where $\text{Adj}(v)$ is the set of edges sharing a vertex $v \in V_{\mathcal{G}}$. The objective functional minimizing the total edge weights is given by (5.1). Constraint (5.2) assures that per vertex of G , exactly one incident edge is taken. In \mathcal{M} , this is equivalent to

assigning exactly one closed separatrix to each port. Condition (5.3) is a translation of the illegal crossing constraints into inequality constraints. It assures the quad topology of the final layout. Before going into the main theorems for quad layouts on manifold meshes, we give the following Lemma on the quad topology of a feasible solution of \mathbf{P}_1 .

Lemma 5.1. If all edges in a minimum weight perfect matching Q of \mathcal{G} are conflict free, then Q is a quad layout of \mathcal{M} .

Proof. Suppose that there exists a patch P of Q which is a n -gon, $n \neq 4$ bounded by legal edges. Consider the parameterized surface element S which contains P . In S , we can characterize locally to which parameter lines the edges of P are aligned. If n is odd, then a cyclic uv -assignment of the edges will always produce a uu or vv intersections which contradict legality. If n is even, then the tangential property of the lines induces again a uu or a vv intersection. In fact, it is only possible for $n = 4$ or if P contains a singularity which is not the case here. \square

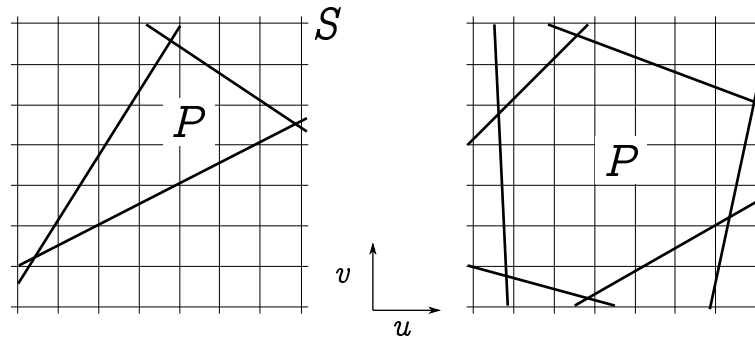


Figure 5.1: Examples of non-quad patches which can not be locally bounded by legal edges.

This Lemma gives a guarantee that constraints (5.3) assure the quad topology of the resulting perfect matching. We now formulate a Theorem for closed parameterized surface.

Theorem 5.1 (Quad Layout for Closed Parameterization). *Given a closed parameterized surface \mathcal{M} and the graph of the ports $\mathcal{G} = (V, E, w_e)$ with the set of illegal crossings I . An optimal quad layout of \mathcal{M} relative to the weight w_e is a solution of \mathbf{P}_1 .*

Proof. The existence of a feasible solution is guaranteed by the closeness of the parameterization obtained by its base complex. The quad topology of the resulting perfect matching layout is assured by Lemma 5.1. \square

From this theorem, we can measure without ambiguity the quality of the base complex of the parameterization in terms of coarseness and geometric feature alignment. More, we can improve the base complex of the surface by solving the binary program \mathbf{P}_1 . A practical use of this Theorem is for example the global structure optimization for quadrilateral meshes presented in Chapter 7.

The existence of a feasible solution of \mathbf{P}_1 on open parameterized surfaces is not a priori guaranteed because they do not have a predefined base complex. Finding an initial base complex of open parameterized surfaces is exactly the quad layout problem. Assuming that \mathcal{G} has a perfect matching (not necessarily constrained), we can always design a base complex using what we call a *bridge* at the intersection points of two illegal crossing edges. The main idea is to interpret a geometric meaning of Lemma 5.1. If we succeed to repair geometrically all conflicting edges in a perfect matching of \mathcal{G} , then the repaired subgraph matching is added to \mathcal{G} to assure the existence of a solution. Notice, that the repaired graph is not necessarily optimal and may not even satisfy the singularity-free condition. It is only a topological quad layout which in case of degeneration need a post processing relaxation.

Consider a perfect matching Q in \mathcal{G} and two edges e_0 and e_1 which are in conflict. Without loss of generality, suppose that the two edges are directed such that their starting vertices are on the “left” of their intersection point m and their end vertices are on the “right” (see Figure 5.2) on \mathcal{M} . We bridge the two edges at m such that $\text{start}(e_0)$ is connected to $\text{end}(e_1)$ and $\text{start}(e_1)$ is connected to $\text{end}(e_0)$. The two edges become then conflict free. We apply iteratively this local repair until no two edges are in conflict. Even though, the resulting quad layout is not always visually pleasing and might even be far from a solution of \mathbf{P}_1 , it is a base complex of the open parameterized surface. An example of this procedure is given in Figure 5.2.

Theorem 5.2 (Quad Layout for Open Parameterization). *Given an open parameterized surface \mathcal{M} and the graph of the ports $\mathcal{G} = (V, E, w_e)$ with the set of illegal crossings I . If \mathcal{G} has a perfect matching, then an optimal quad layout of \mathcal{M} relative to the weight w_e is a solution of \mathbf{P}_1 .*

A solution of \mathbf{P}_1 is geometrically constructed from a non constrained perfect matching of \mathcal{G} with the local repairs defined in the previous Chapter. We add it to the graph, recompute the illegal crossing constraints and use Theorem 5.1. If \mathcal{G} does

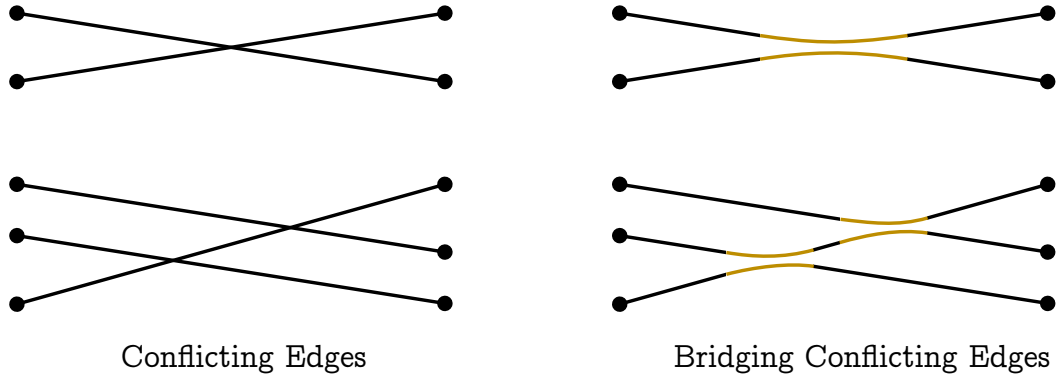


Figure 5.2: Resolving conflicting edges geometrically by introducing a *bridge* at their intersection points.

not have a perfect matching, then the graph construction did not collect enough candidates for one given port. Showing that any construction of \mathcal{G} admits a perfect matching subgraph of the ports is an open problem. Since checking if a graph has a perfect matching can be done in polynomial time (using Edmonds' Blossom algorithm), we can always increase the shooting maximum length and let the graph grow accordingly until a perfect matching is found. In practice, we never had to use this approach since our default parameters always give a perfect matching.

5.2 COMPLEXITY

Understanding the complexity of \mathbf{P}_1 can be achieved by formulating a maximum stable set problem. We construct a graph $\mathcal{G}' = (V', E', w_v)$ derived from $\mathcal{G} = (V, E, w_e)$ including the constraints I . Solving \mathbf{P}_1 in \mathcal{G} is then equivalent to solving a new integer program \mathbf{P}'_1 in \mathcal{G}' . The vertices of \mathcal{G}' are the edges of \mathcal{G} . Two vertices of \mathcal{G}' are connected if the corresponding edges in \mathcal{G} share the same vertex. At this point \mathcal{G}' is called the *line graph* of \mathcal{G} . We add the constraints as follows. If two edges of \mathcal{G} cross illegally, then we connect the two vertices corresponding to these edges in \mathcal{G}' . The edge weight w_e becomes a vertex weight $w_v = 1/w_e$ in \mathcal{G}' . In summary, \mathcal{G}' is a graph with $|E|$ vertices and $|E|(|E|-1)/2 + |I|$ edges. In Figure 5.3 is an example of a graph \mathcal{G}' (dotted) derived from \mathcal{G} . The two red edges $p_i q_j$ and $p_j q_i$ cross illegally. An edge is then added between their respective vertices which connects the two complete graphs in \mathcal{G}' . The binary program \mathbf{P}_1 is equivalent to

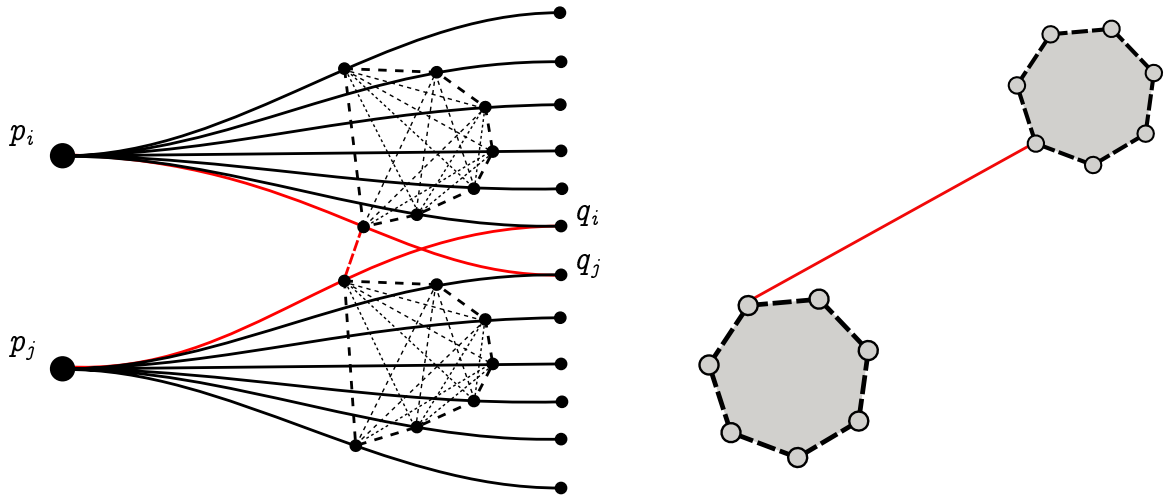


Figure 5.3: Example of a construction of the graph \mathcal{G}' from \mathcal{G} . The red edges are crossing illegally (left). Illegal crossing edges in \mathcal{G} are edges in \mathcal{G}' connecting complete graphs (right).

the following binary program \mathbf{P}'_1 ,

$$\text{maximize } \sum_{v \in V'} w_v x_v \quad (5.5)$$

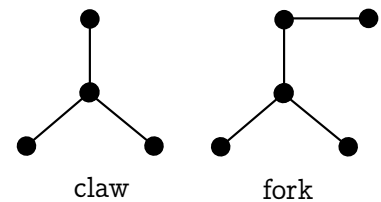
$$\text{subject to } x_{v_0} + x_{v_1} \leq 1, \quad \text{for all } v_0 v_1 \in E' \quad (5.6)$$

$$x_v \in \{0, 1\}, \quad \text{for all } v \in V' \quad (5.7)$$

This is a maximum weight stable set problem which assigns one out of two numbers (0 or 1) to all vertices such that no two adjacent vertices have the same numbering.

The structure of \mathcal{G}' depends heavily on I . The complete sub-graphs can be interpreted as a set which are connected by disjunctive “vertices”. There are only few class of graphs on which the maximum independent set is known to be solvable in polynomial time. These are *claw-free*, *fork-free* and *line graphs*. In our case, \mathcal{G}' would be the line graph of \mathcal{G} without constraints.

A *claw* is a graph with four vertices composed by a central vertex connecting to the other three which in their turn are not connected to each other. It is also called a 3-star graph. A graph is *claw-free* if it does not have an induced claw subgraph. A *fork* is an extension of a claw with one more vertex attached to the surrounding three vertices of a claw. A graph is *fork-free*, if it does not have



an induced fork subgraph. A claw-free graph will be necessarily fork-free but a fork-free is not necessarily claw-free which makes the class of fork-free graph larger.

Lemma 5.2. The graph \mathcal{G}' is not in general fork-free.

Complete graphs are always fork-free. The same as the connection of two complete graphs by an edge. \mathcal{G}' would be fork-free if any edge e_0 of \mathcal{G} satisfies one of the following conditions: e_0 crosses exactly one edge e_1 illegally and not any other edge or e_0 crosses several edges sharing the same vertex in \mathcal{G} (see Figure 5.4). A fork exactly appears in \mathcal{G}' when an edge is crossed illegally more than once in \mathcal{G} . We could get such a fork free graph by restricting the number of candidates per port in \mathcal{G} according to these conditions but the space of solutions become too small which would be unsuitable to our application and even a non-constrained perfect matching may not exist. The graph \mathcal{G}' constructed in Figure 5.3 is for example fork-free as well as claw-free.

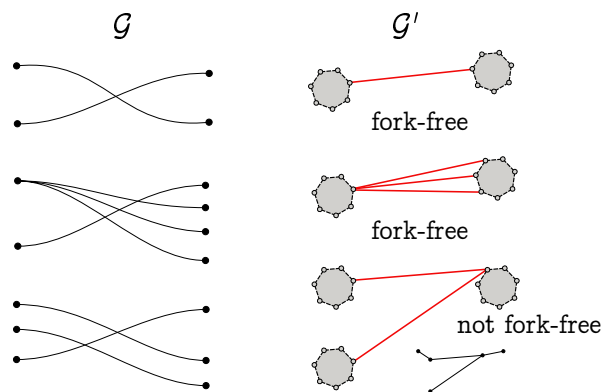


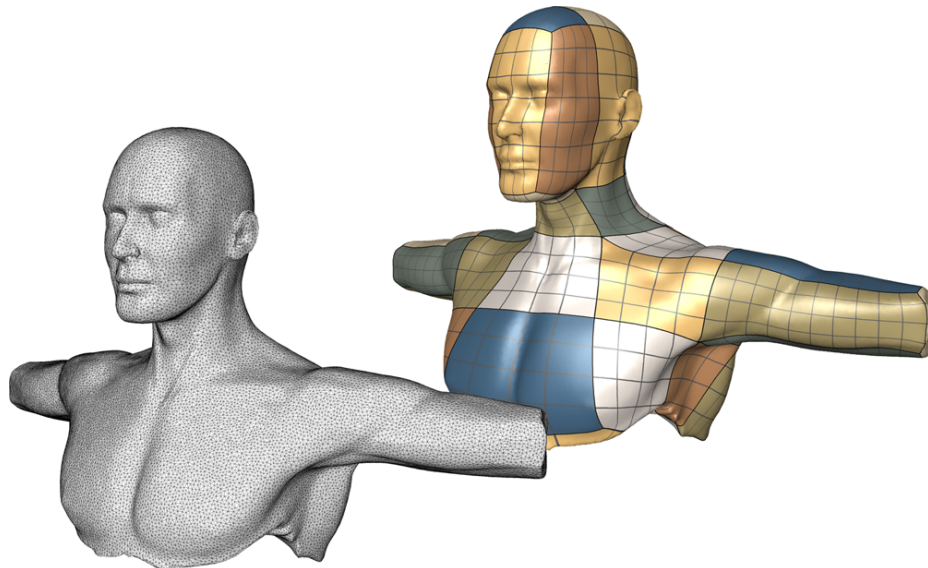
Figure 5.4: Appearance of a fork-free subgraph in \mathcal{G}' caused by a multiple illegal crossing edges in \mathcal{G} .

Theorem 5.3. *The integer program P_1 can be solved in polynomial time if and only if the maximum weight independent set problem and hence P'_1 can be solved in polynomial time.*

The fact that \mathcal{G}' does not lie in some class of known graphs where a polynomial time algorithm exists makes the problem a general maximum weight independent set problem on \mathcal{G}' which is known to be NP-hard.

CHAPTER 6

APPLICATION I: QUAD LAYOUT ON TRIANGLE MESHES



This Chapter shows the effectiveness of the perfect matching formulation to generate coarse quad layouts on simplicial surfaces. In these class of surfaces, the charts are represented by triangles. A parameterization is derived by generating first a per triangle guiding frame field and integrating the field following the approach of [KNP07] but avoiding the greedy integer rounding. The resulting parameterization is an open parameterization whose gradients align best to the input guiding field in a least square sens.

Typically, methods such as QuadCover [KNP07] and MIQ [BZK09] produce high quality quadrilateral meshes within small amount of time. But, these algorithms are not well suited for quad layout generation. Singularities might be mapped to the

same grid points in the parameterization phase. Our approach avoids the integer rounding by matching directly the ports induced by the input frame field.

6.1 SIMPLICIAL SURFACES

Definition 6.1. A *simplicial surface* is a piecewise linear surface obtained geometrically by glueing *simplices* or *triangles* at their edges and is homeomorphic to a topological manifold.

On a simplicial surface, the charts are represented by triangles forming a set $T_{\mathcal{M}}$. They are connected by edges forming a set $E_{\mathcal{M}}$ and edges are bounded by vertices forming a set $V_{\mathcal{M}}$.

Definition 6.2. A *discrete global parameterization* \mathcal{M} is a collection of triangle charts $\{(T, f_i)\}_{T \in T_{\mathcal{M}}}$ such that $\mathcal{M} = \cup_{T \in T_{\mathcal{M}}} T$ and if two triangles T_i and T_j share an edge e_{ij} , then there exists a transformation φ_{ij} such that

$$f_j(e_{ij}) = \varphi_{ij} \circ f_i(e_{ij}).$$

φ_{ij} is again a composition of rotation and translation as in the smooth case. A gradient vector in a triangle is defined as a piecewise constant vector lying in that triangle. Along the common edge of two triangles, the gradient vector is not defined since they are different in both triangles. Hence, simplicial surfaces are only piecewise smooth surfaces.

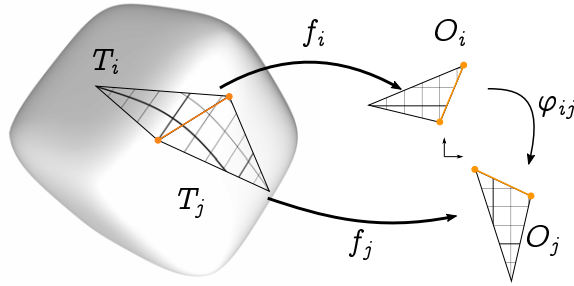


Figure 6.1: Triangle charts transition map on a simplicial surface.

6.2 FUNCTION SPACES ON SIMPLICIAL MANIFOLDS

On a simplicial surface \mathcal{M} , we consider two types of piecewise linear (PL) function spaces, the *conforming Lagrange space* $S(\mathcal{M})$ and the *non-conforming space* $S^*(\mathcal{M})$. Both spaces are classic in finite element literature, but here we will see how both function spaces team up to mimic the concept of primality and duality of grids in the framework of function spaces on the same underlying geometry, thus allowing to use the simplicial surface \mathcal{M} as single base geometry. For example, solutions of the discrete Cauchy-Riemann equations will consist of a pair of a discrete conforming and non-conforming harmonic map in S and in S^* resp. vice versa, all defined on \mathcal{M} .

Definition 6.3. The *piecewise linear conforming* $S(\mathcal{M})$ and *non-conforming* $S^*(\mathcal{M})$ function spaces on a 2-dimensional simplicial surface $\mathcal{M} \subset \mathbb{R}^2$ are given by:

$$\begin{aligned} S &:= \left\{ f : \mathcal{M} \rightarrow \mathbb{R} \mid f|_T \text{ is linear on each triangle } T, \text{ and } f \in C^0(\mathcal{M}) \right\} \\ S^* &:= \left\{ f^* : \mathcal{M} \rightarrow \mathbb{R} \mid f^*|_T \text{ is linear, and continuous at edge midpoints} \right\} \end{aligned}$$

On first sight, the missing global continuity of functions $f^* \in S^*$ sounds like a drawback but the space of non-conforming functions will turn out as a good match to S . Both function types have piecewise gradients by ordinary differentiation.

Definition 6.4. The *gradient field* ∇f of a function $f \in S$ or S^* is a constant tangent vector in each triangle. The *co-gradient field* $\delta f := J\nabla f$ is obtained by rotation J of the gradient ∇f , i.e. by $\frac{\pi}{2}$ in each triangle.

6.3 FIELD CURL ON SIMPLICIAL SURFACES

Piecewise constant vector fields were introduced to geometry processing in [PP03] as a natural discretization of tangential vector fields on simplicial geometries. Among the useful properties of PC vector fields, say compared to Whitney type differential forms, are the formulation of the Hodge star operation in function spaces instead of introducing a pair of primary and dual meshes. After a short overview of PC vector fields and their integrability conditions we show how such theory can be extended to sets of vector fields called *frame fields* in the goal of deriving a parameterization on which we can reliably construct the graph \mathcal{G} .

Definition 6.5. The space of *piecewise constant tangential vector fields* $\Lambda^1(\mathcal{M})$ on a 2-dimensional simplicial surface \mathcal{M} is given by:

$$\Lambda^1(\mathcal{M}) := \left\{ \mathbf{v} : \mathcal{M} \rightarrow \mathrm{T}\mathcal{M} \mid \mathbf{v}|_{\text{triangle } T} \text{ is a constant tangent vector in } T \right\}$$

The gradient and co-gradients fields of functions in S or S^* introduced above are examples of piecewise constant (PC) tangential vector fields.

Definition 6.6. On a simplicial surface \mathcal{M} let $\mathbf{v} \in \Lambda^1(\mathcal{M})$, p a vertex and m an edge midpoint. Then the (total) *discrete curl* is given by

$$\begin{aligned} \mathrm{curl} \mathbf{v}(p) &:= \frac{1}{2} \sum_{i=1}^k \langle \mathbf{v}|_{T_i}, \mathbf{e}_i \rangle \\ \mathrm{curl}^* \mathbf{v}(m) &:= -\langle \mathbf{v}|_{T_1}, \mathbf{e} \rangle + \langle \mathbf{v}|_{T_2}, \mathbf{e} \rangle \end{aligned}$$

where \mathbf{e}_i is an edge of the oriented boundary of star p resp. \mathbf{e} the common edge of the triangles T_1 and T_2 .

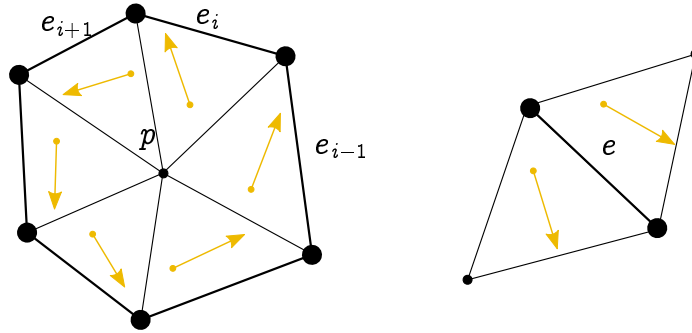


Figure 6.2: Computation of curl and curl* on a simplicial surface.

Theorem 6.1 (Local integrability conditions for vector fields). *Let \mathcal{M} be a simply connected simplicial surface. Then a PC vector field $\mathbf{v} \in \Lambda^1(\mathcal{M})$ can be characterized as gradient field in term of the discrete curl operator:*

1. v is a gradient field of a function in $S \iff$

$$\mathrm{curl}^* \mathbf{v}(m) = 0 \text{ at all edge midpoints } m.$$

2. v is a gradient field of a function in $S^* \iff$

$$\mathrm{curl} \mathbf{v}(p) = 0 \text{ at all vertices } p.$$

Definition 6.7 (Frame Fields). An orthogonal *frame field* or *cross field* \mathcal{F} is a set of pairs of constant tangent vectors $\{(\mathbf{u}_T, \mathbf{v}_T)\}_{T \in \mathcal{T}_M}$ based at each triangle of the surface \mathcal{M} . At a triangle T_i , it is defined by an angle θ and three period-jumps (p_{ij}, p_{ik}, p_{il}) defined on the edges, where T_j, T_k, T_l are the neighboring triangles.

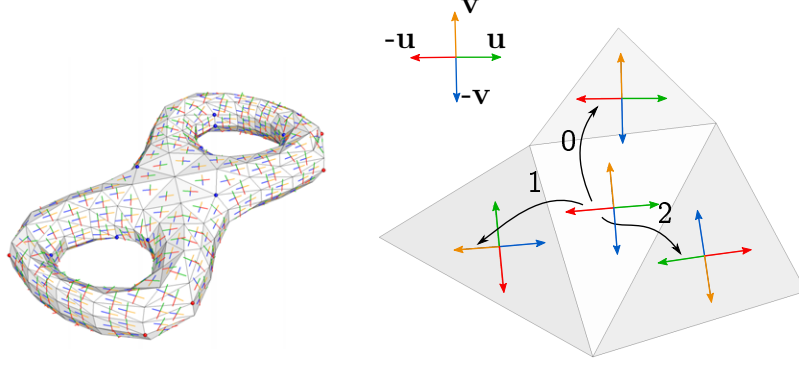


Figure 6.3: A cross field decomposed into four vector fields defined by the period jumps.

The angle θ is used to construct the frame in each triangle from a single edge direction. On a flat surface, the smoothness of a cross field is determined by the smoothness of the angle function between neighboring triangles,

$$E_{\text{smooth}} = \sum_{e_{ij} \in \mathcal{E}} (\theta_i - \theta_j)^2$$

where θ_i is the angle in the triangle T_i and θ_j is the angle in T_j when flattening T_j in the plane of T_i . On curved surfaces, we use the formulation of [BZK09] for the general smoothness energy incorporating the period jumps to detect singularities and the intrinsic coordinate transformation between both triangles,

$$E_{\text{smooth}} = \sum_{e_{ij}} \left(\theta_i + \kappa_{ij} + \frac{\pi}{2} p_{ij} - \theta_j \right)^2$$

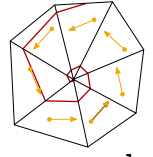
where $\kappa_{ij} \in (-\pi, \pi]$ is the curvature of the field from triangle T_i to T_j . It is computed as the difference of the angles of both frame in the same local coordinate system.

Theorem 6.2 (Local integrability conditions for frame fields). *Let \mathcal{M} be a simply connected simplicial surface. A triangle based frame field $\mathcal{F} = \{(\mathbf{u}_T, \mathbf{v}_T)\}_{T \in \mathcal{T}_M}$ is curl-free if as a vector field, $(\mathbf{u}_T)_T, (\mathbf{v}_T)_T, (-\mathbf{u}_T)_T, (-\mathbf{v}_T)_T$ are curl-free, taking the period jumps' assignment into account.*

This theorem also apply for poly-vector fields. For frame fields, it is sufficient to have $(\mathbf{u}_T)_T, (\mathbf{v}_T)_T$ curl-free. A proof of the theorem can be done using the branch covering approach presented in [KNP07] which transforms a frame field \mathcal{F} into a single vector field on a 4-sheeted covering of \mathcal{M} . The branch points of the covering are the singularities and the layer shifts are defined at edges where the period jumps are not trivial.

6.4 CURL-FREE VS NON-CURL-FREE

It is tempting to directly trace streamlines from the singularities of the frame field and apply the graph construction. However, we have to handle the occurrence of *limit cycles* which is caused by the curl of the field. Limit cycles are known in dynamical systems as closed paths such that at least one streamline is spiraling into them as time approaches infinity (see inset Figure for an example). They are not numerically stable. In our graph construction, it can then happen that a streamline never intersects another one making the matching problem infeasible. To avoid this issue, we remove the curl part of the frame field and use the isolines induced by the parameterization. Although, the isolines have infinite length, they are stable and as a helix, they have constant pitch which is very useful in our design of a stopping criterion.



Robust tracing of field guided streamline has been recently published [MPZ14,RS14] which can be used directly in our setting (see Chapter 10). They use a vertex based frame field which is not piecewise constant in each triangle. The field topology inside the triangles is then identified during the streamline tracings as well as limit cycles. In our setting, we use a per triangle based frame field which is piecewise constant in each triangle making our construction simple and robust. The isolines do not have finite lengths. We then propose a strong stopping criterion to assure the feasibility of the matching problem.

6.5 INJECTIVITY

In the computation of the curl free part of the input frame field \mathcal{F} , it is important that the local frames orientation are preserved to obtain an injective parameterization. This means that if

$$\langle \mathbf{u}_T, \mathbf{v}_T \rangle_T > 0 \quad \text{for all } T \in T_{\mathcal{M}} \quad (6.1)$$

then they all should stay positive after the curl removal.

Generating an injective parameterization is still a very active research topic producing a wealth of innovative ideas [Lip12, BCE⁺13, MPZ14, DVPSH15b]. They all attempt to approximate the non-linear and non-convex constraints relation 6.1 into a linear [BCE⁺13], convex cone constraint [Lip12] or by an iterative nonlinear curl reduction [DVPSH15b].

We propose a very simple approach to approximate the nonlinear constraints under the condition that the input frame field is smooth and the resulting curl-free field is not allowed to deviate more than 45° from the initial field. We call it a *bisector constraint* in contrast to the trisector constraint of [BCE⁺13]. Suppose that $(\tilde{\mathbf{u}}_T)_T, (\tilde{\mathbf{v}}_T)_T$ are the curl-free field derived from the initial fields $(\mathbf{u}_T)_T, (\mathbf{v}_T)_T$ (taking always period jumps into account). If $\tilde{\mathbf{u}}_T, \tilde{\mathbf{v}}_T$ satisfy

$$\langle \tilde{\mathbf{u}}_T, \mathbf{u}_T + \mathbf{v}_T \rangle > 0 \text{ and } \langle \tilde{\mathbf{u}}_T, \mathbf{u}_T - \mathbf{v}_T \rangle > 0 \quad (6.2)$$

and

$$\langle \tilde{\mathbf{v}}_T, \mathbf{u}_T + \mathbf{v}_T \rangle > 0 \text{ and } \langle \tilde{\mathbf{v}}_T, \mathbf{v}_T - \mathbf{u}_T \rangle > 0 \quad (6.3)$$

then the frame fields $\{(\mathbf{u}_T, \mathbf{v}_T)_T\}_T$ and $\{(\tilde{\mathbf{u}}_T, \tilde{\mathbf{v}}_T)_T\}_T$ have the same orientation in each triangle. The space of triangle deformations on the induced parameterization is composed by translations, scalings and rotations within $(-45^\circ, 45^\circ)$ which is smaller than the trisector space but works very well for open parameterizations.

6.6 CURL MINIMIZATION: POISSON PARAMETERIZATION

In practice, instead of removing the curl of a frame field $\mathcal{F} = \{(\mathbf{u}_T, \mathbf{v}_T)\}_{T \in \mathcal{T}_M}$, one looks for a potential $f = (u, v)$ whose gradient field $\{(\nabla u, \nabla v)_T\}_T$ is the closest to \mathcal{F} in a least square sense. f is called a *Poisson parameterization* and is computed by the least square minimization

$$\min_{f \in \mathcal{S}} \int_{\mathcal{M}} \left\| \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} - \nabla f \right\|^2 \quad (6.4)$$

resp. its discrete version of minimizing E_f

$$E_f = \sum_{T \in \mathcal{T}_M} \mathcal{A}_T \left(\|h \nabla u - \mathbf{u}_T\|^2 + \|h \nabla v - \mathbf{v}_T\|^2 \right), \quad (6.5)$$

subject to the following curl-free condition. If two triangles T_i and T_j share an edge e with edge direction \mathbf{e} , then

$$\mathbf{e}|_{T_j} = J^{p_{ij}} \mathbf{e}|_{T_i} \quad (6.6)$$

where \mathcal{A}_T is the area of the triangle T , h is a sizing field which controls the size of the grid patches, J is a 90° 2D rotation matrix and p_{ij} is the period jump inherited from the local smoothness of the frame field. The curl-free constraints means that the representation of \mathbf{e} in the chart of T_j is related by a 90° rotation to T_i .

The size of the system consists of $6|T_{\mathcal{M}}|$ (uv -coordinate per vertex per triangle) which is reduced by defining a dual spanning tree crossing a primal edge if the period jump from the current triangle to the neighboring triangle is zero. The remaining primal edges form a graph connecting the singularities and defining a boundary which makes the surface a topological disc. Once an initial triangle of the tree is mapped to 2D, all the triangles along the spanning tree are determined. Hence, only those along the cut path has to be solved by a global optimization approach. To assure injectivity, we use the inequality constraints 6.2 and 6.3. All these procedure are illustrated in Figure 6.4, the implementation details can be found in the Appendix.

From this point, we will always use the metric induced by the Poisson parameterization on the surface \mathcal{M} . We denote this open parameterized surface by \mathcal{M}_f . Bommers et al. [BCE⁺13] uses a decimation algorithm on Poisson parameterized surface in order to reduce the size of their integer quadratic program. In the layout optimization of Campen et al. [CK14], a multiple computation of the Poisson parameterization is used to reduce the global parameterization distortions by moving singularities. The structure of \mathcal{M}_f is not far from an integer-grid map, i.e. a seamless continuous parameterization. The singularities have the same angle defects and port enumerations as for rounded parameterizations. The difference lies in the local integrability of \mathcal{M}_f . Isolines of \mathcal{M}_f do not close and have infinite length. In contrast to direct field tracing approaches, tracing isolines on \mathcal{M}_f is relatively simple. Deciding if two isolines intersect or not can be done by classical 2D intersection checks. \mathcal{M}_f is governed by transition functions, which have to be taken into account, for each line and patch construction. For example, to compute a locally shortest geodesic path between two points, we use the algorithm proposed by Lee and Preparata [LP84]. For a given triangle run containing the two points, instead of an isometric unfolding, we use the parameter domain of \mathcal{M}_f using transition functions. In parameter domain, singularities behaves like small holes such that geodesics get stuck on these special points unless the homotopy type of the geodesic is changed.

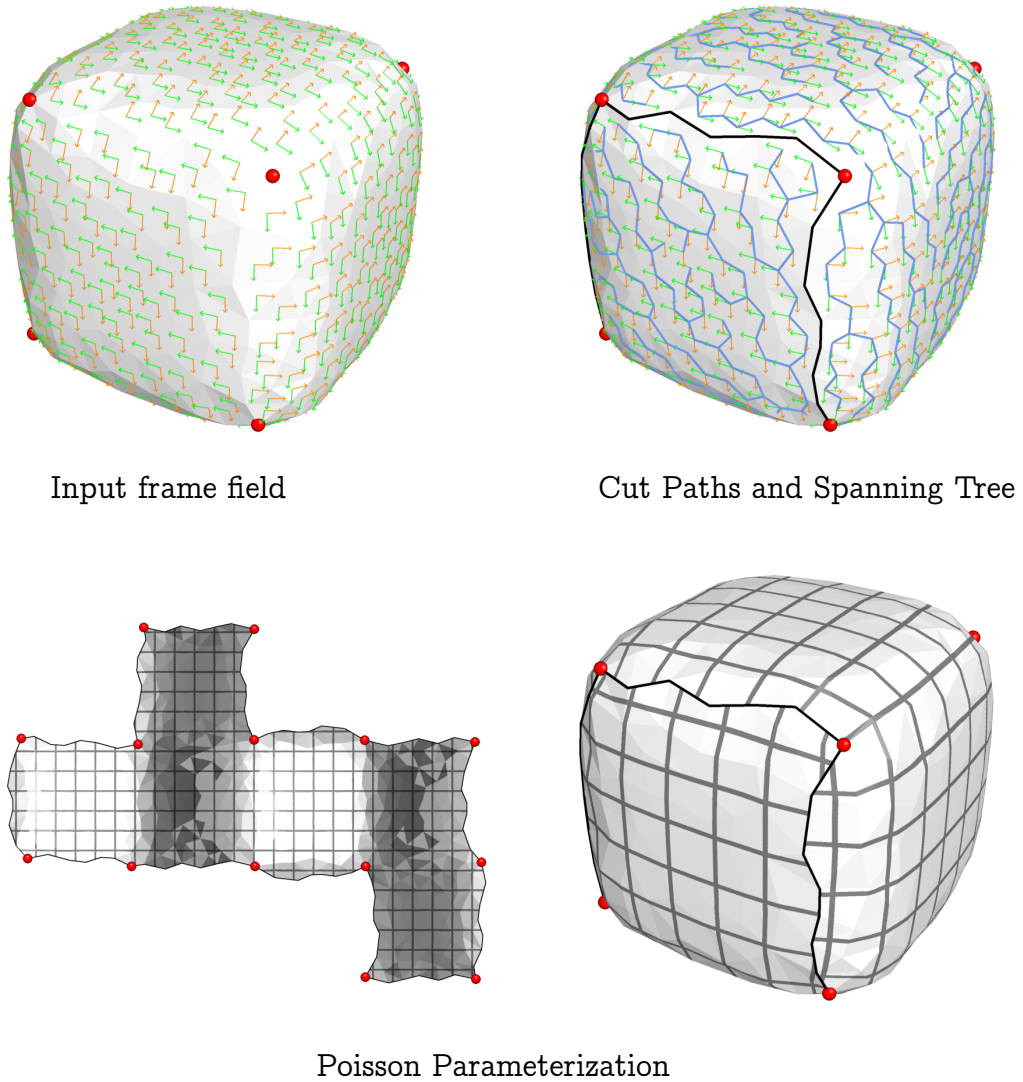


Figure 6.4: Computation of a curl-free frame field inducing a global parameterization of the surface \mathcal{M} .

6.7 GRAPH CONSTRUCTION

The graph construction follows Chapter 4 where the charts are represented by the triangles. In Figure 10.3 is an example of a port connection along an isoline. Non singularity-free triangles must be detected along the tracing line.

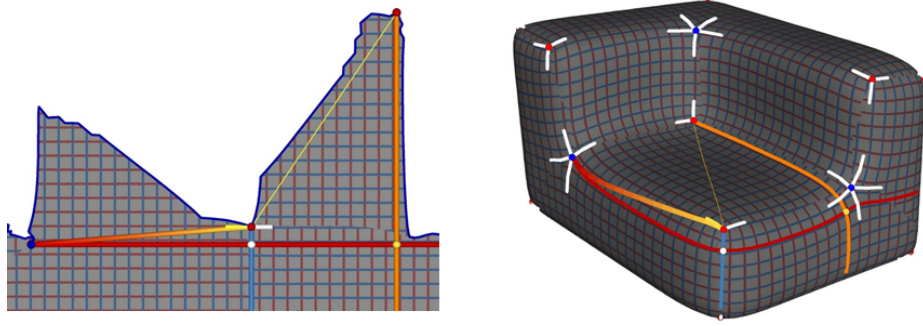


Figure 6.5: Tracing of an isoline on a parameterized surface and evaluation of ratios of lengths at the intersection points.

6.8 STOPPING CRITERION

In general, the isolines of \mathcal{M}_f have infinite length. The line growing approach can never end unless either all open separatrices end at a boundary, or in the very rare case where they all hit singularities. This situation happens, for example, when the surface is very symmetric. We need a correct way to stop the growing procedure and make sure that enough edges are in \mathcal{G} in order to get a large space of possible solutions.

One approach is to fix a maximum growing length L_{\max} relative to the diameter of the geometry and to terminate when all lines reach the prescribed length. This approach will produce enough edges in \mathcal{G} but can be very slow in practice, especially if \mathcal{M}_f contains too many singularities. A more elegant approach is an adaptive stopping criterion. Mainly, a line stops growing while others continue. The main observation is that the isolines of \mathcal{M}_f are spiralling at some region. Our termination condition is to stop an isoline when it starts to spiral. However, this procedure involves the careful understanding of the structure of spirals (a priori unknown) on \mathcal{M}_f and a robust algorithm to detect them as done in [BLK11] on quadrilateral meshes.

If a growing line intersects another one the second time, then one of them is suscep-

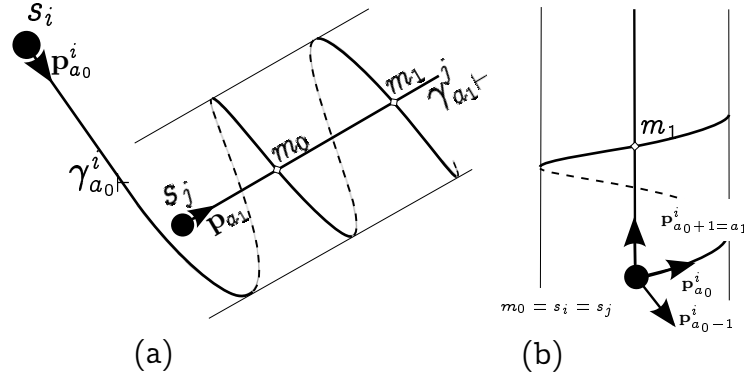


Figure 6.6: Spiral detection during the growing procedure: (a) a separatrix starting from a far way singularity before spiralling in a cylindrical region; (b) separatrices starting at a cylindrical region based at the same singularity.

tible to be a spiral. It is then convenient to stop it. The decision depends on a fixed threshold ρ controlling the pitch of helices of \mathcal{M}_f . Consider two open separatrices $\gamma_{a_0+}^i$ and $\gamma_{a_1+}^j$ which intersect each other the first time at a point m_0 and the second time at a point m_1 of \mathcal{M}_f . If

$$|L_{a_0}^i(m_1) - L_{a_0}^i(m_0)| > \rho |L_{a_1}^j(m_1) - L_{a_1}^j(m_0)| \quad (6.7)$$

then $\gamma_{a_0+}^i$ stops growing. Else if

$$|L_{a_1}^j(m_1) - L_{a_1}^j(m_0)| > \rho |L_{a_0}^i(m_1) - L_{a_0}^i(m_0)| \quad (6.8)$$

then $\gamma_{a_1+}^j$ stops growing. If $i = j$ and $a_1 = a_0 \pm 1$, then the inequality is checked at the first intersection point of the two separatrices (see Figure 6.6). In that case, $m_0 = s_i = s_j$.

If ρ is large, then the growing step might collect too many unwanted large pitch spirals affecting considerably the size of \mathcal{G} and the construction's running time. If ρ is too small, then some isolines might stop too early even if they do not start to spiral, reducing the size of \mathcal{G} and hence the space of possible solutions. In our implementation, choosing the fix value $\rho = 10$ works very well.

In Figure 6.7 are two examples of stopping strategies. In the first approach, we use a fix maximum tracing length L_{\max} for all isolines which produces many unwanted spirals while in the second approach, we trigger the spiral detection during the tracing which removes all unwanted spirals produced by the first approach.

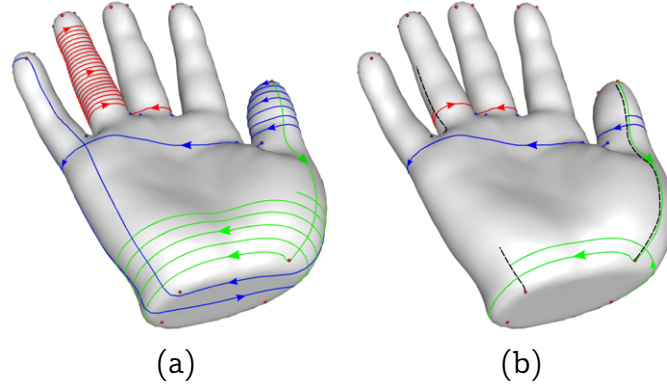


Figure 6.7: (a) Example of spirals for a fixed maximum tracing length L_{\max} ; (b) isolines are stopped before spiraling using $\rho = 10$ in parameteric length unit.

6.9 RESULTS AND ANALYSIS

We apply our algorithm on several datasets which are mainly classical benchmark models. The input of the algorithm is either a frame-field or directly a Poisson parameterization \mathcal{M}_f . We use default parameters for all quad layout generated on the benchmark models by choosing a balance $\alpha = 5$, a maximum separatrix length $L_{\max} = 3/h \times \text{Diam}(\mathcal{M})$, a sizing field $h = 10^{-2} \times \text{Diam}(\mathcal{M})$, and a spiral detection constant $\rho = 10$. The choice of α is motivated by the experiment illustrated in Figure 6.8 but the use of different values of α is also encouraged. For the benchmarking, we use the frame-fields also taken as input from the corresponding previous works.

Data structure In our implementation, we use a triangle based data structure to efficiently generate the graph \mathcal{G} and the set of illegal constraints I . Each triangle knows all open separatrices going through it together with the intersection points used in the construction of I . The graph \mathcal{G} is represented as an adjacency matrix of the ports whose entries are zero or one using classical sparse matrix format. For each non zero entry of that matrix, we store the corresponding triangles and their geometries for a fast visualization of the layouts on the geometry.

Robustness The algorithm runs very well on badly triangulated meshes. In Figure 6.9 (a) is a bad triangulation of the Joint model which contains many skinny triangles. Our method still produces correct quad layout on that model. We also

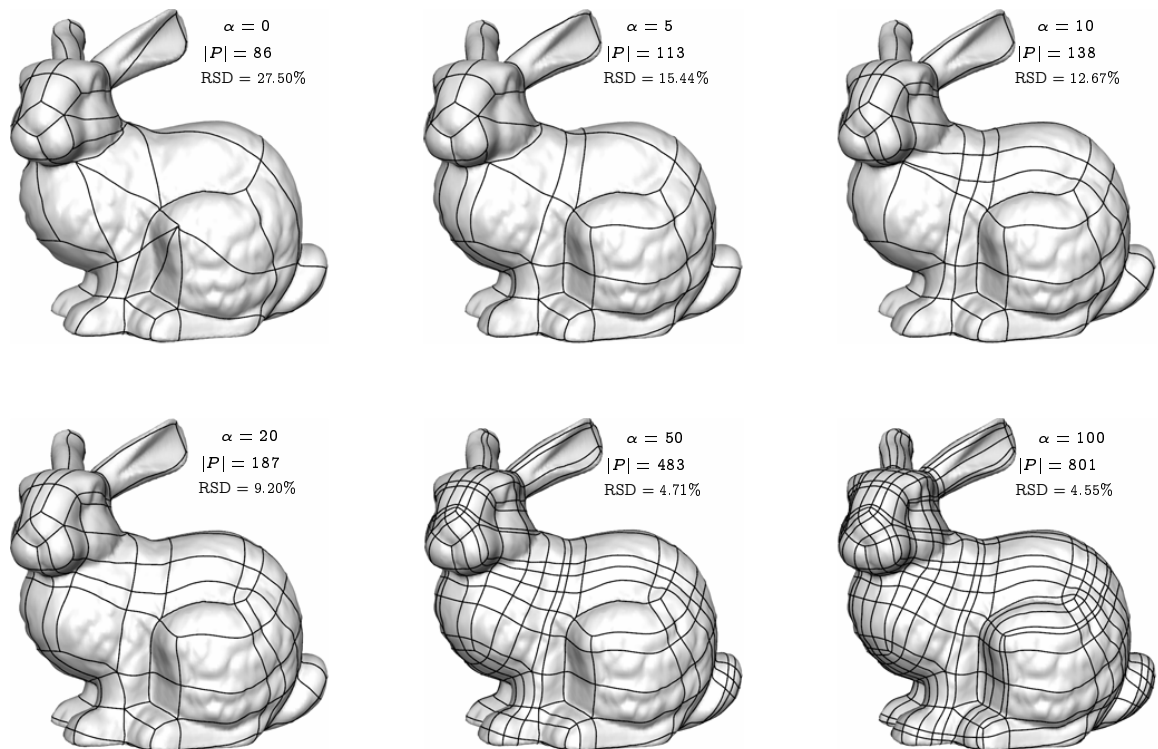


Figure 6.8: Several quad patches on the Bunny model for different values of α . $|P|$ denotes the number of patches and RSD is the Relative Standard Deviation of the patch angles measured in parametric space. Big values of α give a better alignment to the input field at the cost of increasing number of patches. We find $\alpha = 5$ to be a good balance.

generate random noises on some region of the Bumpy Sphere model with bad shaped triangles, over-foldings and self-intersections. The algorithm still produces valid quad patches, as illustrated in Figure 6.9 (b).

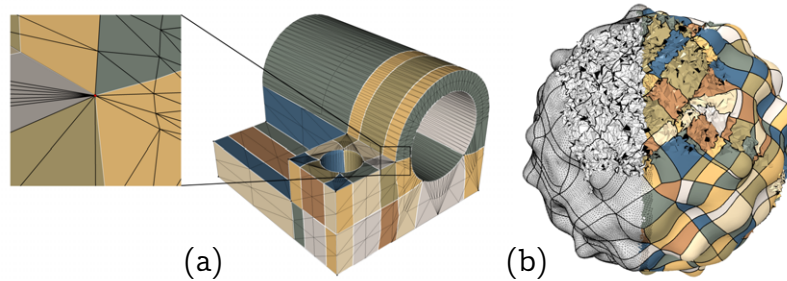


Figure 6.9: (a) A bad triangulation of the Joint model. (b) Exaggerated noise on the Bumpy Sphere model.

Singularity placement In the quest of a good quad layout, the position of singularities is very important to assure geometrically meaningful patches. Our algorithm is not very sensitive to local changes of singularity positions but can generate a completely different layout (but still optimal with respect to the new singularity positions) if the change induces a global effect on the field as illustrated in the Figure 6.10.

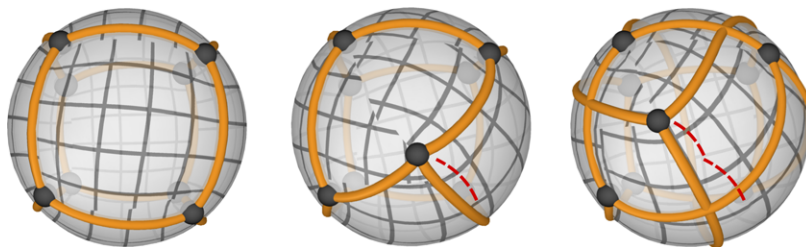
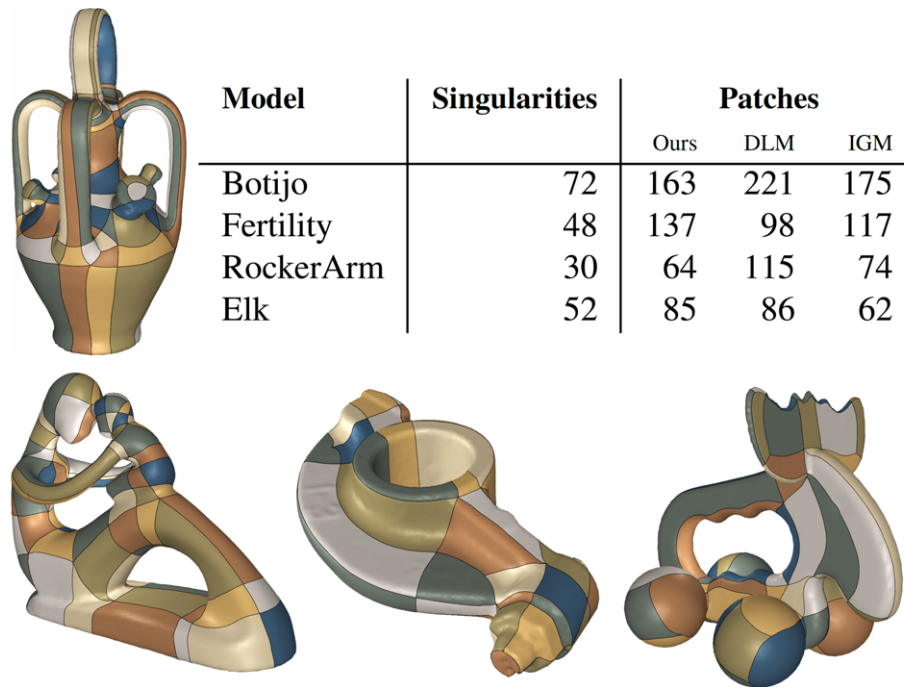


Figure 6.10: Computation of the Layout under a local movement of a singularity.

6.10 COMPARISON

We compare against the Dual loops meshing (DLM) [CBK12] and the Integer-grid map (IGM) [BCE⁺13] for the same models and cross-fields. The results are shown in Figure [?] together with a comparison table. Our algorithm produces results as good as both methods. The main advantage of our approach is its speed and

control over the final layout. Changing the value of α or adding user constraints can be done in a split second. The DLM approach is greedy and works on the dual complex. It is not very much affected by singularity placement but unfortunately, it is not clear how user constraints could be incorporated once the layout is computed. The integer-grid map approach finds directly an optimal solution without any balancing parameters. This method, unfortunately, requires a fold-over free Poisson parameterization where in our case it is only needed for a very small set of triangles. Our method also gives a balancing parameter which makes the layout computation more flexible. In contrast to IGM, our method is guaranteed to be spiral-free which on one hand allows smaller patch numbers but on the other hand can produce nonuniform dense layout (see Appendix). In Table 7.2 is an overall statistics of our algorithm. The timing does not include the frame field generation nor the Poisson parameterization. We consider these two steps as preprocessing and they can be taken as input to the algorithm. In our implementation, solving P1 is in average one second making the computation of a new solution extremely fast.



Model	Singularities	Patches		
		Ours	DLM	IGM
Botijo	72	163	221	175
Fertility	48	137	98	117
RockerArm	30	64	115	74
Elk	52	85	86	62

Figure 6.11: Our algorithm applied to benchmark models also taken as input by DLM and IGM.

In Figure 6.12 is a close comparison of the layouts generated by IGM [BCE⁺13] (left) and our method (right). On the Botijo model, we noticed that the layout

generated by IGM still contains spirals (on the handle) where on our layout, this is ignored. The IGM has, on the other hand, a better node connection on the Elk model. From the Figures, we can clearly see that IGM tends to optimize for regularity i.e. close to 90° angles at the intersection points.

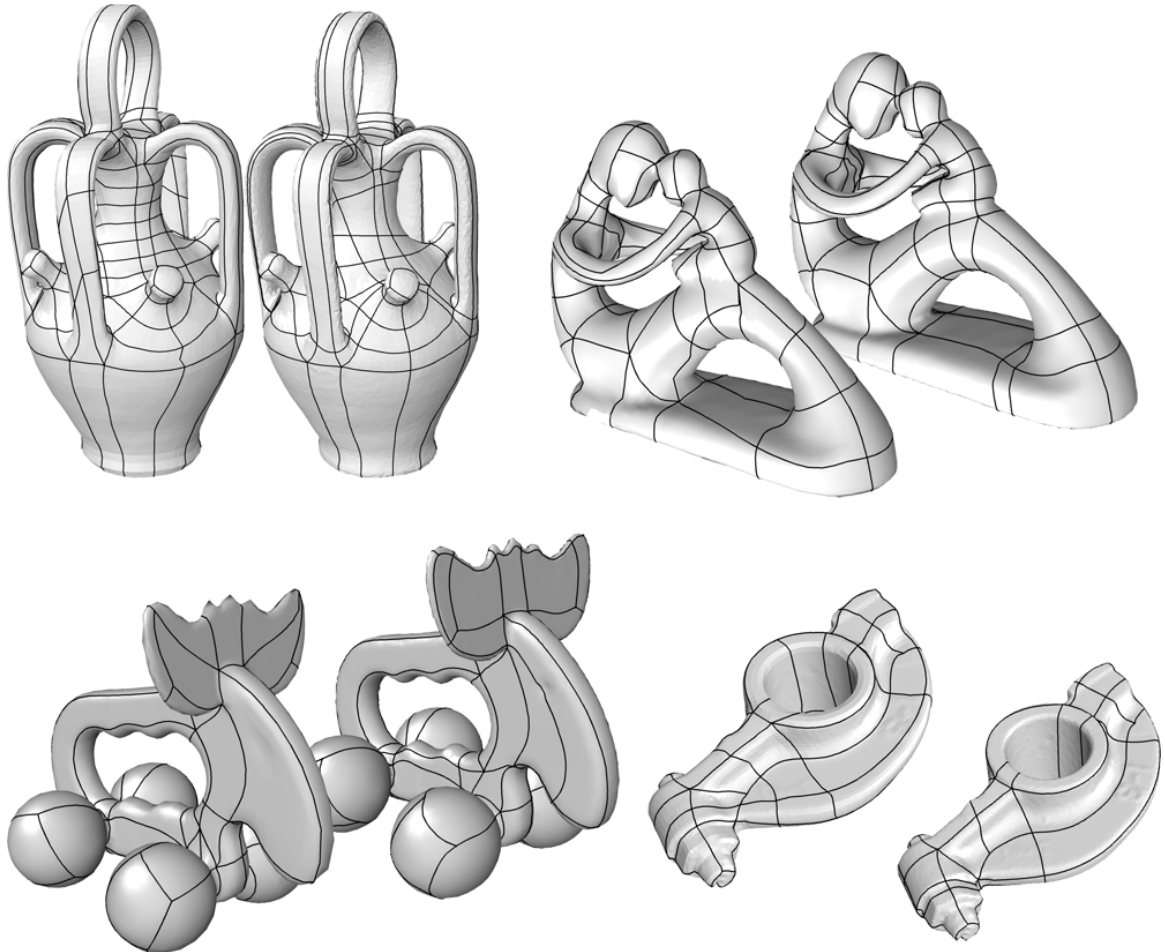


Figure 6.12: Comparison of our method (right) to IGM (left). The layouts of IGM corresponds to the base complexes of each model on the quadrilateral meshes provided by IGM’s authors which make the renderings sub-optimal but the layouts are exactly the same as in [BCE⁺13].

In Figure 6.13 are other models, where the frame fields are generated by our own implementation. The models are especially chosen to show reliability of the proposed approach even for large sized models. We did not add any sharp edge constraints for the Casting model. The resulting layouts are naturally produced by the fix value of $\alpha = 5$.

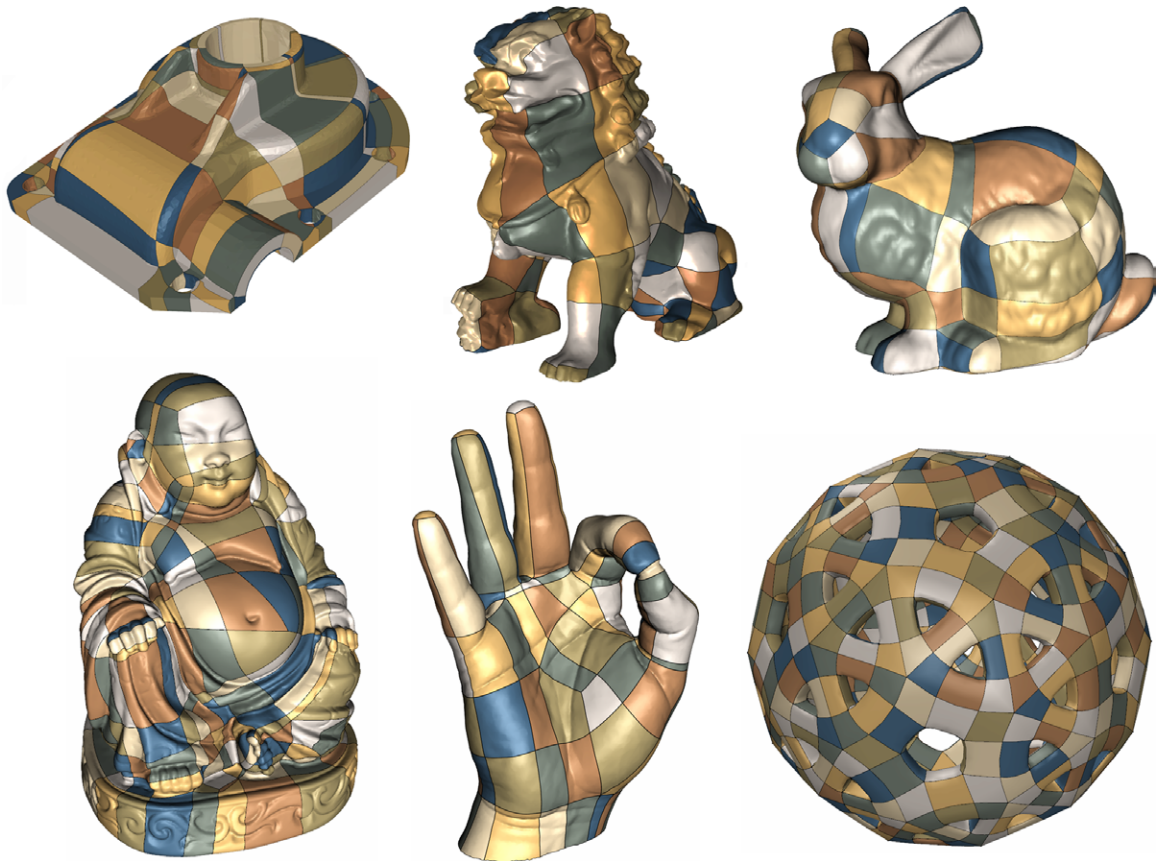


Figure 6.13: Perfect matching quad layouts generated on the Casting, Dragon, Bunny, Boudha, Hand and the Genus-91 models.

Models	$ T_{\mathcal{M}} $	$ S $	Graph		Av. length $1/N \sum \ \gamma_{a_i}\ _f$	$ P $	RSD angles (%)	Time in s	
			$ E_G $	$ I $				P_1	Layout
Torso	74628	28	528	7892	162.10	42	17.71	0.12	4.61
Sofa	29020	14	229	2162	68.23	24	4.90	0.06	1.73
Fertility	27954	48	1118	18831	80.33	137	15.84	1.09	2.54
Elk	18308	52	1023	14150	95.04	85	15.21	0.46	1.91
Botijo	29994	74	1736	32192	204.04	163	17.76	0.64	5.77
RockerArm	70318	30	609	9441	127.49	64	18.65	0.15	5.13
Joint	1784	24	255	6085	55.96	79	7.32	0.03	0.61
Bumpy sphere	102996	150	4194	122991	184.82	480	17.78	2.71	63.74
Boudha	198736	70	1804	48888	161.72	255	17.45	1.93	55.51
Genus-91	28800	720	15903	151430	86.46	1500	6.29	3.85	31.22
Casting	66914	100	2683	83489	87.21	235	18.92	1.89	14.59
Hand	99999	38	763	20297	84.36	117	18.26	1.03	14.64
Bunny	69666	60	1347	23939	119.52	113	14.95	0.43	8.48
Dragon	102400	44	1006	22412	283.81	105	14.15	0.37	23.30
Bimba	149524	76	1944	64551	141.27	231	20.37	1.95	59.08

Table 6.1: Statistics of the proposed method applied to several benchmark models. $|S|$ is the number of singularities, $|P|$ the number of patches and RSD the patch angles relative standard deviation measured in parameter space. P_1 is the time taken to solve the constrained minimum weight perfect matching. Layout denotes the time needed to compute a solution from a given Poisson parameterization (without per patch colorings).

CHAPTER 7

APPLICATION II: QUAD MESH STRUCTURE OPTIMIZATION



In many application scenarios, surfaces are often given as quadrilateral meshes where the global structure characterized by the base complex of the mesh is pre-defined. These base complexes have in most cases too many patches. Coarse base complexes for quadrilateral meshes are for example very important in the quad mesh modeling of [Vax14]. Several algorithms have been proposed to optimize the base complex of a given quad mesh. Bommès et al. [BLK11] propose to remove helical configurations present in the mesh using a state machine composed by local atomic

operations such as shifts and edge collapses. Tarini et al. [TPP⁺11] disentangle the graph of separatrices in order to get another graph with less separatrix energy and hence less patches. Both methods preserve the singularities of the initial quad mesh. Other methods [DSSC08, TPC⁺10] use quad mesh simplification techniques aiming at a coarse base mesh of the surface and allowing singularities to collapse. Since our method preserves the initial singularities, we compare our results to Bommes et al. and Tarini et al. The main advantage of our approach is first optimality, we formulate the problem as a global optimization problem and second simplicity, implementing our algorithm only requires a binary program solver.

7.1 PROBLEM

The main issues appearing in automatically generated quad meshes are the so called *near misses* illustrated in Figure 7.1. A near miss characterizes a parameter line which misses to connect to a near by singularities increasing the number of helical configurations and hence the number of patches in the base complex of the mesh. By rating those near misses, Tarini et al. use a trial and error approach which repairs the near misses locally and finds a global closing moves to preserve the quad topology of the base complex. Quadrilateral meshes are examples of closed

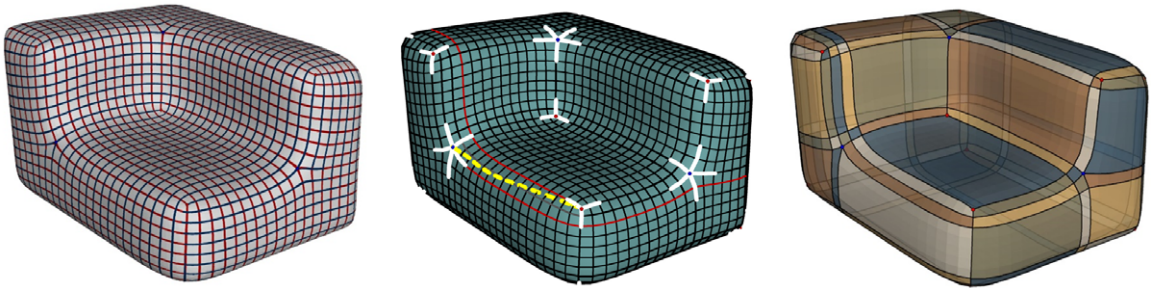


Figure 7.1: Problematic cases on automatically generated quad meshes characterized by near misses which induce the increased number of patches in the base complex of the mesh.

parameterized surfaces where the charts are quads and the metric is the integer metric on the edges. The graph construction can then be applied directly on the surface without the need of an input guiding field or a parameterization.

7.2 PRELIMINARY

On a quad mesh \mathcal{M} , a vertex is regular if it is the intersection points of four quads or two quads for the case of a boundary. It is singular otherwise. The valence of a vertex is the number of quads sharing the vertex. In our setting, we use the integer metric induced by the quad grid where the distance between two neighboring vertices is always one. The ports are simply the normalized edge vectors incident to the singularities.

We call a *trivial base complex* the base complex \mathcal{B} of \mathcal{M} obtained by the union of all discrete parameter lines connecting two singularities. The construction of \mathcal{B} is straightforward but the induced quad layout often has too many patches. In a sense, we would like to measure the quality of \mathcal{B} in terms of coarseness and geometric feature alignment but we provide on top a better base complex in case it is not optimal.

7.3 ALGORITHM OVERVIEW

The construction of \mathcal{G} is made along the separatrices of \mathcal{B} following the ratio conditions for the creation of new edges in \mathcal{G} . The main step of our algorithm is summarized as follows.

1. Generation of \mathcal{B} .
2. Extension of \mathcal{B} to a supergraph \mathcal{G} .
3. Construction of the set of illegal edge crossings I .
4. Finding of a quad layout \mathcal{Q} of \mathcal{M} as a MWPMPC of \mathcal{G} .

We generate \mathcal{B} by tracing all parameter lines emanating from the singularities of \mathcal{M} until other singularities or boundaries are reached. It is equivalent to removing all loops in \mathcal{M} and keeping only the remaining parameter lines. If the input mesh has a "good" connectivity information i.e. no near misses, then \mathcal{B} can be already the "best" quad layout of \mathcal{M} . In practice, quad meshes are generated either by frame aligned global parameterization based methods [KNP07, BZK09] or spectral based method [DBG⁺06, ZHLB10] or direct methods [TPC⁺10, RLS⁺12] which does not give a priori high quality base complexes due to their greedy nature.

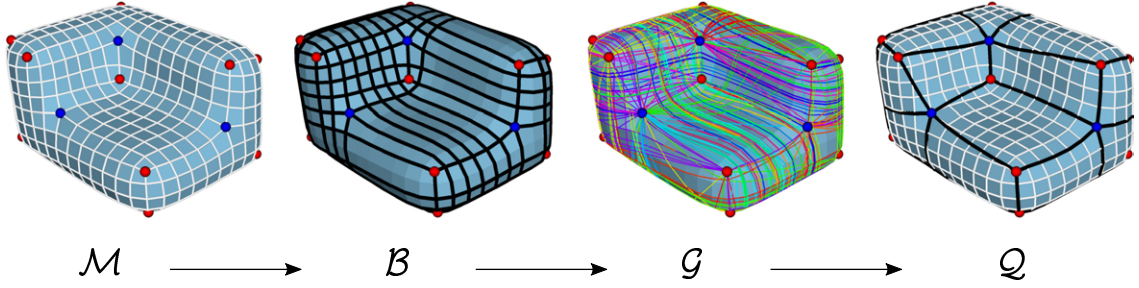


Figure 7.2: Main steps of the quad layout optimization on quadrilateral meshes.

We extend \mathcal{B} to a supergraph \mathcal{G} by evaluating at each regular vertex of \mathcal{M} along the separatrixes of \mathcal{B} the ratio of lengths of the two separatrixes intersecting at that vertex following the graph construction in Chapter 4. The set of illegal edge crossings I is constructed at the regular vertices along the separatrixes of \mathcal{B} where two closed separatrixes intersect. Finally, the optimum quad layout according to the value of α is obtained using Theorem 5.1 in Chapter 5.

7.4 GRAPH PROPERTIES

By construction \mathcal{G} has $\sum_{s \in S} \text{valence}(s)$ vertices where S is the set of singularities of \mathcal{M} . It has at most $(4|V_{\mathcal{M}}| + |E_{\mathcal{B}}|)$ edges, where $|V_{\mathcal{M}}|$ is the number of regular vertices of \mathcal{M} and $|E_{\mathcal{B}}|$ the number of separatrixes in \mathcal{B} . The upper bound is reached when all regular vertices of \mathcal{M} lie on a singularity-free triangle. \mathcal{G} can have multiple edges i.e. two vertices of \mathcal{G} can be connected with several edges with different weights. In Figure 7.3, a spiraling separatrix can intersect another separatrix several times, all intersections satisfying the singularity-free conditions and the ratio test. These helical configurations are intensively studied in [BLK11] which are known to increase significantly the number of patches in the quad mesh's base complex.

We could already choose which one of these edges we add to \mathcal{G} according to their weights, but it implies the reconstruction of \mathcal{G} for each value of the parameter α which is not computationally efficient. Hence we keep them all, at the cost of a bigger graph size. If \mathcal{B} has no near misses, then \mathcal{G} can be equal to \mathcal{B} . In this case no regular vertices along the separatrixes satisfy the singularity-free and the ratio lengths conditions. \mathcal{B} is then the best quad layout of \mathcal{M} .

Datastructure For the implementation, we use a vertex-based datastructure for a quick evaluation of the ratios and illegal crossings. Each non singular vertex of

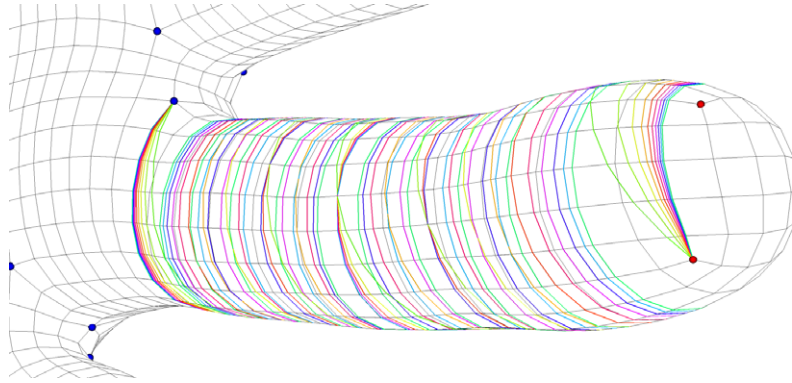


Figure 7.3: Two ports are connected by several separatrices on a region with helical configuration.

the base complex stores the two intersecting separatrices together with the corresponding port directions. The tracing starts at a singularity and goes successively along the vertices of the base complex in direction of a port until another singularity or a boundary is reached. The window reduction strategy is used to avoid non singularity-free triangles.

7.5 RESULTS AND ANALYSIS

We apply the proposed quad layout optimization on several datasets which are classical benchmark models. For rendering purpose, we add one or two refinement steps to each of them which does not affect the base complex but slightly the computation running time. The only parameter in our framework is α .

Choosing α In general, the choice of α depends on the user requirements as in the triangle mesh case. To get a reasonable default value of α , we did sequences of test ranging from 1 to 100. The selection criterion is again based on the Relative Standard Deviation (RSD) of the angles of the patches with respect to the original quad mesh against the number of patches. We choose the *Fertility* model with 21436 quads and 42 singularities. The initial base complex of the model induces 11432 patches with $RSD = 8.30\%$. In Figure 7.5 are the series of tests where we found $\alpha = 5$ to be a very good balance similar to the triangle mesh case.

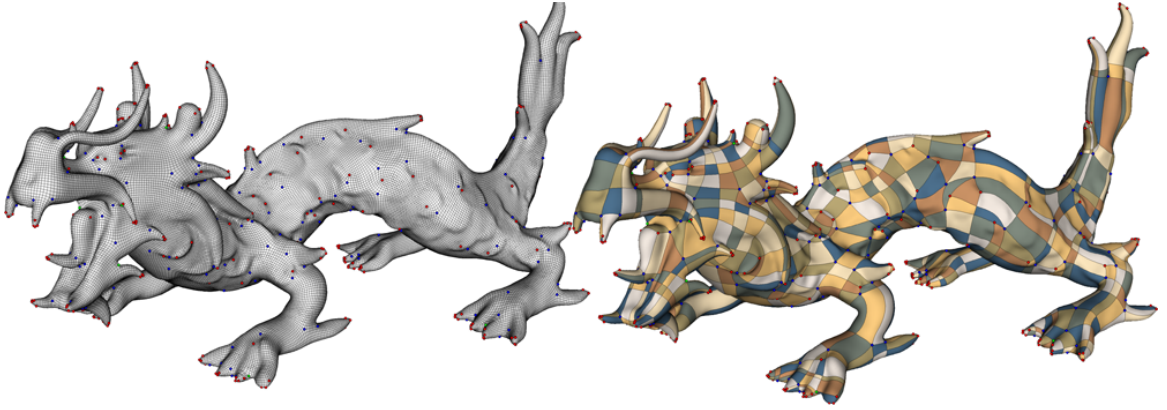


Figure 7.4: The Dragon model, generated by [ECBK14], has 18632 patches from its base complex with 611 singularities. Optimized to 2760 patches using our optimization approach.

Robustness A common approach to test the robustness of the algorithm is to use models with several singularities. In Figure 7.4 is a Dragon model, generated by [ECBK14], which contains 611 singularities with initially 18632 patches from its base complex. The number of patches is optimized by the algorithm to 2760 within 8 seconds.

Isosceles Triangles In the graph construction, it can happen that the ratios of length of the two intersecting separatrices is one i.e. $L_{a_0}^i(m) = L_{b_0}^j(m)$, where m is the intersection point of the two separatrices $\gamma_{a_0+}^i$ and $\gamma_{b_0+}^j$. We handled this case by adding extra disjunctive constraints to the matching problem. Nevertheless, the generated diagonal edges will always have less separatrix energy independent of the values of α . They are minimum in both length along the tracing separatrices and offset deviations. They reduce the number of patches of the initial base complex at the cost of high angle deviations at some singularities which cannot be resolved unless α is very large as illustrated in Figure 7.6.

A solution to this problem is for example to allow only far apart singularities to be diagonally connected. This is a parameter dependent since closeness is not well defined. A post validation could also be applied by identifying these configurations, removing the corresponding edges from the graph and recomputing a solution. In our implementation, we did not do any of these optimizations. We keep diagonal edges to avoid missing some interesting singularity layouts (see Figure 7.7).

In Figure 7.8 are examples of classical benchmark models optimized by our al-

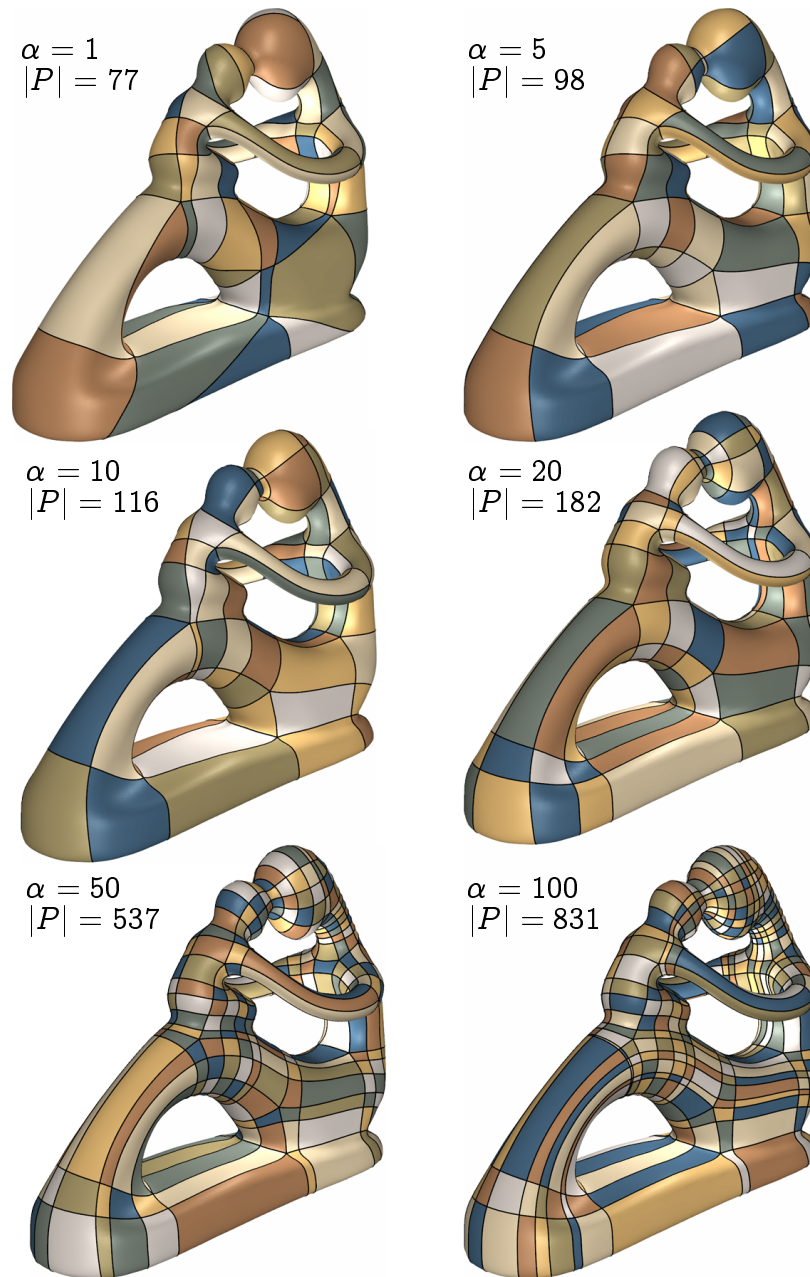


Figure 7.5: Quad layouts on the Fertility model for different values of α . Smaller values of α tend to produce coarse quad layout at the cost of feature deviation. Taking large values of α produces a nice feature aligned edges but unfortunately produce too many patches. We find $\alpha = 5$ is a good balance.

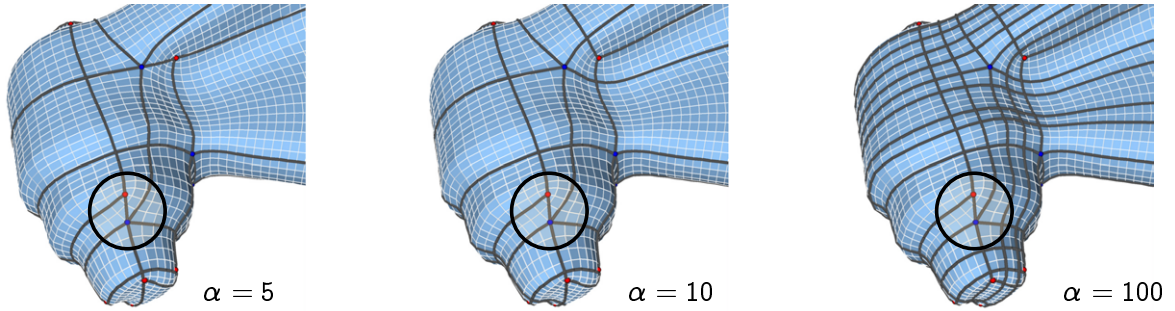


Figure 7.6: Effect of allowing diagonal connections which introduce big angles distortions at singularities that cannot be resolve by increasing the value of α .

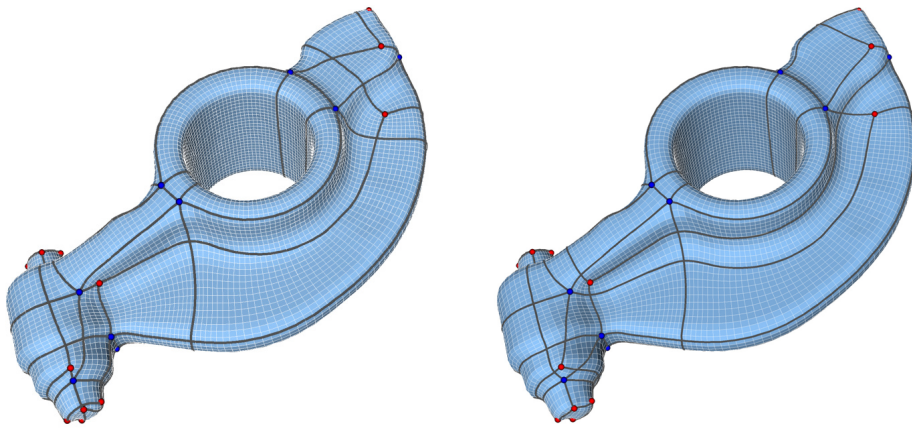


Figure 7.7: Comparison of the singularity layout generated on the RockerArm model: (left) allowing diagonal edges, (right) avoiding diagonal edges.

gorithm. On the FanDisk model, the sharp edges are naturally preserved in the solution. The graph construction on quadrilateral meshes is much faster than on triangle meshes. But because of multiple edges, the size of the illegal crossing constraints is large and hence it may take time for the solver to find a feasible solution.

7.6 COMPARISON

Simple Quad Domains Tarini et al. [TPP⁺11] propose a powerful heuristic to simplify the trivial base complex into another base complex with less energy. Compared to our approach, their algorithm collects potential candidate ports within the corridors of the separatrices (the first left and right parallels separatrices delimited by singularities). They check if connecting to one of the candidate ports decreases

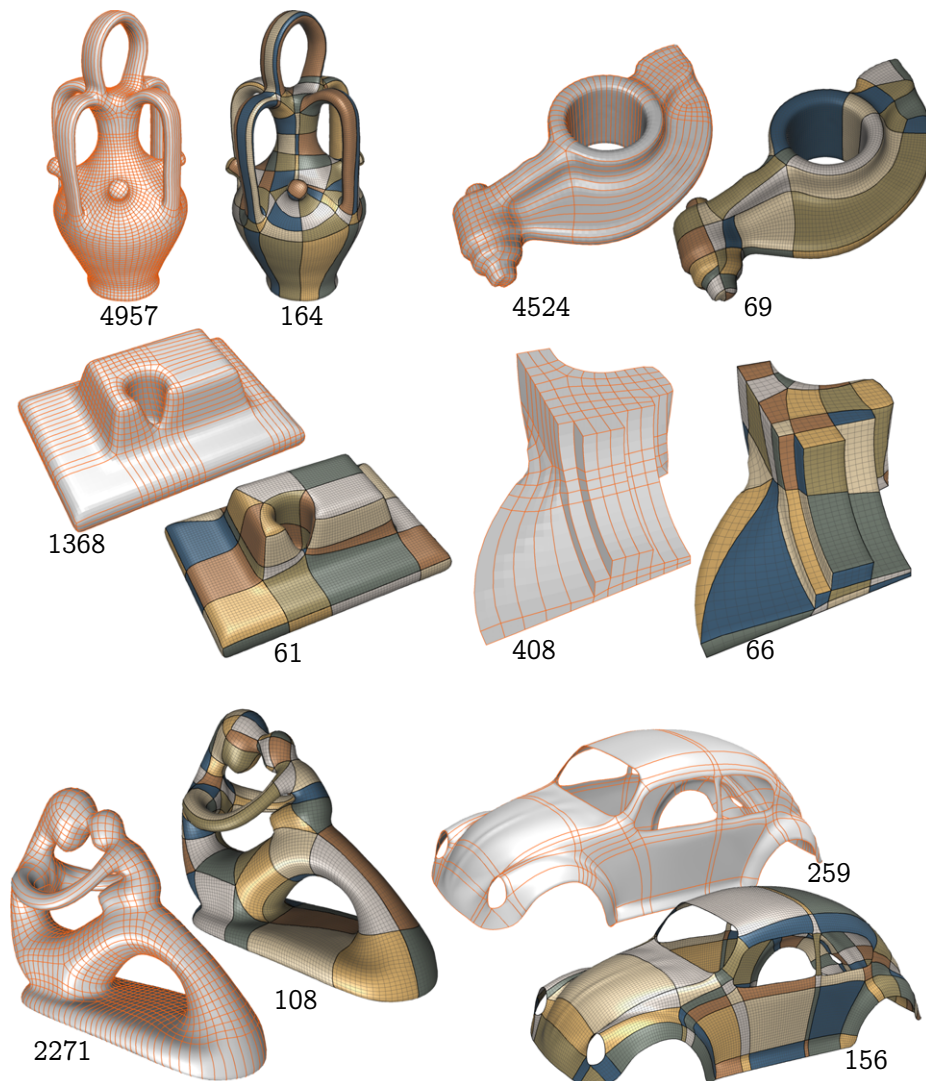


Figure 7.8: Several classical benchmark quadrilateral meshes optimized by our algorithm. The models were generated by [BZK09]. The models with the orange grid lines are the base complexes of the original models.

the global energy and try to find a closing move which keeps the global quad topology of the base complex. If a closing move is not found, then the configuration goes back to the last best move. A tree of all previous decision is then built defining all local valid configurations until the last moves. A main drawback of their approach is its greedy nature. Optimality is not guaranteed and valid configurations are only driven by the existence of global closing moves. In contrast to our approach, we are

not limited to the corridor of separatrices. We include ports outside the corridors which satisfy the deviation and singularity-free triangle constraints. Instead of a trial and error approach, we build a graph of all possible layouts of the surface induced by the trivial base complex and formulate a global optimization problem. Nevertheless, the method proposed by Tarini et al. does not involve an integer program. Their algorithm can be efficiently implemented and produces respectable results.

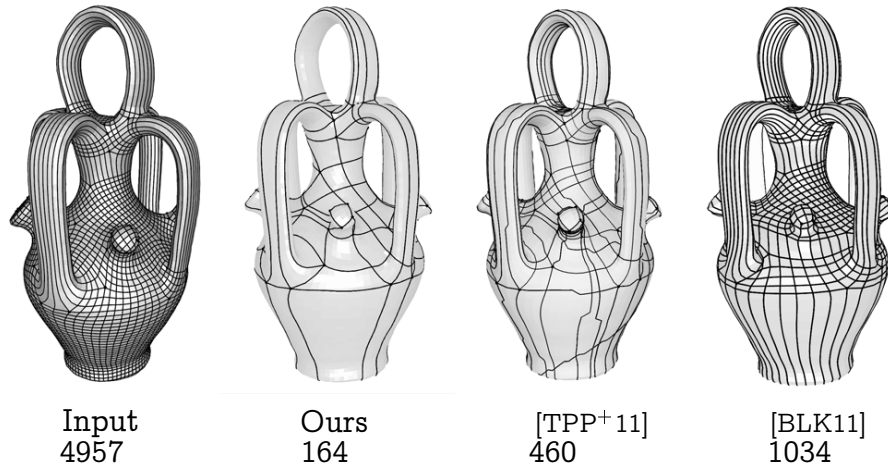


Figure 7.9: Comparison of our methods to recent state of the art quad mesh optimization algorithms.

Global Structure Optimization Bommes et al. [BLK11] propose to reduce the number of helical configurations present in the initial quad mesh which are the main cause of increased number of patches. They used a state machine composed by successive atomic operations. These operations are local mesh modification such as flips and collapses. The first step of their algorithm is to detect the helices and remove those which agree with the state machine. This approach does not need a reparameterization which in our case and Tarini's case is necessary. The state machine modifies directly the input quad mesh to another quad mesh. However, there are still helices in the final layout of their solution. The state machine is greedy and is not guaranteed to give the best possible layouts of the surface.

In Figure 7.9 is a comparison of our method to both methods. Visually and based on the number of patches, our method generates less patches and still aligns with the features of the geometry. We conjecture that the solutions of Tarini et al. and Bommes et al. are perfect matching subgraphs of our graph which do not have

minimum weight.

Model	S	F of \mathcal{B}	Optimized			RSD	Total
			Ours	SQD	GSO	angles (%)	time (s)
Botijo	74	4957	164	460	1034	27.27	1.14
Fertility	48	2271	100	253	526	28.37	0.78
Rockerarm	36	4524	64	98	178	28.36	0.26
Drill hole	26	1368	61	82	216	22.83	0.88
Fandisk	30	408	66	88	144	16.26	0.18
Beetle	56	259	156	-	-	15.20	0.07

Table 7.1: Comparison to other quad mesh optimization algorithm with the same input models also taken by [TPP⁺11] and [BLK11]. \mathcal{B} denotes the base complex of the quad mesh.

Parameterized Triangle Meshes. Given the recent development of robust quad mesh extraction [EBCK13] for frame-field guided parameterized triangle meshes, it is natural to ask if generating a quad mesh using [KNP07] and applying the global structure optimization is more suitable than working on multi-chart parameterized surface. This will avoid the needs of a stopping criterion and will increase the computation speed. The quad layout generated by our algorithm on triangle meshes and quad meshes are not necessarily the same. First, the graph generated from the Poisson parameterized surfaces does not contain diagonal edges where they appear frequently on quad meshes. Second, the rounding step in the integer-grid mapping already predetermined the number of singularity triangles. Matching subgraphs are not necessarily isomorphic in both cases. Since we allow multiple edges to appear on quad meshes, the number of the disjunctive constraints can be very large compared to the triangle mesh case. In Figure 7.10 is a close comparison where the MWPMPC is computed in 8 seconds on the quad mesh and in 0.86 seconds on the triangle mesh.

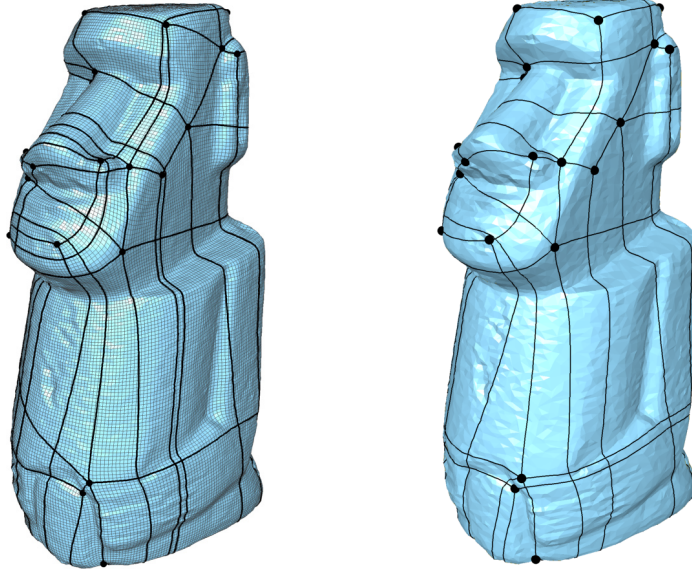


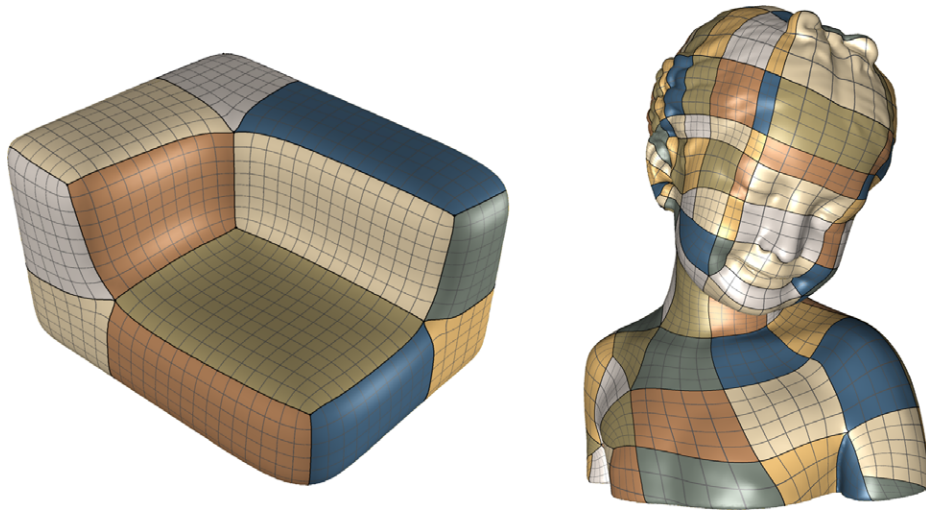
Figure 7.10: Comparison of a generated quad layout on quad meshes (left) with 136 patches and on triangle meshes (right) with 120 patches, with the same $\alpha = 2.5$.

Models	$ S $	Graph		Patches	Time in s	
		$ E_G $	$ I $		P_1	Total
Pegaso	28	528	7892	42	0.12	4.61
Bretzel	14	229	2162	24	0.06	1.73
Dragon	48	1118	18831	137	1.09	2.54
Botijo	52	1023	14150	85	0.46	1.91
RockerArm	74	1736	32192	163	0.64	5.77
DrillHole	30	609	9441	64	0.15	5.13
FanDisk	24	255	6085	79	0.03	0.61
Fertility	150	4194	122991	480	2.71	63.74
Bettle	70	1804	48888	255	1.93	55.51

Table 7.2: Statistics of the proposed method applied to several benchmark models. P_1 is the time taken to solve the constrained minimum weight perfect matching. The overall running time is from the construction of \mathcal{G} until the generation of the layout (without per patch colorings).

CHAPTER 8

REPARAMETERIZATION



There are several post processing algorithms which can be applied to the final layout. One attractive approach is the layout relaxation of Campen et al. [CK14] consisting of realigning the frame field to the arcs direction and relaxing the singularities to reduce distortions. Another approach is the patch parameterization of Tarini et al. [TPP⁺11] together with an iterative smoothing technique. Both methods still require a huge amount of work. Instead, we use a more direct approach which takes advantage of both Poisson parameterization and the computed patches.

Once the layout is generated, we would like to parameterize each patch such that the resulting integer-grid map is not far away from the already computed Poisson parameterization. First, we refine the mesh along all patch edges. This is done by clipping the polygonal curve against the surface. Next, we cluster each polygons in parameter space according to their 3D colorings. At this point, we could used meth-

ods similar to [MPZ14] with the iterative optimization suggested in [Lip12] but we would like to directly minimize the distortion as close to the initial parameterization as possible. A method such as [BCE⁺13] is then more suitable. However, we would have to add arc connection constraints and the per triangle trisector constraints which we avoid in our perfect matching based layouting.

8.1 PROBLEM FORMULATION

In parameter space, the 2D domains induced by the patches are quadrilateral patches with straight edges which are not necessarily aligned to the canonical axis. This is due to the balance α . If α is large, then these quadrilaterals would all be close to rectangular patches (minimizing deviation from the frame direction). We realign the patch edges to the axis directions in parameter space, assign integer lengths to these edges in order to preserve the continuity of the parameter lines between neighboring patches. The problem can be formulated as a mixed integer problem which is solved efficiently since most edge lengths are determined by the continuity constraints. Once a length is assigned to an edge, the opposite edge on the same patch is set to have the same length. This is recursively applied to the sequence of opposite edges until the starting edge is reached.

Denote by $(e_i, l_i)_{i \in I}$ the set of edges of the resulting quad layout \mathcal{Q} where l_i is the length of e_i measured in parameter space. We determine the closest integer lengths $(\tilde{l}_i)_{i \in I}$ by solving the following minimization problem.

$$\min \sum_{i \in I} |l_i - \lambda \tilde{l}_i|^2 \quad (8.1)$$

$$\text{s.t.} \quad \tilde{l}_i = \tilde{l}_j, \quad \text{if } e_i \text{ is opposite to } e_j \quad (8.2)$$

$$\tilde{l}_i \in \mathbb{N}^* \quad (8.3)$$

where λ is a constant which allows a control over the final grid size. This minimization problem has a solution $\tilde{l}_i = 1$ by forcing $l_i = 1$ for all i , which means ignoring or individual lengths and aiming for a coarse base mesh suited for subdivision surfaces. Otherwise, the resulting parameterization is a quasi-conformal parameterization with as uniform grid line as possible in each patch.

Unfortunately, our approach depends on the initial layout. If a patch strip contains at the same time an edge with a small length and another edge with a large length,

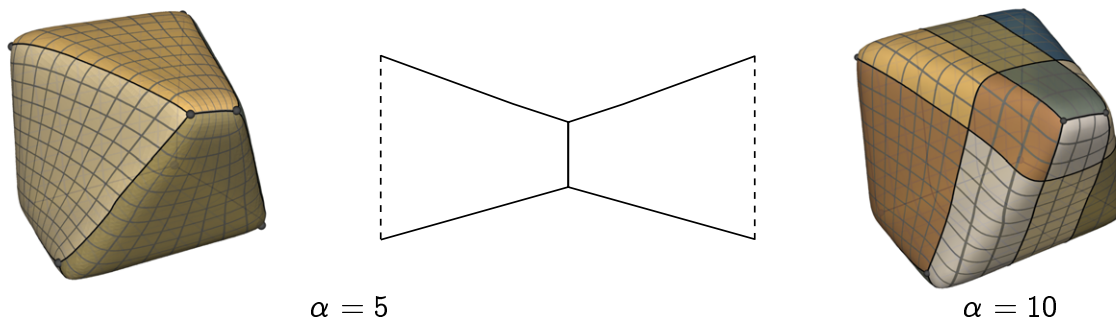


Figure 8.1: Grid compression along a patch stripe which has big edge lengths distribution can be solved by increasing the value of α .

then the grid discretization cannot be optimal for both edges (see Figure 8.1). Moving the singularity positions as done in [CK14] or increasing the value of α at the cost of increased number of patches can solve this issue.

8.2 MAPPING THE QUADS

After assigning integer lengths to each patch edges, the next step is to map each 2D quadrilateral patch to a rectangle with the associated integer lengths. We do it by a simple Wachspress mapping which maps the corners of the source quadrilaterals to a target rectangle having the computed integer lengths and interpolate the inner surface using Wachspress coordinates [Wac75], see Figure 8.2. This is a bijective mapping and hence the resulting parameterization is still flip free. Unfortunately, using a per patch parameterization, we only obtain a zero order continuity between neighboring patches. Since the initial patches induced by the Poisson parameterization are already close to being rectangular, the first order discontinuity is not really dominating in most cases. This is only visible when the deviation from the input field is big. A more elaborate mapping which also considers first order continuity between neighboring patches would be a nice feature to have. For that the quasiconformal maps introduced in [WMZ12] is a good starting point.

8.3 EXAMPLES

For all models shown in Figure 8.3, we choose $\lambda = 1$. We use Ilog CPLEX to solve the integer program. These models have been generated by the authors of [MPZ14] as Poisson parameterizations. These are taken as input to our algorithm and con-

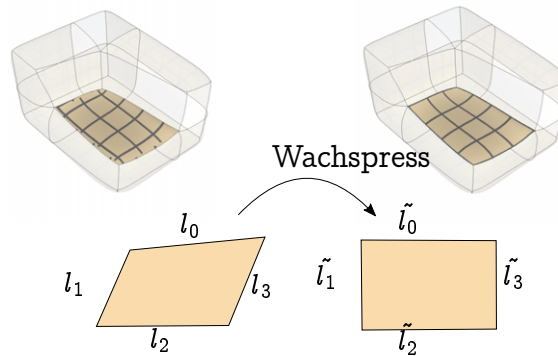


Figure 8.2: Mapping of the 2D domains for a given prescribed lengths using a simple Wachspress coordinates.

tinuously reparameterized.



Figure 8.3: Perfect matching quad layouts with globally continuous per patch parameterizations on models provided by the authors of [MPZ14].

CHAPTER 9

REGULAR MAPS

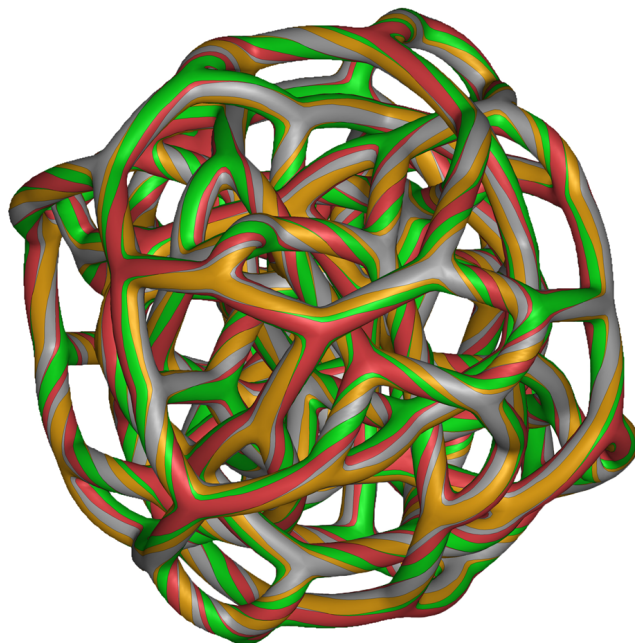


Figure 9.1: A regular map of genus 61 appearing on the cover of the Bridges conference proceedings in 2014.

The concept of map was first introduced by Coxeter and Moser [CM80]. A *map* is a family of equivalent polygonal faces such that any two faces share an edge or vertex, or are disconnected. Each edge of the map belongs precisely to two faces, the faces containing a given vertex form a single cycle of adjacent faces and between any two faces is a chain of adjacent faces. In other words, it is a closed 2-manifold

without boundaries obtained by glueing topologically equivalent polygonal faces. If the map has p -gonal faces and q -gonal vertex-figures (number of faces around a vertex), then it has the Schläfi symbol $\{p, q\}$.

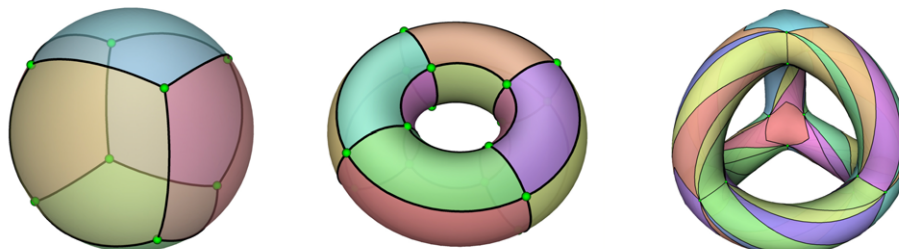


Figure 9.2: Examples of regular maps from left to right: a spherical cube, a $\{4, 4\}$ tiling of a torus and the Klein Quartic.

A regular map is a map which is flag transitive. On the surface, if a vertex or an edge or a face is mapped to another vertex or an edge or a face, then the map is one to one and preserves all adjacency properties. They can be viewed as generalizations of the Platonic solids to higher genus surfaces. They define regular tilings of closed surfaces and their group structure can be used to move along the surface behaving like an "hyperbolic" parameterization. In Figure 9.2 are some examples of low genus regular maps which include the cube, a checker-board tiling of a torus and the Klein Quartic.

The visualization of regular maps is a very challenging problem. Until 2009, only a handful regular map has been visualized. Then van Wijk [vW09] came with an heuristic which finds matching between regular maps. His approach derives space models of regular maps as the tubular neighborhood of the edge graph of other regular maps. These are given as pairs consisting of source and target regular maps. He was able to realize 50 cases of regular maps up to genus 29 using a group theoretical heuristic to find the pairings. Unfortunately, his method requires an embedding of the target map such that most of these pairings could not be visualized since some of the target regular maps do not have a known realization. Our main contribution in this thesis is the design of an algorithm which overcomes this limitation by generating directly the target regular map from the hyperbolic space. In this process, some important quality measurement should be met.

Quality Measures A regular map is a surface, hence a 3D realization of a regular map should have a manifold structure in order to achieve pleasing computer

rendered models. The transitivity properties of the tiling should be kept as well as all adjacency relations between individual faces. They should exhibit maximal symmetry preferably geometrical symmetry as well as topological symmetry. These include: geometrically equivalent faces (i.e. there exists a sequence of Euclidean motions which maps a face to another face), a subset of geometrically equivalent faces, and meaningful petrie polygon shapes (a closed zig-zag path on the edges). In Figure 9.3 is an example of a realization of a regular map where it is first isometrically drawn in the the hyperbolic space and realized as a closed genus five surface where all faces are geometrically equivalent. It is flag transitive and has 24 quads such that six of them always share a vertex.

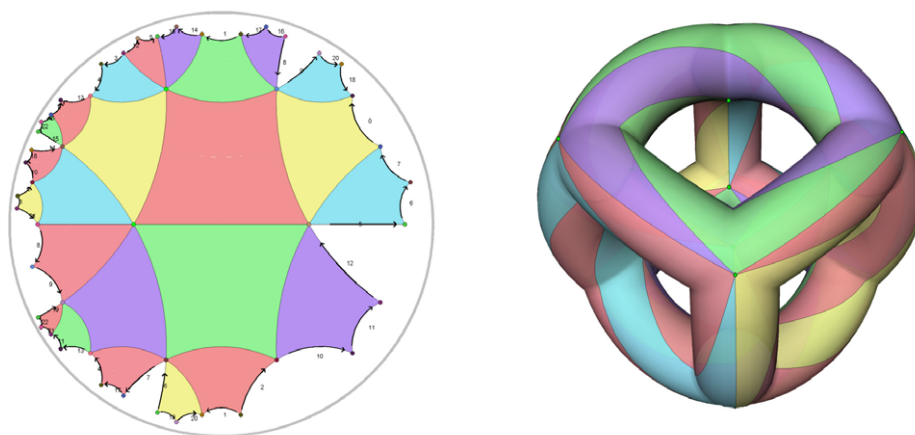


Figure 9.3: A $\{4,6\}$ tiling mapped on a genus 5 surface where all faces are geometrically equivalent.

The Chapter is organized as follows. First, we give theoretical backgrounds on regular maps including geometric and algebraic characterizations. Second, we describe the basic procedure to obtain, with maximal symmetry, large genus surfaces. And finally, we give a description of the method used to realize regular maps on these large genus surfaces.

9.1 BACKGROUND NOTIONS

The realization of a tiling depends on the ambient space where they are embedded. These are: the Sphere, the Euclidean plane and the hyperbolic plane. Examples of spherical isometric tilings are the Platonic solids. The Euclidean plane can be isometrically tiled by checker-board patterns and Honey-comb like structure. The

hyperbolic plane are tiled by p -gons for large p 's.

A closed 3D realization of a sub-tiling of the Euclidean plane is a torus (see Figure 9.2). The closeness of the torus is topologically derived from a parallelogram in the Euclidean plane wrapped in 3D by identifying opposite sides. In this 3D realization, isometry is lost but the topology of the tiling is still preserved. We then only talk about *combinatorial* or *topological* transitivity. Similar 3D realizations can also be done from the hyperbolic plane. A genus $g > 1$ surface is derived by taking a $4g$ -gon in the hyperbolic plane and identifying pairwise edges. Hence, any sub-tiling of the hyperbolic plane can be realized as 3D surfaces by finding a proper $4g$ -gon partition of this tiling with the correct identification at the boundary.

If a map has V vertices, E edges and F faces, then its genus g is given by $g = (2 - \chi)/2$ where $\chi = V - E + F$ is the Euler-Poincaré characteristic. This is a property of the surface, independent of the map; the dual map has also the same Euler-Poincaré characteristic χ . Intuitively, the genus of a surface is the number of tunnel or handle in this surface. Depending on their genus, regular maps can be abstractly realized as quotients of spherical tilings, Euclidean tilings or hyperbolic tilings.

Definition 9.1. A *finitely generated group* is a group of the form $\langle \mathcal{G} \mid \mathcal{R} \rangle$, where \mathcal{G} is a set of generators and \mathcal{R} is a set of relations. If $R_i \in \mathcal{R}$, then $R_i = I$ which is the identity of the group.

Definition 9.2. A regular map is a finitely generated group of the following form

$$\text{Sym}(M_S) = \langle R, S, T \mid R^p, S^q, T^2, (RS)^2, (ST)^2, (RT)^2, \mathcal{R}_1, \dots, \mathcal{R}_m \rangle, \quad (9.1)$$

where R is a rotation by $2\pi/q$, S is a rotation by $2\pi/p$ and T is a reflection.

R , S and T are transformations acting on a fundamental triangle with corner angles π/p , π/q and $\pi/2$ (see Figure 9.4). Depending on p and q , they can be Euclidean motions, special orthogonal matrices (for spherical) or Moebius transformations (for hyperbolic). $\mathcal{R}_1, \dots, \mathcal{R}_m$ are extra relations making the group finite. The expression 9.1 is called the symmetry group of the map. It is the set of all automorphisms of the regular maps as topological surfaces [CM80].

The symmetry group of the cube (a regular map of type $\{4, 3\}$) is for example defined by

$$\text{Sym}(\text{Cube}) = \langle R, S, T \mid R^4, S^3, T^2, (RS)^2, (ST)^2, (RT)^2 \rangle. \quad (9.2)$$

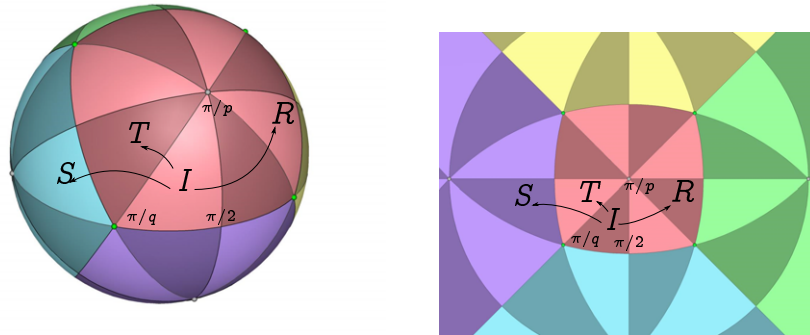


Figure 9.4: Two different representations of the cube using a Sphere and stereographic projection into the plane.

It can be realized on a blown up cube, taking as fundamental triangle a spherical triangle with corner angles $\pi/4, \pi/3$ and $\pi/2$. It can also be visualized as a 2D surface using stereographic projection which is only conformal but not isometric as depicted in Figure 9.4.

Orientable regular maps are denoted in Conder [CD01] by $Rg.i\{p, q\}$, which is the i th-reflexible orientable map of genus g . $\{p, q\}$ is the Schläfi symbol of the tiling. Reflexible means that the transformation T in Equation 9.1 is also an automorphism of the map. Analogously, the dual map is represented by $Rg.i'\{q, p\}$. Conder listed all reflexible regular maps of genus 2 to 302. They are given as symmetry groups and used as input to our algorithm.

9.2 GENERATING LARGE GENUS SURFACES

In this section, we explore in depth techniques to generate and visualize large genus surfaces. Our aim is not only to generate some genus g surfaces but also surfaces with rich topological structures and nice looking shapes.

Tubification Process A genus g surface can be generated by taking a sphere and drilling non intersecting g tunnels on it. Another approach, very useful for teaching, is the sphere with g handles. It consists of taking tori and glueing them on a sphere to form handles. It is unclear where the tori should be placed and if the resulting surface can be used to visualize symmetric tilings.

A better approach is the use of a *tubification* process or *regular surface*. The

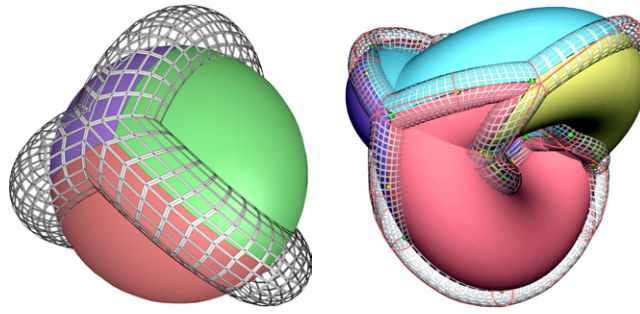


Figure 9.5: Regular surfaces (transparent).

process consists of taking a tiling of a surface, turning its edges into tubes, its vertices into junctions and its faces into holes. For example, a genus 2 surface can be derived from a tubified Hosohedron $\{2, 3\}$ as illustrated in Figure 9.5. Surfaces with rich structures or regular surfaces can be derived by taking regular tilings having more than two vertices. The Platonic solids are direct examples of these. Even more, we can take any regular maps and apply the tubification process to derive large genus surfaces. In Figure 9.5, we show an example of a tubification of the regular map $R2.8'\{8, 3\}$ giving a genus 9 surface.

As in [vW09], a pairing of source and target map is used to generate higher genus surfaces. The source map is the actual regular map that we want to realize and the target map is the regular map which after tubification gives a space model for the source map. Let $(M_i)_{i \in I}$ be a finite sequence of regular maps such that a space model of M_{i+1} is the tubification of M_i . For a given n , if for all $i < n$, the M_i 's are realized, then a tubification of M_{n-1} is a space model of M_n . Otherwise, we cannot give a space model for M_n . This becomes now a classical method used to visualize successfully large class of regular maps. In the next section, we show that, in fact, the sequence of M_i 's is not needed, only the pairing of source-target map is enough.

Targetless tubification The tubification of an existing regular map needs an actual realization of the target regular map. If the target regular map does not have a 3D embedding, then the tubification cannot be applied and thus no regular surface can be generated.

We give a solution to this restriction by taking advantage of the planar representation of the target regular map. We call this process a *targetless tubification*. Targetless in the sense that no actual 3D embedding of the target regular map is needed but only an embedding of its edge graph is sufficient. In Figure 9.6 is an

illustration of the overall process in comparison to the torus case.

We generalize the construction on the torus to hyperbolic space. Suppose we have a tiling of a flat torus with its 2D edge graph. This edge graph can be mapped to 3D using the usual torus parameterization and hence a tubular surface is derived naturally by the induced normals. In this process, only the edge graph is needed to be embedded in space, not the 2D tiling. We apply the same process in hyperbolic space which is done as follows. We identify explicitly boundary edges of the map followed by the constrained relaxation described in [Raz12]. An example of a genus 5 surface obtained by the edge graph of the regular map $R2.4\{5, 10\}$ is illustrated in Figure 9.6. In this illustration, we start with the hyperbolic realization of $R2.4$ together with the identifications at the boundary (represented by the arrows). We then match the boundary edges having the same label, head to head and tail to tail. This results in a 2D connected graph which has the same combinatorics as the edge graph of the underlying regular map. This 2D graph is smoothed using a simple spring energy following Hook's law. Notice that no actual embedding of the regular map $R2.4$ is needed (see [S10] for a 3D realization of this map). Our technique can be applied to any planar representation of a regular map to generate a 3D tubular surface obtained from its edge graph.

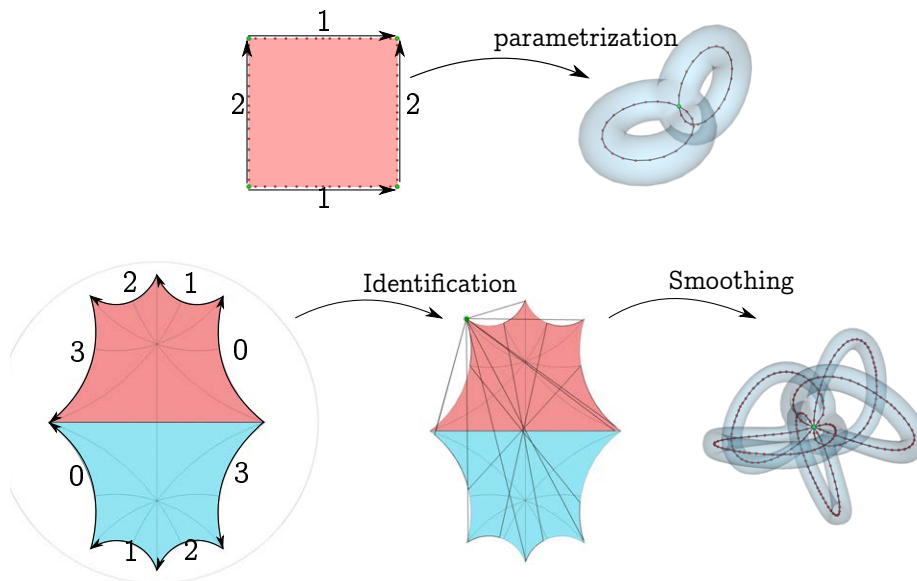


Figure 9.6: Construction of a high genus surface by embedding directly the edge graph of the target regular map.

9.3 IDENTIFICATION ALGORITHM

In this section, we give a detail description of the identification algorithm. Given the symmetry group of the target regular map, we give an automatic algorithm to obtain the boundary edges to be identified. We first give a topological description of the required settings and then the algorithmic construction.

Topological Representation Let G_{pq} be a regular map of genus ≥ 2 whose symmetry group is given by $\text{Sym}(G_{pq})$. This group can be realized as a triangular tiling in the hyperbolic plane, obtained from a triangle $M_0N_0O_0$ with corner angles $(\pi/p, \pi/q, \pi/2)$ setting R to be the rotation of angle $2\pi/p$ around M_0 , S the rotation of angle $2\pi/q$ around N_0 and T the reflection across the edge M_0N_0 (see Figure 9.7). We denote this tiling by $\text{Tri}(G_{pq})$. Each triangle of $\text{Tri}(G_{pq})$ is a geometrical representation of an element of $\text{Sym}(G_{pq})$.

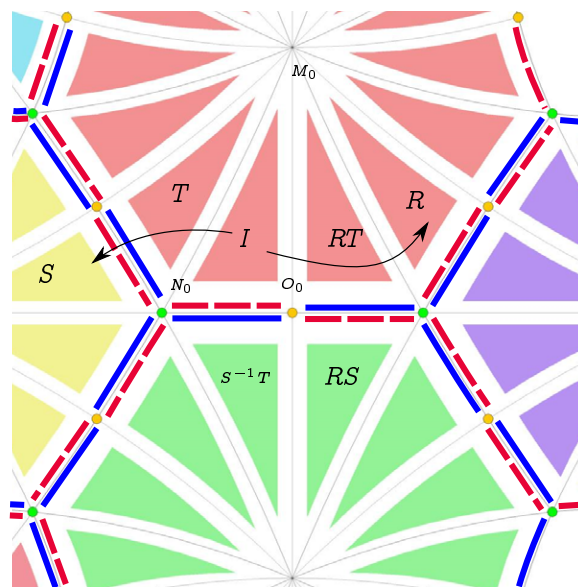


Figure 9.7: $\text{Tri}(G_{pq})$ and $\text{Line}(G_{pq})$.

To make $\text{Tri}(G_{pq})$ a topological closed surface, we need to make correct identifications for the triangles at the boundary. This is similar to the representation of a torus as a rectangle in the Euclidean plane with identified opposite edges (see Figure 9.6). The correct identification at the boundary is obtained by finding the missing neighbour of each triangle of $\text{Tri}(G_{pq})$ in $\text{Sym}(G_{pq})$ using the coset table of the group. For example, in Figure 9.8(a), triangle T does not have a neighbour

in $\text{Tri}(G_{pq})$ which in this case should be S . The relations of $\text{Sym}(G_{pq})$ imply that S and $S^{-1}R^3$ lie in the same equivalence class. Since a triangle corresponding to $S^{-1}R^3$ exists in $\text{Tri}(G_{pq})$, we identify their NO edges (blue lines). We proceed in the same way for all boundary triangles to obtain finally a planar realization of the regular map. Figure 9.8.b is an example of a topological representation of the regular map $R2.4\{5, 10\}$ preserving flag transitivity.

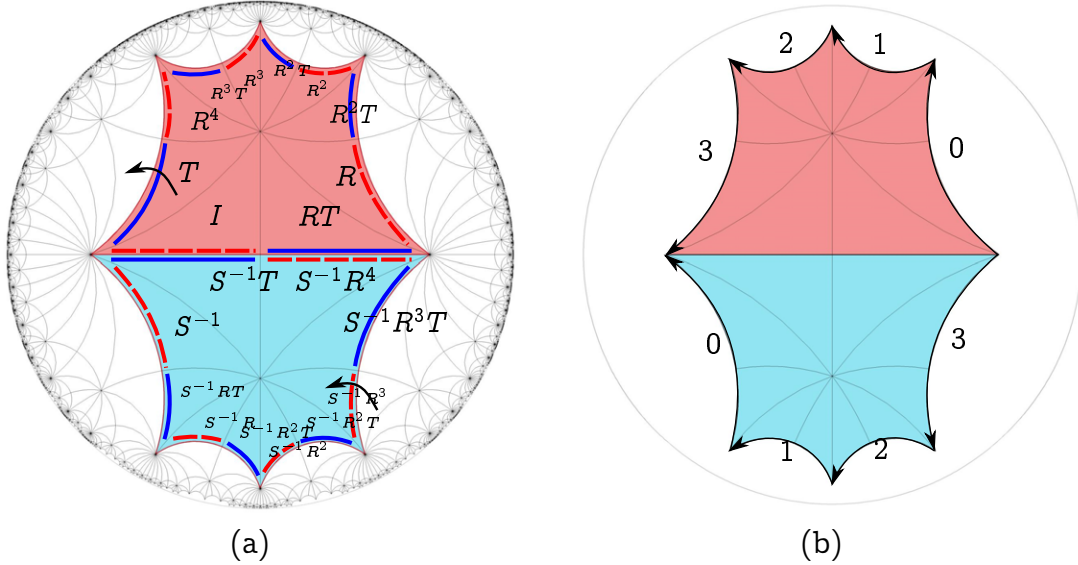


Figure 9.8: Representation of $\text{Sym}(R2.4)$ in the hyperbolic space: (a) without identification and (b) with correct identifications.

Let Δ_i be a triangle of $\text{Tri}(G_{pq})$ with corner vertex $M_i N_i O_i$. We denote by $\text{Line}(G_{pq})$ the set of all $N_i O_i$ segments of $\text{Tri}(G_{pq})$ which is again a geometrical realization of $\text{Sym}(G_{pq})$. This group can be viewed as a “double covering” of the medial axis of the associated regular surface. It is represented by red dashed segments and blue segments in Figure 9.8. In the topological representation, the boundary arrows show where a triangle or a NO segment should be glued. We can use this information to make sure that an element $L \in \text{Line}(G_{pq})$ has the same geometrical position as $LS^{-1}T$. Once all the L and $LS^{-1}T$ have the same geometrical position, we can generate the regular surface by associating to each NO segment a half-tube extending halfway to the middle of one of the tubular edges similar to [vW09].

Identification The goal of this Section is to give the elements L and $LS^{-1}T \in \text{Line}(G_{pq})$ (red dotted and blue segments on the boundary) the same geometrical position. We do this by using the identifications induced from $\text{Tri}(G_{pq})$. As stated

in the previous section, only the segments of $\text{Line}(G_{pq})$ on the boundary triangle of $\text{Tri}(G_{pq})$ need to be moved. We could also try to do the identification directly on the edges of $\text{Tri}(G_{pq})$ but it is hard if not impossible to resolve the intersecting faces of the triangles after the identification. The identification is combinatorial and to obtain a reasonable 3D initial configuration of the graph, we choose the Jemisphere model of hyperbolic space. The Jemisphere model is a hemisphere like model of hyperbolic space which is also conformal. Infinity is represented by the equator (Figure 9.9).

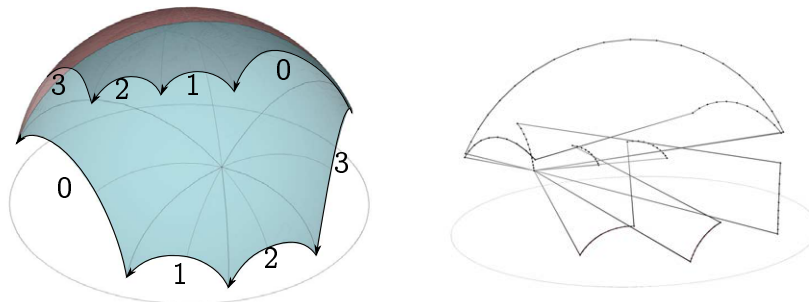


Figure 9.9: $R2.4$ projected on the Jemisphere model (left) and the resulting identification applied to $\text{Line}(R2.4)$ on this model (right).

Suppose that we identify a directed edge a with a directed edge b . Define the tail of a to be t_a and t_b the one of b . Find all segments of $\text{Line}(G_{pq})$ having t_a as endpoint and move these vertices to the tail t_b of b . In this process, only the endpoints of a are moved. We do the same for the head h_a of a to the head h_b of b . Once both head and tail are matched, we update the geometrical position of a to be the one of b . We iterate this process until all the boundary NO segments of $\text{Line}(G_{pq})$ are identified. Figure 9.10 is an illustration of the overall process applied to $\text{Line}(R2.4)$. It is only a conceptual illustration to show how the identification is done. In state 1, only four elements of $\text{Line}(G_{pq})$ are pairwise at the the same geometrical positions (dotted segments are adjacent to continuous segments). The rest need to be moved according to the glueing procedure defined previously. The orange dots represent the edges to be identified. The red dots represent edges to be removed or updated. The algorithm starts by identifying edge 0, first head to head and then tail to tail. Once these are done (state 3), the red dotted line is updated to where it is identified. Then, it continues with edges 1 and 2 with the same strategy. At state 10, there is only one step to be done since the curve 3 is a loop.

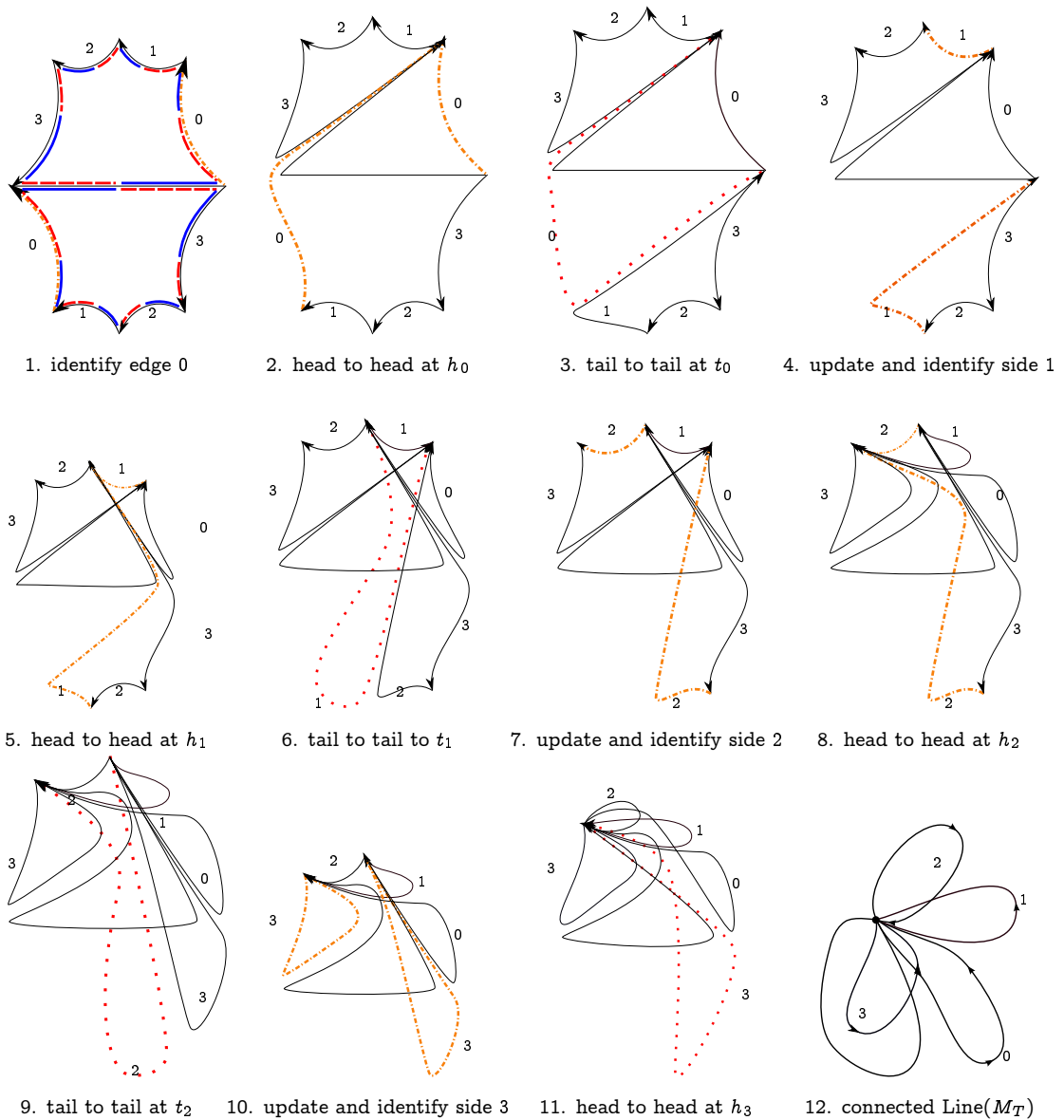


Figure 9.10: The process of identifying all elements $L \in \text{Line}(M_T)$ to have the same geometrical positions as $LS^{-1}T$ and vice versa. On the final shape, each red and blue line in state 1 are pairwise identified.

The input of the algorithm is the set of segments $\text{Line}(G_{pq})$ and the output is again $\text{Line}(G_{pq})$ but the elements L and $LS^{-1}T$ have the same geometric positions.

The identification does not depend on the order of the glueing. The resulting curves

and loops are the same. The only problem happens at the end of the identification where the ordering of the segments at the nodes is not well-defined. If we try to derive the regular surface, it might result in a degenerate tubular surface (see Figure 9.13). We solve this ambiguity in the next section.

9.4 REGULAR SURFACES

Construction The regular surface obtained by the tubification of the edge graph of a regular map G_{pq} is also a geometric realization of $\text{Sym}(G_{pq})$. It is derived by associating to each element $L \in \text{Line}(G_{pq})$ a quarter-tube Q_L . The folding of the quarter-tubes depends on the corresponding element in $\text{Sym}(G_{pq})$. If the element contains T (a continuous segment in $\text{Line}(G_{pq})$), then it is folded left. Otherwise, it is folded right. Here, left and right are defined according to the normals from O to N . The geometric construction of the quarter-tubes is illustrated in Figure 9.11. The normals along each segment are interpolated from the junctions.

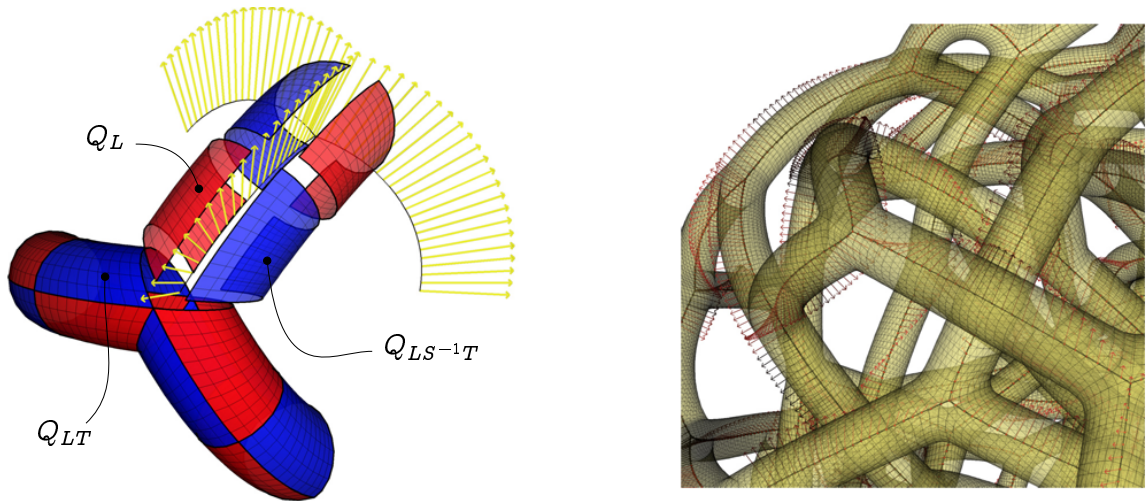


Figure 9.11: Construction of the quarter-tube group $\text{Tub}(G_{pq})$ using normals defined by the local neighborhood at the junctions vertices.

We call the resulting group $\text{Tub}(G_{pq})$. For all $Q_L \in \text{Tub}(G_{pq})$, Q_L and $Q_{LS^{-1}T}$ form a half-tube. This is possible due to the identification done previously. To have correct tube junctions at the nodes of $\text{Line}(G_{pq})$, it is necessary to have, additionally, each quarter-tube Q_L adjacent to Q_{LT} . This is equivalent to having a local ordering of each segment of $\text{Line}(G_{pq})$ at the nodes.

The local ordering at each node is obtained from $\text{Sym}(G_{pq})$. Suppose that we are at

a node v which is contained in the segments $L_0, \dots, L_{2q-1} \in \text{Line}(M_T)$. The ordering of the segments, as in the hyperbolic space, is reconstructed in a well defined plane at v . If $L_0 = L$, then we must find a plane on which $L_1 = LT, L_2 = LS, \dots, L_{2q-2} = LS^{q-1}$, and $L_{2q-1} = LS^{q-1}T$. In our implementation, we use this information as a constraint and put $\text{Line}(G_{pq})$ in a spring relaxation procedure. An example of this relaxation procedure is shown in Figure 9.12 for the case of $\text{Line}(R2.4)$. The basic idea is to consider each vertex of the graph as a charge particle attracted by its neighbors and repulsed by the other vertices. This simple physical system gives symmetrical shape but can also get stuck in local minima. For more details on the implementation, we refer to [Raz12]. The resulting tubular surface is illustrated in Figure 9.13.c.

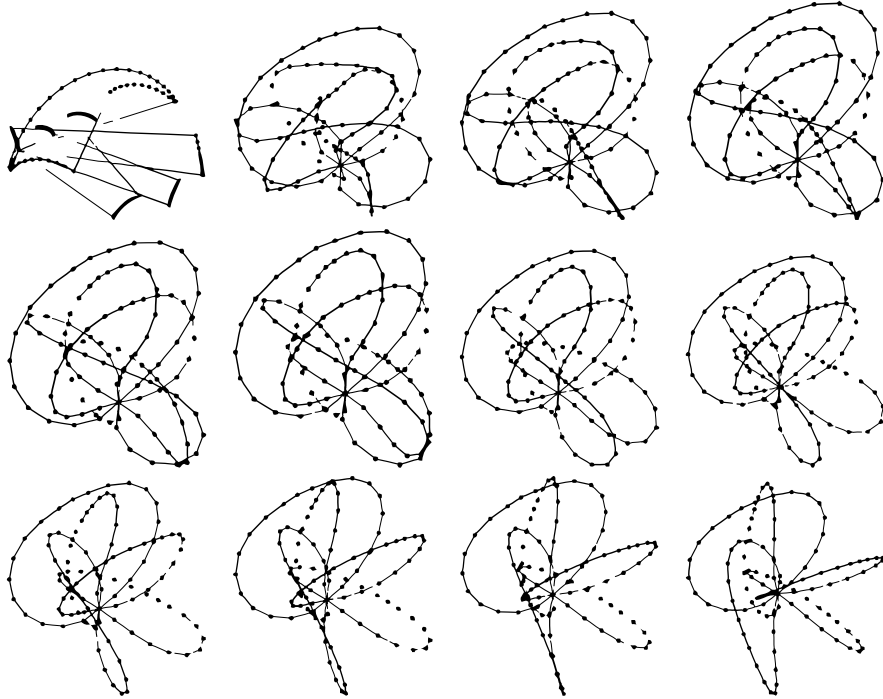
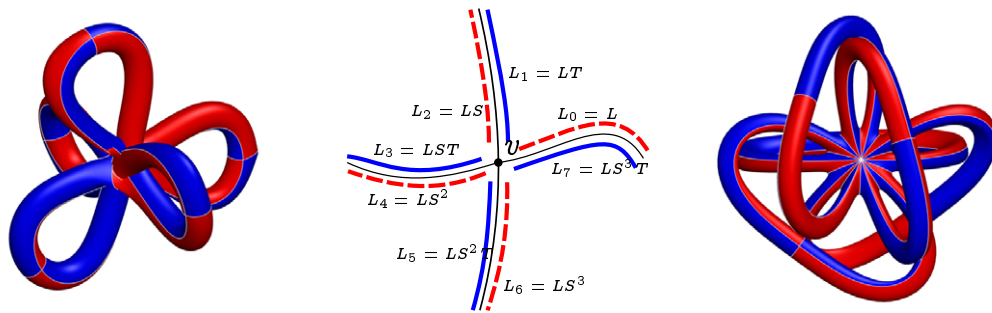


Figure 9.12: Spring relaxation of $\text{Line}(R2.4)$ with the plane constraint at the junction.

Figure 9.13.b is an example of such a plane where $q = 4$. Figure 9.13.a is an example of a minimum state where the spring relaxation is not constrained inducing a degenerate surface. The identification procedure does not unfortunately guarantee this orientation. Figure 9.13.c is an example of $\text{Line}(G_{pq})$ for which $\text{Tub}(G_{pq})$ does not have self-intersections with a well defined tangent plane at the node.



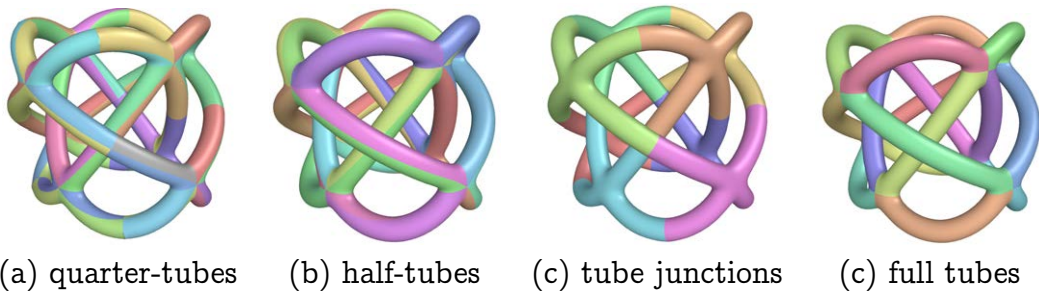
(a) Self intersection (b) Correct neighboring at a node (c) Correct tube junction

Figure 9.13: Wrong ordering at a junction may lead to a degeneracy of the regular surface. The correct ordering is found on a suitable tangent plane.

9.5 GROUP STRUCTURE

In this section, we define a topological group structure on the surface generated previously that we denote S_g .

Partition by Tube Elements The recursive tubification procedure derived in Section 9.2 allows us to choose a tiling of S_g . We have for example a tiling with quarter-tubes, with half-tubes, with tube junctions, with full tubes or with multiple quarter-tubes (see Figure 9.14). One piece of each tiling form a *fundamental domain* of S_g , and as for every group, we can cover the surface by copies of this fundamental domain. These tilings are induced naturally from the underlying regular map used to derive the tubular surface.



(a) quarter-tubes (b) half-tubes (c) tube junctions (d) full tubes

Figure 9.14: Example of a partition of S_g with some elements of the tubes.

The next step is now to define a group structure induced by the tube element in order to define a parameterization of S_g . This parameterization will be then used to map other regular maps as described in Section 9.5.

Deriving the Symmetry Group We restrict our construction to the case of a tiling with quarter-tubes as in [vW09]. The other cases can be handled analogously. Let \mathcal{Q} be the set containing all colored quarter-tubes of S_g . We label the edges of a quarter-tube by a, b, c and d , where a is the one at the junction and b, c, d are the next counter-clockwise edges as illustrated in Figure 9.15. The orientation is defined by the normals of the surface at each quarter-tube.

We define a basic operation Adj_x on \mathcal{Q} which takes a quarter-tube Q and returns the quarter-tube adjacent to Q at edge x :

$$\begin{aligned} \text{Adj}_x: \mathcal{Q} &\rightarrow \mathcal{Q} \\ Q &\mapsto \text{quartertube adjacent to } Q \text{ at edge } x \end{aligned}$$

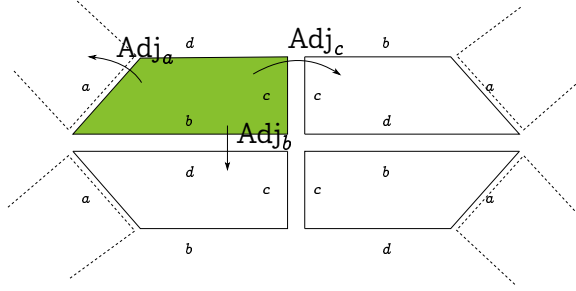


Figure 9.15: The adjacency operator defined on the set of quarter-tubes.

For example, $(\text{Adj}_b)^2$ is the identity since making two quarter-tube steps around a tube returns to the same starting quarter-tube. $(\text{Adj}_a)^2$ is also the identity. Let Q_I be a fundamental domain of \mathcal{Q} (it can be any quarter-tube of \mathcal{Q}). We define three operations A, B and C on \mathcal{Q} as follows:

- A shifts Q_I two positions positively around a hole, more precisely $Q_A = \text{Adj}_a(\text{Adj}_c(Q_I))$;
- B rotates Q_I around the junction, $Q_B = \text{Adj}_d(\text{Adj}_a(Q_I))$;
- C shifts Q_I one position down, $Q_C = \text{Adj}_b(Q_I)$.

Here, Q_M denotes the quarter-tube obtained by applying a transformation M to Q_I .

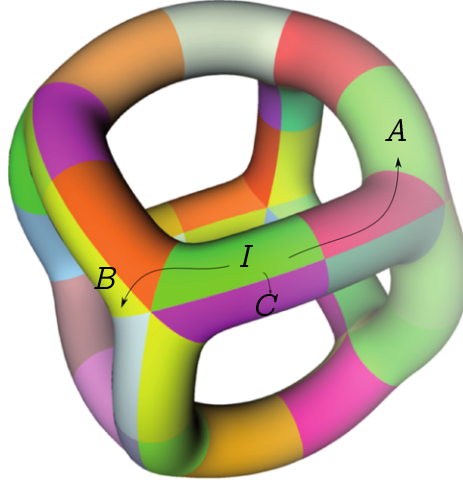


Figure 9.16: Three adjacency operators acting on the quarter-tube tiling: A shifts Q_I two positions positively around a hole; B rotates Q_I around the junction; C shifts Q_I one position down.

We can see A as a transformation moving a tube around a hole, B switches from one hole to another hole and C enables to reconstruct a full tube from a quarter of a tube. Using Adj_x , we can derive the following relation

$$(CBA)^2 = (BA)^2 = (CB)^2 = I$$

where, I denotes the identity transformation. Using the underlying symmetry group of the tiling used to build S_g we can define a symmetry group of S_g as

$$\text{Sym}(S_g) = \langle A, B, C \mid A^{p_t}, B^{q_t}, C^2, (CB)^2, (BA)^2, (CBA)^2, \\ g_1(A, B, CB), \dots, g_n(A, B, CB) \rangle,$$

where, the g_i 's are the extra relations of the symmetry group of the underlying regular map.

A group structure on the genus 5 surface shown in Figure 9.16 is given by

$$\text{Sym}(M_5) = \langle A, B, C \mid A^4, B^3, C^2, (BA)^2, (CB)^2, (CBA)^2 \rangle.$$

This group has exactly 12×4 quarter-tubes. Once the group structure is introduced on S_g , we can unfold this surface in hyperbolic space to embed a regular map on it.

Matching Symmetry Groups We give a brief description of the use of the symmetry group introduced on S_g to realize a regular map. The regular map is defined with its symmetry group $\text{Sym}(G_{pq})$ realized as a planar tiling in the hyperbolic space.

The first step of the algorithm is to make an hyperbolic parameterization of S_g . The parameterization is done by choosing a suitable fundamental quadrilateral hQ_I in hyperbolic space as a fundamental domain of $\text{Sym}(S_g)$. The idea is to make a 2D realization of $\text{Sym}(S_g)$ using another fundamental domain. Once the parameterization is done, the regular map can be naturally mapped using the inverse mapping. An overview of the algorithm is illustrated in Figure 9.17. The remaining problem is then on the construction of hQ_i and the hyperbolic transformations corresponding to the elements of $\text{Sym}(S_g)$.

The construction of hQ_I depends on the matching between $\text{Sym}(S_g)$ and $\text{Sym}(G_{pq})$. These matchings are heuristics which check if there exists a partition of $\text{Sym}(G_{pq})$ by $\text{Sym}(S_g)$. A necessary condition is that the order of $\text{Sym}(S_g)$ should divide the order of $\text{Sym}(G_{pq})$ and a sufficient condition is the existence of a subgroup of $\text{Sym}(G_{pq})$ isomorphic to $\text{Sym}(S_g)$. In case of success, hQ_I can be constructed, otherwise S_g is not a suitable space model for $\text{Sym}(G_{pq})$ and the mapping cannot be done. Matchings between regular maps are generated using van Wijk [vW09] heuristic. His heuristic also provides the exact position of the four points of hQ_I in hyperbolic space.

In his lists, there are several mappings which cannot be visualized since the target map does not have a 3D realization. This is mainly the case for the large genus regular maps which depend on several lower genus ones. These cases are handled by the targetless tubification in Section 9.2.

9.6 NEW REALIZATIONS

Unlike regular maps, regular surfaces can always be visualized. One weak point of van Wijk's approach [vW09] is that the embedding of a regular surface depends on the embedding of the underlying regular map. For example, the regular surface of the map R2.5{5, 10} can be used as a space model of the map R5.8{4, 20} but if R2.5 is not embedded, so is R5.8. Our algorithm handles this case and depends only

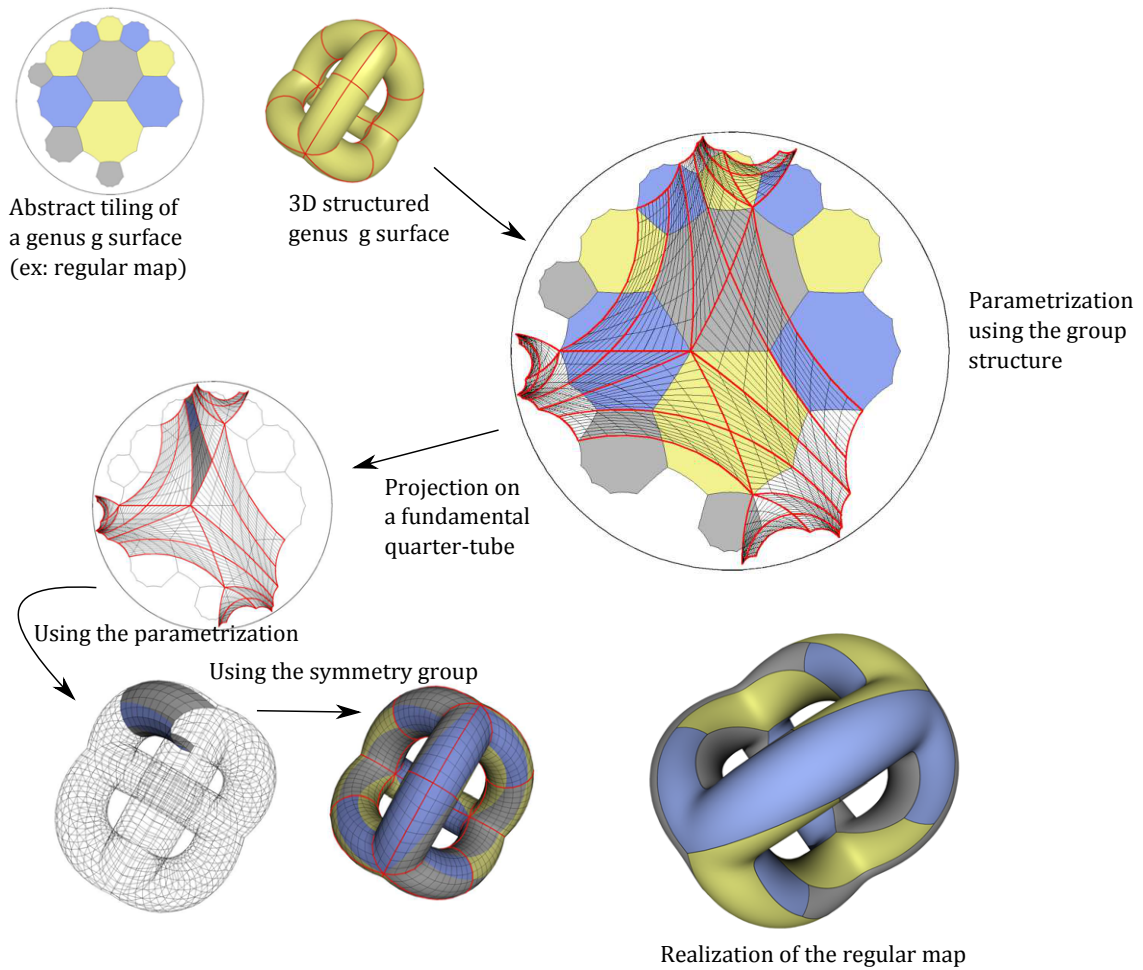


Figure 9.17: Pipeline to visualize a regular map on a structured genus g surface.

on the regular surface of the target map, not the target map itself. In Figure 9.18 is our embedding of R5.8 together with its dual.

We find new realizations of regular maps by combining van Wijk's heuristic [vW09] with our algorithm. For the purpose of visualization, we show regular maps for which q is not too large and regular surfaces whose number of junctions are not 2 (hohohedral surfaces). In all generated regular maps, we emphasize on the aesthetic visualization of the realizations in our most symmetric embeddings. Four realizations are our favorite embeddings so far, mapped on the edge graphs of double covered Platonic solids whose branch points are the centers of faces. They are: R9.3' $\{6, 4\}$ on a 2-Cube, the 6-rings or R13.1' $\{12, 3\}$ on a 2-Octahedron (this has also a nice relation with the Borromean rings [RP13]), R21.3' $\{6, 4\}$ on a 2-

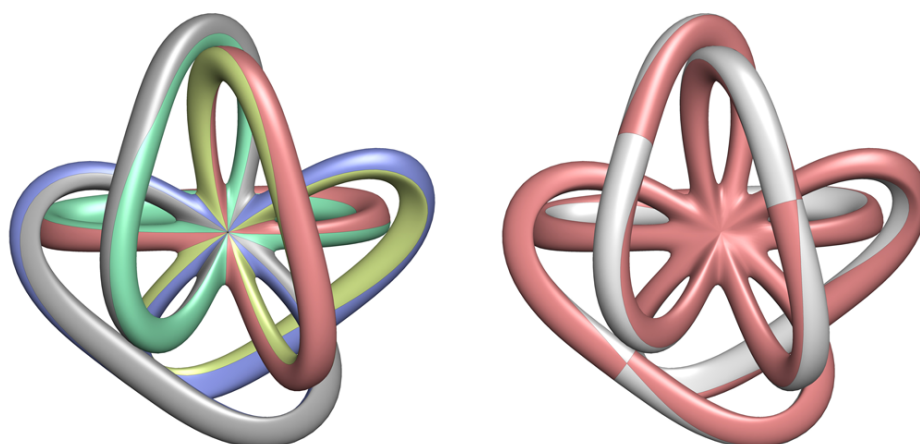


Figure 9.18: A realization of the regular map $R5.8\{4, 20\}$ (left) and its dual (right).

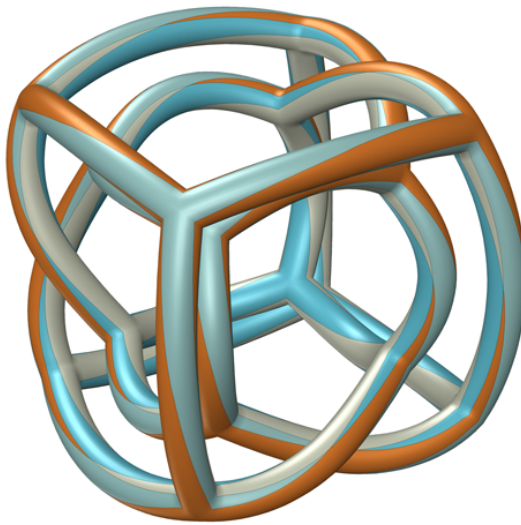
Dodecahedron and $R37.2'\{15, 3\}$ on a 2-Icosahedron, see Figure 9.19.

Unfortunately, due to the even-sided faces of the cube, we could only achieve one $C3$ axis and one $C2$ axis on the double-covered geometry. For the other Platonic solids, we only lost the non-oriented symmetry. The double covering of the Tetrahedron gives the cube, so we do not show any regular maps mapped on its edge graph since the edge graph of the cube is more symmetric.

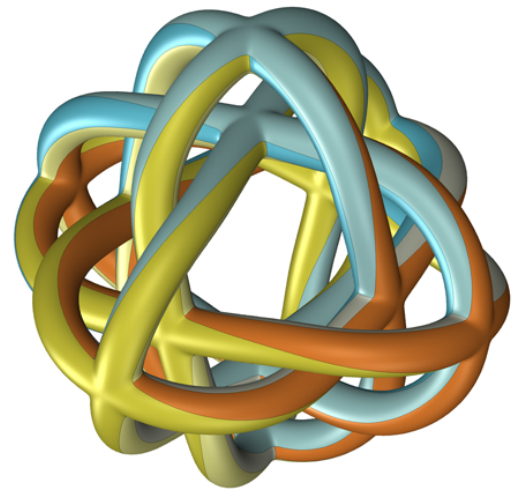
In Figure 9.20 are more examples of regular maps we succeed to embed. For the first three surfaces, we succeed in giving a rotational symmetry to the regular maps since the valence of the junctions is not too high except those which have only one junction. High valence junctions make the task of getting symmetry hard. Hence, for those classes of regular maps, we only took a minimum of the spring relaxation energy as the final shape.

In Figure 9.21 are three “really” large genus regular maps generated by our method. For these classes of regular maps, obtaining symmetry is a huge amount of work. Even a non self-intersecting surface is hard to obtain. However, we believe that a careful understanding of the tubes and tiles can improve the symmetry of the shape. $R85.8'$ is the regular map with the highest genus ever visualized so far (excluding the hosohedral type of surfaces) where previously it was only a genus 29 regular maps [vW09]. We asked ourselves how can one find an embedding even with one $C2$ axis of such a complex topological object?

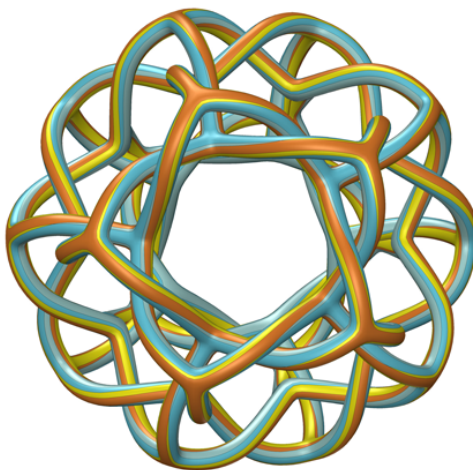
The algorithm does not have any limitation as long as the matchings between the regular maps are provided. Van Wijk’s heuristic gives correct matchings of hundreds



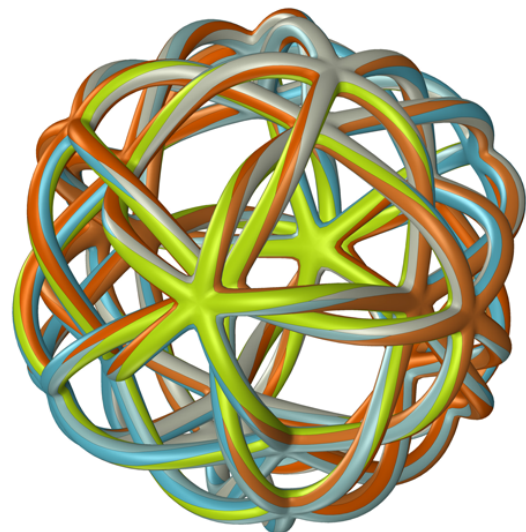
R9.3' on a 2-Cube



R13.1' on a 2-Octahedron



R21.3' on a 2-Dodecahedron



R37.2' on a 2-Icosahedron

Figure 9.19: High genus regular maps embedded on the regular surfaces of double covered Platonic solids.

of regular maps which can all now be visualized using our technique.

We showed how to generate an embedding of a regular surface implicitly from a regular map. The regular map itself does not have a priori an embedding. Hence,

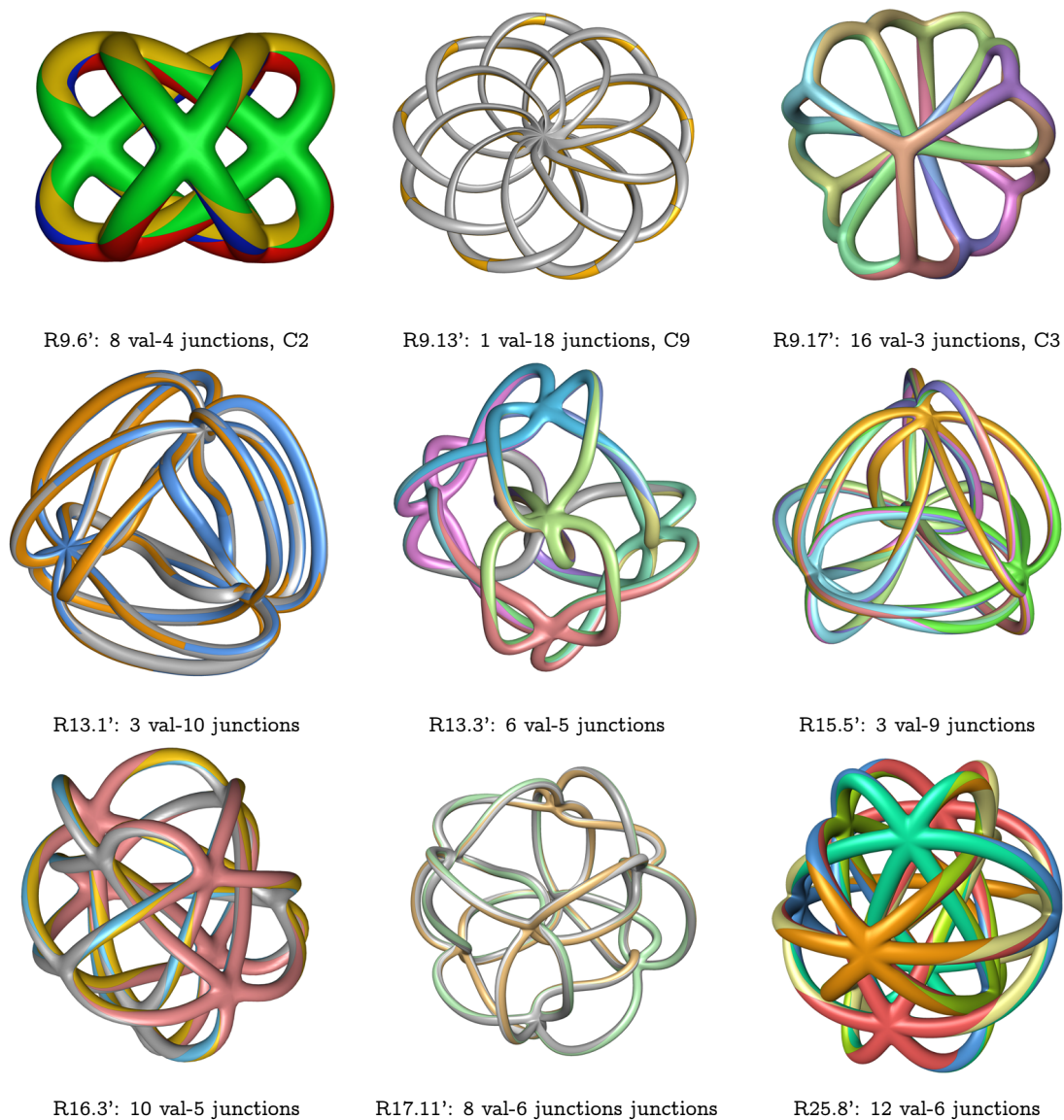
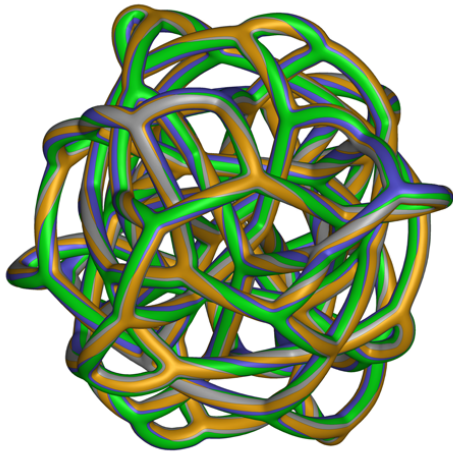
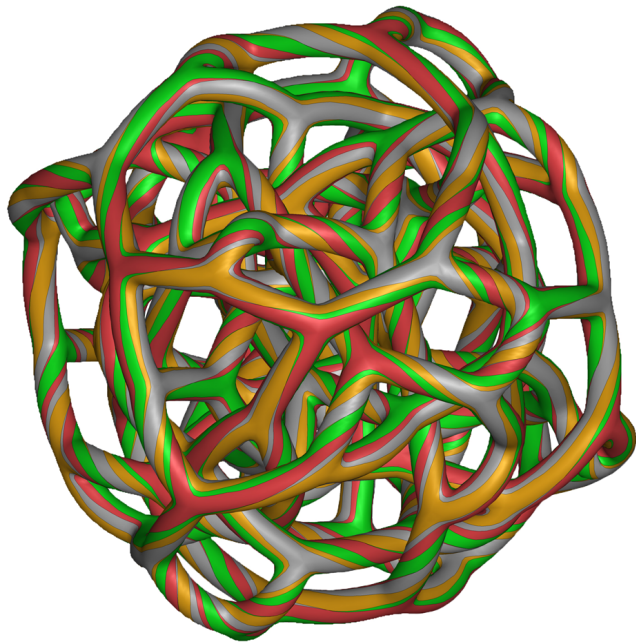


Figure 9.20: Some selected high genus regular maps generated by the algorithm. In the first row, junctions are placed manually on a target shape while for the rest, they are automatically generated.

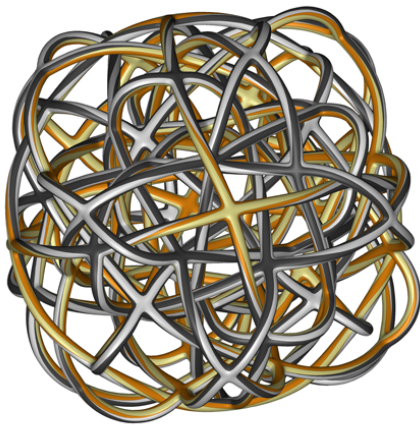
such maps (for example, R2.5) are not handled by our algorithm. Our algorithm also does not find regular maps for the Hurwitz surfaces. An interesting problem will be to find a similar procedure to realize directly regular maps.



R41.3': 80 val-3 junctions



R61.1': 120 val-3 junctions



R85.8': 84 val-4 junctions

Figure 9.21: Some large genus regular maps generated that are intersection free.

CHAPTER 10

EXTENSIONS AND FUTURE WORKS

This Chapter brings light to a conceptual extension of the perfect matching quad layout algorithm on T-meshes and volumetric meshes. We leave the implementation as well as practical evaluations as future works.

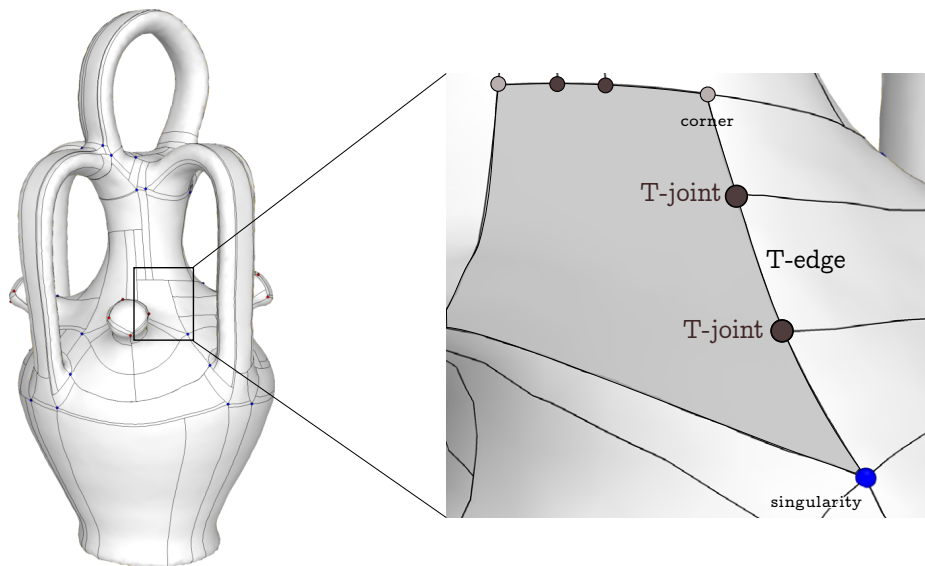


Figure 10.1: An example of a T-layout on the Botijo model where the grey patch is a quadrilateral face with four corners (grey and blues) bounded by T-edges.

10.1 T-LAYOUT TO QUAD LAYOUT

In our graph construction, we either stop a tracing line when it starts to spiral (on triangle mesh) or we stop when a singularity is reached (on quadrilateral mesh). In the later case, the set of isolines gives the base complex of the surface while in the first case we have constructed the so called *T-mesh* or in the spirit of coarse layout, a *T-layout* of \mathcal{M} induced by all stopped isolines.

Definition 10.1. A T-layout is a partition of \mathcal{M} into quadrilateral patches but patch edges are split into sub-edges called *T-edges* delimited by vertices called *T-joint*.

A quad layout is a particular type of T-layout where there is no T-joint. Following the notation of [MPKZ10], there are three types of node vertices on a T-layout: *T-joints*, *regular* and *singular* nodes. A *T-joint* is a valence three vertex where three quads meet at an edge. A vertex is called *regular* if it has valence four (or three on the boundary) and it is singular otherwise. A quad of a T-layout has four corners which are not T-joint with respect to the face and four edges which are split into T-edges (see Figure 10.1).

The graph construction presented in Chapter 4 can be directly adapted to work on T-layouts. By finding a consistent T-edge weights as proposed in [MPZ14], we can bypass the computation of a parameterization and work directly on the metric induced by these weights.

Definition 10.2 (Geometric Integrability). A T-layout is *integrable* if it can be realized as 2D periodic rectangular patches without degeneration. It is non-integrable otherwise.

An example of an integrable T-layout is the motorcycle graph obtain by the singularity ports of a Poisson parameterization. The term integrable is here inherited from the curl freeness of the induced gradient field. Non-integrable T-layouts are characterized by the existence of *limit cycles*. Those are layouts which cannot be realized as a 2D rectangular patch without collapsing some T-edges, e.g. in Figure 10.2.

Definition 10.2 is too tight. It already assumes a geometric realization of the layout which resumes to a global parameterization of \mathcal{M} . Nevertheless, we could ignore the realization and still work in a combinatorial manner on the layout graph as in the quadrilateral mesh case.

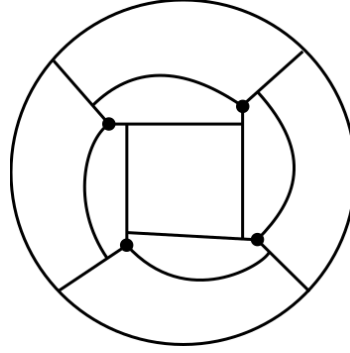


Figure 10.2: An example of a non-integrable T-mesh.

Definition 10.3 (Combinatorial Integrability). A T-layout is *integrable* if there exists a non trivial edge map $h : \mathbb{E}_{\mathcal{T}} \rightarrow \mathbb{R}_+^*$ such that the sum of the T-edge weights on one side of a patch is equal to the sum of the T-edge weight of the opposite side of the same patch.

In a more mathematical formalism, let $(e_i)_{i \in I}$ be the set of T-edges along an edge of a quad patch and $(e_j)_{j \in J}$, the opposite T-edges in the same patch. Then h satisfy the following equality,

$$\sum_{i \in I} h(e_i) = \sum_{j \in J} h(e_j). \quad (10.1)$$

Definition 10.2 and Definition 10.3 are equivalent. In the combinatorial representation, we do not need a geometric realization of the patches as a parameterization which is advantageous since we can directly construct \mathcal{G} on a purely combinatorial computation. Our construction can also be extended to non-integrable T-layouts allowing zero edge weights to be an indicator as a stopping criterion. Without loss of generality, we suppose that for a given T-joint, there exists a sequence of monotone T-edges, i.e. without turning left or right on the T-layout, connecting the T-joint to a singularity. We denote by d_h the distance on the T-edges induced by h .

Isoline Tracing Isoline tracings on T-layouts are much faster than on Poisson parameterized surfaces. In the later case, for each triangle of the surface, we always need to store all separatrices going through it which is mesh resolution dependent. On a T-layout we can save this information on the T-joints and the T-edges. More,

the T-layout can be given as input without the extra computation of a parameterization.

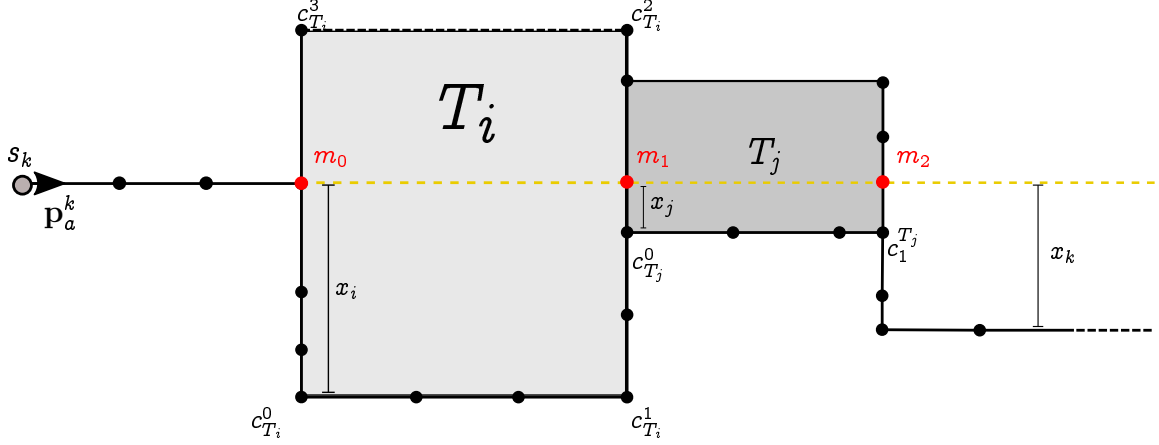


Figure 10.3: Isoline tracing on a T-layout.

Suppose we want to trace an isoline γ_{a+}^k on the T-layout. We start at a port p_a^k and follow the monotone T-edges until a T-face is intersected orthogonally at a T-joint m_0 . We denote by T_i the oriented T-face with corner vertices $c_{T_i}^0, c_{T_i}^1, c_{T_i}^2, c_{T_i}^3$ which contains m_0 . In this notation, $c_{T_i}^0$ is always the first corner on the right of m_0 in T_i . We denote by x_i the distance of m_0 to $c_{T_i}^0$ using the metric induced by h . Then γ_{a+}^0 leaves T_i at a point m_1 with distance x_i from $c_{T_i}^1$ at a T-edge opposite to m_0 . We compute the new distance x_j of m_1 with respect to the adjacent T-face T_j containing m_1 . It is computed by taking the difference of x_i with $d_h(c_{T_i}^1, c_{T_j}^0)$.

$$x_j = x_i - d_h(c_{T_i}^1, c_{T_j}^0)$$

We proceed similarly for the next T-faces until a stopping criterion or a maximum tracing length is reached.

Given a point $m = \gamma_{a+}^k(t_m)$ such that m is on a T-edge of a T-face generated by the previous isoline tracing approach, the distance of m to the singularity s_i is defined as

$$L_a^i(m) = d_h(s_k, m_0) + \sum_{k=0}^n d_h(c_{T_k}^0, c_{T_k}^1).$$

The point m_0 is the T-joint where γ_{a+}^i intersects orthogonally a T-face for the first time. Here, n is the number of T-faces previously visited by γ_{a+}^i such that $m \in T_n$.

Graph construction The construction of \mathcal{G} is similar to the quad mesh case. The collection of the possible ports connection is done in three parts. First, at each T-joint, we evaluate the ratio of the two intersecting separatrices. The corresponding edges are then added temporarily to \mathcal{G} . Second, we do a discrete tracing from each T-joint and evaluate ratio of separatrices whenever a tracing intersect a T-edge of a T-face. Finally, we evaluate the inter-relationship of all traced lines at a T-face and deduce the edges of \mathcal{G} accordingly. To avoid non singularity-free triangles in the final graph, we post process all collected candidate ports of a port by sorting them along their base length and applying the window reduction.

10.2 HEXAHEDRAL MESH SIMPLIFICATION

Hexahedral meshes or cube meshes are 3D volumes made from cubes glued at their faces and aligned smoothly to the boundary surface. In contrast to quadrilateral meshes, the generation of hexahedral layouts are far more complicated. On these meshes, there are two types of singularities. Most of them are edge singularities i.e. edges whose number of adjacent cubes is not four and few of them are vertices. Vertex singularities or *node points* are inner points where singular edges

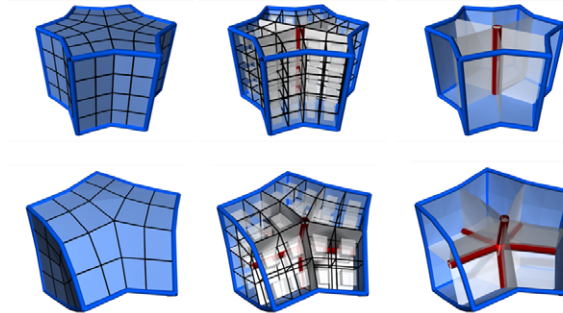


Figure 10.4: Examples of edge singularities in a hexahedral mesh.

ends. Deriving base complexes for hexahedral meshes is achieved by tracing all the separation surfaces emanating from the singular edges. Isolines become then *isosurfaces*. The ports become *surface directions* and the separatrices are again surfaces or *separation surfaces*. In Figure 10.4 are examples of singularities and ports initialization on hexahedral meshes.

The graph construction algorithm applied to the isosurfaces cannot be extended naturally. It only works for the trivial extruded 2D case, i.e. the base complex does not contain node singularities and the isosurfaces do not contain any singularities,

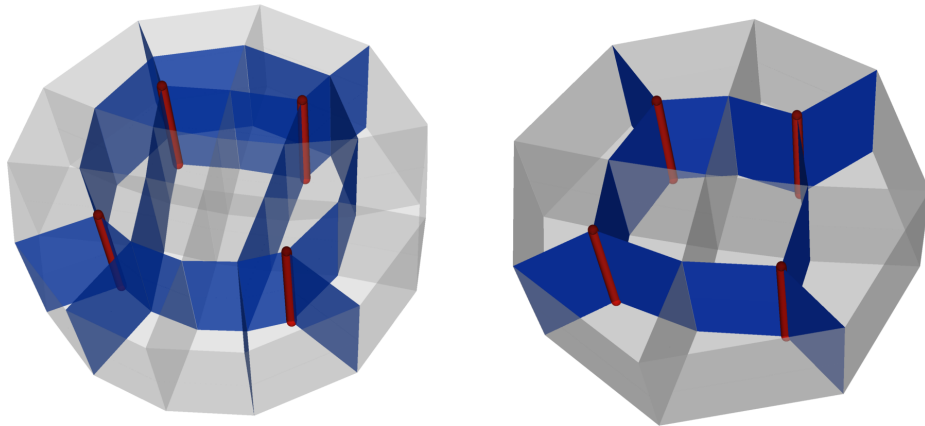


Figure 10.5: Base complex optimization can be trivially extended if the hexahedral mesh does not contain singular nodes. Here, a 2D optimize hexahedral meshing of a cylinder.

as in Figure 10.5. The problems appear when the isosurfaces contain singularities generated by orthogonal intersecting isosurfaces. The isosurface can have branch points where in the 2D case, we will just stop the tracing when a singularity is reached. Understanding operations on the singularities of hexahedral meshes such as collapses, creations and splits are still an open problem. In our investigation, we realize that merging singular edges into node points is indispensable to obtain an optimal base complex for hexahedral meshes. This is also not anymore a matching problem.

However, we can still apply the graph construction and the matching formulation by restricting the list of candidate connections only on valid removal base complex sheets as in [GDC15]. The main problem appears on the handling of singular nodes. There is no much degrees of freedom for the ports at these singularities. They behave like boundary ports in the quadrilateral mesh case i.e., we cannot assign any candidates to them in order to preserve the singularity of the initial hexahedral mesh. Hence, they do not enter in the matching problem but only in the illegal crossing surfaces.

CHAPTER 11

CONCLUSIONS

Through the development of this thesis, we have been able to understand the graph nature of the quad layout problem on manifold meshes. Previous works have attempted to solve the problem either as iterative greedy constructions or as a quadratic optimization. We have used the graph nature of the problem and found its beautiful connection to the perfect matching problem with disjunctive constraints which is classical in graph theory. Our construction is based on the concept of singularity-free graph and disjunctive constraints guaranteeing the quad topology of the resulting layouts. We have shown the effectiveness of our approach for triangle meshes and for the global structure optimization of quad meshes. The new approach is not limited to these classes of meshes, it could as well be used for quad layout generation on non-conforming meshes such as T-meshes or applied to the global optimization of hexahedral meshes.

Moreover, we have introduced the concept of regular surfaces which are very symmetric space models for regular maps. The visualization of regular maps is a very challenging problem but our concept of targetless tubification increases the number of realized regular maps in their most symmetric visualization. Using a novel identification in hyperbolic space which takes advantage of the group structure of the regular map, we are able to visualize the regular surfaces of all regular maps without an actual embedding of the surface. We have also introduced a covering approach which enables us to make use of symmetric shapes such as the Platonic solids as base shape to generate higher genus surfaces. These surfaces turn out to be nice space models for other regular maps.

There are many interesting open questions which can be addressed for future works from both topics.

- **Quad Layouts**

- *A strong proof of existence of a perfect matching for the graph construction on triangle meshes.*

We already proved that a non-constrained perfect matching would be enough to construct a solution of the linear program P_1 (introduced in Chapter 5) but there is no guarantee that the graph constructed by our method gives even one perfect matching. Nevertheless, we conjecture that stopping at the spirals always gives a perfect matching in \mathcal{G} .

- *The correct number and placement of singularities.*

The quality of a quad layout depends strongly on the number of singularities, allowing singularities to move or merge will improve the layout quality.

- *A larger graph \mathcal{G} .*

We use singularity-free triangles for consistent candidate collections.

The introduction of non-singularity-free triangles which ignore geodesic homotopy classes will enlarge the space of feasible solutions.

- *Extension of the matching problem to 3D volume.*

The tracing of isolines in a parameterization does not generalize on hexahedral meshes. Isosurfaces might have branch lines on hexahedral meshes, making them non simply connected where for isolines, such phenomenon does not appear. We would like to see the extent of our algorithm on hexahedral meshes.

- **Regular Maps**

- *Better heuristics to find pairings of source and target regular maps.*

Our pairing of source and target maps is based on [vW09] which reduces the number of possible realizations to having only 4-fold symmetry in each tube. The correlation between the symmetry groups of regular maps is not yet fully understood. In other words, why can some regular surfaces of regular maps be used as space models of others and some not? This is a deep group theoretical problem which may involve advanced algebraic geometric notions.

- *Visualization of the Hurwitz surfaces.*

Hurwitz surfaces are compact Riemann surfaces with exactly $84(g - 1)$ automorphisms where g is the genus of the surface. These are genus g surfaces with the highest symmetry. They are $\{3, 7\}$ triangle tilings

and the lower genus one include: the Klein quartic (genus 3), Macbeath surface (genus 7) and the First Hurwitz triplet (genus 14). We give a realization of the Klein Quartic in Figure 9.2. Finding a symmetric embedding of the other two regular maps is still an open problems.

APPENDIX A

ENERGY MINIMIZATION

Problem Formulation Given a triangle based cross field $\mathcal{F} = \{(\mathbf{u}_t, \mathbf{v}_t)\}_{t \in T_{\mathcal{M}}}$, we would like to find a parameterization map $\{t_i, f_i = (u_i, v_i)\}_{t_i \in T_{\mathcal{M}}}$ whose per triangle gradients align best to \mathcal{F} . It is equivalent to minimizing the energy

$$E_f = \sum_{t \in T_{\mathcal{M}}} \text{Area}(t) E_t \quad (\text{A.1})$$

with $E_t = (\|h\nabla u_t - \mathbf{u}_t\|^2 + \|h\nabla v_t - \mathbf{v}_t\|^2)$ such that for all edge $e \in f_i(t_i \cap t_j)$, there exists a transformation φ_{ij} satisfying the curl free condition

$$\varphi_{ij}(\mathbf{e}) = J^{r_{ij}}(\mathbf{e}). \quad (\text{A.2})$$

Discretization Let us consider the space of continuous piecewise linear functions in each triangle,

$$S_h = \{u \in C^0(\mathcal{M}), u \text{ is linear in each triangle of } \mathcal{M}\}$$

where $C^0(\mathcal{M})$ is the space of continuous function.

S_h is a vector space spanned by the vertex based Lagrange basis functions $(\varphi_i)_{i \in V_{\mathcal{M}}}$ which are linear functions defined at each vertex satisfying the Kronecker delta,

$$\varphi_i(j) = \delta_{ij}$$

which is 1 at the vertex p_i and zero everywhere else. For all $u \in S_h$ and for all

$t \in T_{\mathcal{M}}$,

$$u_t = u_0^t \varphi_0 + u_1^t \varphi_1 + u_2^t \varphi_2$$

and

$$\nabla u_t = u_0^t \nabla_{|t} \varphi_0 + u_1^t \nabla_{|t} \varphi_1 + u_2^t \nabla_{|t} \varphi_2$$

where φ_i , $i = 0, 1, 2$ are the basis functions defined at the vertices of t_i . The gradient of a basis function $\nabla_{|t} \varphi_i$ in the triangle t is defined by the 90 degree rotated edge e opposite to p_i in t scaled by the inverse of the area of t ,

$$\nabla_{|t} \varphi_i = \frac{1}{2\text{Area}(t)} J e.$$

Consider $u_t = (u_0^t, u_1^t, u_2^t)$ and $v_t = (v_0^t, v_1^t, v_2^t)$ the coefficient of the scalar functions at a triangle t . Then, E_t can be formulated into the following quadratic energy.

$$E_t = (u_t^T \mathbf{A} u_t + \mathbf{B}_u u_t + C_u) + (v_t^T \mathbf{A} v_t + \mathbf{B}_v v_t + C_v) \quad (\text{A.3})$$

$$= E(u_t) + E(v_t) \quad (\text{A.4})$$

where $\mathbf{A} \in \mathbb{R}^{3 \times 3}$, $\mathbf{B}_x \in \mathbb{R}^3$ and $C_u \in \mathbb{R}$ such that

$$\begin{cases} \mathbf{A}_{ij} &= h^2 \langle \nabla_{|t} \varphi_i, \nabla_{|t} \varphi_j \rangle \\ (\mathbf{B}_x)_i &= -2h \langle \nabla_{|t} \varphi_i, \mathbf{x}_t \rangle \\ C_x &= \|\mathbf{x}_t\|^2. \end{cases} \quad (\text{A.5})$$

The usual way to minimize Equation A.1 is to take the partial derivatives, make it equal to zero and solve the linear system. It is a very fast computation but not robust in practice. The resulting parameterization is not in general injective which is not suitable for our application unless stiffening like approach [?] is applied. In order to guarantee (partially or generally) injectivity, we consider the problem as a global optimization problem, i.e. we would like to find (u, v) that minimizes Equation A.1 subject to curl-freeness and local as well as global injectivity. Mainly, we want to solve the following linear program

$$\text{minimize } \sum_{t \in T_{\mathcal{M}}} \text{Area}(t) E_t \quad (\text{A.6})$$

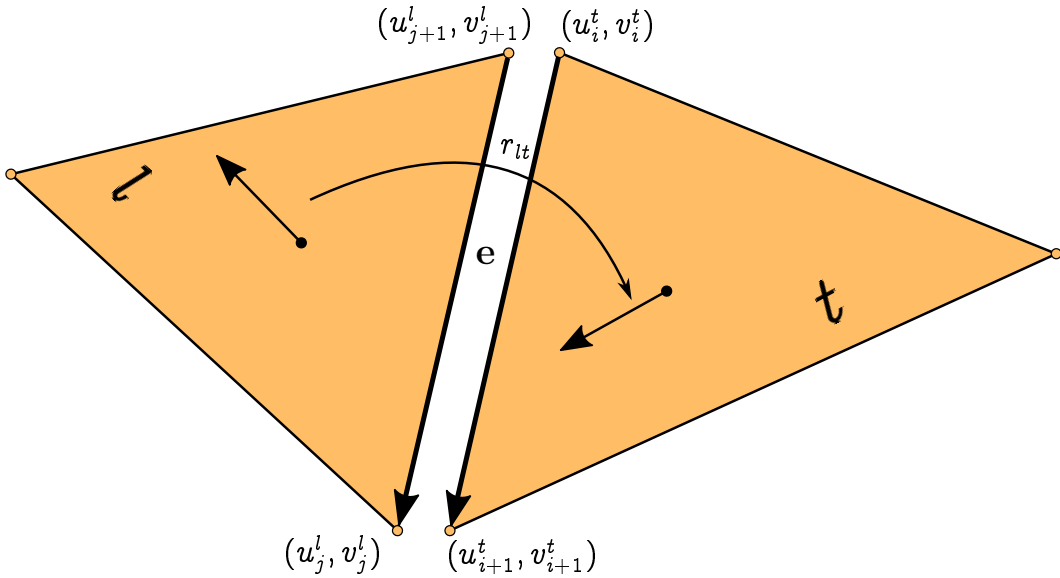
subject to Equation A.2 and the bisector condition for injectivity introduced in Chapter 6. The system has $6|T_{\mathcal{M}}|$ unknowns ((u, v) coordinates per vertex, per face) which can be reduced by defining a spanning tree on \mathcal{M} inducing a cut graph connecting the singularities and making \mathcal{M} a topological disk.

Let us consider two triangles l and t sharing an edge e represented respectively by $(u_{i+1}^t - u_i^t, v_{i+1}^t - v_i^t)$ and $(u_{j+1}^l - u_j^l, v_{j+1}^l - v_j^l)$ in both chart. More, denote by \mathcal{C} the set of edges along the cut path of the surface. Then for all $e \in \mathcal{C}$, the curl-freeness condition becomes,

$$\begin{pmatrix} u_{i+1}^t - u_i^t \\ v_{i+1}^t - v_i^t \end{pmatrix} - \mathbf{J}^{r_{lt}} \begin{pmatrix} u_{j+1}^l - u_j^l \\ v_{j+1}^l - v_j^l \end{pmatrix} = 0 \quad (\text{A.7})$$

and for all $e \notin \mathcal{C}$,

$$\begin{pmatrix} u_i^t \\ v_i^t \end{pmatrix} - \mathbf{J}^{r_{lt}} \begin{pmatrix} u_{j+1}^l \\ v_{j+1}^l \end{pmatrix} = 0 \text{ and } \begin{pmatrix} u_{i+1}^t \\ v_{i+1}^t \end{pmatrix} - \mathbf{J}^{r_{lt}} \begin{pmatrix} u_j^l \\ v_j^l \end{pmatrix} = 0 \quad (\text{A.8})$$



Where r_{lt} is the matching between the two frame directions from l to t . We use CPLEX to solve the linear program by constraining only the triangles adjacent to singularities. Far away fold-over does not affect our isoline tracing algorithm.

APPENDIX B

SOLVER COMPARISON

We compare existing optimization packages to solve the minimum weight perfect matching with pair constraints on triangle meshes. The input graphs are derived from our construction which have in most cases more number of conflicting edges than number of edges. For each solver, we emphasize on the flexibility, i.e. free versus commercial. The free software should be open-source and should run in reasonable amount of time. There exists several open sources as well as commercial software for integer optimization problem. Most popular among them are

- **GLPK** (GNU Linear Programming Kit). GLPK is a free and open source software written in ANSI C which uses the Branch-and-bound algorithm for mixed integer problems. It is authored by Andrew O. Makhorin [Mak00] and is still actively maintained.
- **LP_SOLVE**. `lp_solve` is a Mixed Integer Linear Program solver which is also open source written in ANSI C and uses the Branch-and-bound algorithm. It was originally developed by Michel Berkelaar at Eindhoven University of Technology, several enhancement followed later on [MBN04].
- **SCIP**. SCIP is a non-commercial solver for mixed integer programming developed at the Zuse Institute Berlin. It is implemented as C callable library and can also be used as a standalone program to solve mixed integer programs given in classical data format [Ach09].
- **CPLEX**. CPLEX [IBM15] is one of the standard commercial software to solve mixed integer problems. It uses the primal-dual simplex method implemented in C, highly parallelized routines and several interfacing with high-level programming language. It is unfortunately very expensive but the software can be fully used for academic purpose.

Describing the specific methods and routines used in each software is not our main focus. We would like to check which of them is best appropriate to our specific problem.

B.1 INPUT MODELS

We choose four triangulated models to run the algorithm which are classical benchmark models also used by IGM [BCE⁺13] and DLM [CBK12]. The problem is first generated as .lp file from CPLEX and then given as input to the other software. The CPU timing is in second and is return by the respective packages. If the solver takes too much time to look for a feasible solution i.e. more than one hour, we give it the mention fail.

B.2 STATISTICS

Models	Graph		Timings			
	E	I	GLPK	lp_solve	SCIP	CPLEX
Botijo	1736	32192	9.4	224.5	2.0	0.64
Elk	1023	14150	7.0	11.0	0.9	0.46
Fertility	1118	18831	1.8	0.9	2.6	1.09
Rockerarm	609	9441	0.8	0.2	0.9	0.15
Boudha	2657	81139	49.5	Fail	7.28	1.9

Table B.1: Comparison of several integer program optimization package on closed parameterized surfaces allowing multiple edges in \mathcal{G} .

As expected, CPLEX performs best in our benchmarks. For the free software, it is hard to decide which of them is better. On the Botijo model **LP_SOLVE** took too much time where on the Fertility, it outperformed all other free software. We find **SCIP** to be more stable among the three software. It scales very well with respect to the graph and the number of conflicting edges. **LP_SOLVE** fails to give a solution on the Boudha model where **SCIP** still performs reasonably well.

BIBLIOGRAPHY

- [Ach09] Tobias Achterberg. Scip: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. <http://mpc.zib.de/index.php/MPC/article/view/4>.
- [BCE⁺13] David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. Integer-grid maps for reliable quad meshing. *ACM Trans. Graph.*, 32(4):98:1–98:12, July 2013.
- [BLK11] David Bommes, Timm Lempfer, and Leif Kobbelt. Global structure optimization of quadrilateral meshes. *Computer Graphics Forum*, 30(2):375–384, 2011.
- [BLP⁺12] David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva a, Marco Tarini, and Denis Zorin. State of the art in quad meshing. *Eurographics STARS*, 2012.
- [BMRJ04] Ioana Boier-Martin, Holly Rushmeier, and Jingyi Jin. Parameterization of triangle meshes over quadrilateral domains. *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 193–203, 2004.
- [BZK09] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3):77:1–77:10, July 2009.
- [CBK12] Marcel Campen, David Bommes, and Leif Kobbelt. Dual loops meshing: Quality quad layouts on manifolds. *ACM Trans. Graph.*, 31(4):110:1–110:11, July 2012.
- [CD01] Marston Conder and Peter Dobcsányi. Determination of all regular maps of small genus. *Journal of Combinatorial Theory, Series B*, 81(2):224 – 242, 2001.

- [CK14] Marcel Campen and Leif Kobbelt. Quad layout embedding via aligned parameterization. *Computer Graphics Forum*, 33:69–81, 2014.
- [CM80] H.S.M. Coxeter and W.O.J. Moser. *Generators and relations for discrete groups*. Ergebnisse der Mathematik und ihrer Grenzgebiete. Springer-Verlag, 1980.
- [DBG⁺06] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. *ACM Trans. Graph.*, 25(3):1057–1066, July 2006.
- [DPSW11] Andreas Darmann, Ulrich Pferschy, Joachim Schauer, and Gerhard J. Woeginger. Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159(16):1726 – 1735, 2011. 8th Cologne/Twente Workshop on Graphs and Combinatorial Optimization (CTW 2009).
- [DSC09] Joel Daniels, Cláudio T. Silva, and Elaine Cohen. Semi-regular quadrilateral-only remeshing from simplified base domains. *Comput. Graph. Forum*, 28(5):1427–1435, 2009.
- [DSSC08] Joel Daniels, Cláudio T. Silva, Jason Shepherd, and Elaine Cohen. Quadrilateral mesh simplification. *ACM Trans. Graph.*, 27(5):148:1–148:9, December 2008.
- [DVPSH15a] Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. Integrable PolyVector fields. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 34(4), 2015.
- [DVPSH15b] Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. Integrable polyvector fields. *ACM Trans. Graph.*, 34(4):38:1–38:12, July 2015.
- [EBCK13] Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. Qex: Robust quad mesh extraction. *ACM Trans. Graph.*, 32(6):168:1–168:10, November 2013.
- [ECBK14] Hans-Christian Ebke, Marcel Campen, David Bommes, and Leif Kobbelt. Level-of-detail quad meshing. *ACM Trans. Graph.*, 33(6):184:1–184:11, November 2014.
- [Eck96] Matthias Eck. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *Proc. SIGGRAPH'96*, pages 325–334, 1996.

- [Edm65a] Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research National Bureau of Standards Section B*, 69:125–30, 1965.
- [Edm65b] Jack Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, February 1965.
- [EGKT08] David Eppstein, Michael T. Goodrich, Ethan Kim, and Rasmus Tamstorf. Motorcycle graphs: Canonical quad mesh partitioning. *Proceedings of the Symposium on Geometry Processing*, pages 1477–1486, 2008.
- [GDC15] Xifeng Gao, Zhigang Deng, and Guoning Chen. Hexahedral mesh reparameterization from aligned base-complex. *ACM Trans. Graph.*, 34(4):142:1–142:10, July 2015.
- [IBM15] IBM. Ibm ilog cplex optimization studio. <http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/>, May 2015.
- [KCPS13] Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. Globally optimal direction fields. *ACM Trans. Graph.*, 32(4), 2013.
- [KNP07] Felix Kälberer, Matthias Nieser, and Konrad Polthier. Quadcover - surface parameterization using branched coverings. *Comput. Graph. Forum*, 26(3):375–384, 2007.
- [Lip12] Yaron Lipman. Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph.*, 31(4):108:1–108:13, July 2012.
- [LP84] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14:393–410, 1984.
- [Mak00] Andrew Makhorin. The gnu linear programming kit (glpk). <http://www.gnu.org/software/glpk/>, 2000.
- [MBN04] Kjell Eikland Michel Berkelaar and Peter Notebaert. lpsolve 5.5. <http://lpsolve.sourceforge.net/5.5/>, 2000–2004.
- [MPKZ10] Ashish Myles, Nico Pietroni, Denis Kovacs, and Denis Zorin. Feature-aligned t-meshes. *ACM Trans. Graph.*, 29(4):1–11, 2010.
- [MPZ14] Ashish Myles, Nico Pietroni, and Denis Zorin. Robust field-aligned global parametrization. *ACM Trans. Graph.*, 33(4):XX:1–XX:14, July 2014.

- [OZP13] Temel Öncan, Ruonan Zhang, and Abraham P. Punnen. The minimum cost perfect matching problem with conflict pair constraints. *Computers and Operations Research*, 40(4):920 – 930, 2013.
- [PL86] D. Plummer and L. Lovász. *Matching Theory*. North-Holland Mathematics Studies. Elsevier Science, 1986.
- [PP03] Konrad Polthier and Eike Preuss. Identifying vector field singularities using a discrete Hodge decomposition. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 113–134. Springer Verlag, 2003.
- [PPTSH14] Daniele Panozzo, Enrico Puppo, Marco Tarini, and Olga Sorkine-Hornung. Frame fields: Anisotropic and non-orthogonal cross fields. *ACM Trans. Graph.*, 33(4):134:1–134:11, July 2014.
- [Raz12] Faniry H. Razafindrazaka. Visualization of high genus regular maps. Master’s thesis, Freie Universität Berlin, 2012.
- [RLL⁺06] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. Periodic global parameterization. *ACM Trans. Graph.*, 25(4):1460–1485, October 2006.
- [RLS⁺12] J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen, and C. Geuzainet. Blossom-quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm. *Int. J. Numer. Meth. Engng.*, 89(9):1102–1119, March 2012.
- [RP13] Faniry H. Razafindrazaka and Konrad Polthier. The 6-ring. In *Proceedings of Bridges 2013: Mathematics, Music, Art, Architecture, Culture*, pages 279–286. Tessellations Publishing, 2013.
- [RP14] Faniry H. Razafindrazaka and Konrad Polthier. Regular surfaces and regular maps. In *Proceedings of Bridges 2014: Mathematics, Music, Art, Architecture, Culture*, pages 225–234. Tessellations Publishing, 2014.
- [RP15] Faniry H. Razafindrazaka and Konrad Polthier. *Realization of Regular Maps of Large Genus*, pages 239 – 252. Springer Berlin Heidelberg, 2015.
- [RRP15] Faniry H. Razafindrazaka, Ulrich Reitebuch, and Konrad Polthier. Perfect matching quad layouts for manifold meshes. *Computer*

- Graphics Forum (proceedings of EUROGRAPHICS Symposium on Geometry Processing)*, 34(5), 2015.
- [RS14] Nicolas Ray and Dmitry Sokolov. Robust polylines tracing for n-symmetry direction field on triangulated surfaces. *ACM Trans. Graph.*, 33(3):30:1–30:11, June 2014.
- [S07] Carlo H. Séquin. Symmetric embedding of locally regular hyperbolic tilings. In Reza Sarhangi and Javier Barrallo, editors, *Bridges Donostia: Mathematics, Music, Art, Architecture, Culture*, pages 379–388, London, 2007. Tarquin Publications. Available online at <http://archive.bridgesmathart.org/2007/bridges2007-379.html>.
- [S10] Carlo H. Séquin. My search for symmetrical embeddings of regular maps. In George W. Hart and Reza Sarhangi, editors, *Proceedings of Bridges 2010: Mathematics, Music, Art, Architecture, Culture*, pages 85–94, Phoenix, Arizona, 2010. Tessellations Publishing. Available online at <http://archive.bridgesmathart.org/2010/bridges2010-85.html>.
- [TPC⁺10] Marco Tarini, Nico Pietroni, Paolo Cignoni, Daniele Panozzo, and Enrico Puppo. Practical quad mesh simplification. *Computer Graphics Forum (Special Issue of Eurographics 2010 Conference)*, 29(2):407–418, 2010.
- [TPP⁺11] Marco Tarini, Enrico Puppo, Daniele Panozzo, Nico Pietroni, and Paolo Cignoni. Simple quad domains for field aligned mesh parametrization. *ACM Transactions on Graphics, Proceedings of SIGGRAPH Asia 2011*, 30(6), 2011.
- [Vax14] Amir Vaxman. A projective framework for polyhedral mesh modelling. *Computer Graphics Forum*, 33(8):121–131, 2014.
- [vW09] Jarke J. van Wijk. Symmetric tiling of closed surfaces: Visualization of regular maps. *ACM Trans. Graph.*, 28(3):49:1–49:12, July 2009.
- [Wac75] Eugene L. Wachspress. *A Rational Finite Element Basis*. Academic Press, New York, 1975.
- [WMZ12] Ofir Weber, Ashish Myles, and Denis Zorin. Computing extremal quasiconformal maps. *Comput. Graph. Forum*, 31(5):1679–1689, 2012.

- [ZHLB10] Muiyang Zhang, Jin Huang, Xinguo Liu, and Hujun Bao. A wave-based anisotropic quadrangulation method. *ACM Trans. Graph.*, 29(4):118:1–118:8, July 2010.

ZUSAMMENFASSUNG

Diese Dissertation beschäftigt sich mit zwei grundsätzlichen Konzepten in der Oberflächentopologie. Der erste Teil behandelt das Problem, eine gegebene Oberfläche in Vierecksgitter zu pflastern. Ist eine regelmäßige Pflasterung gegeben, werden im zweiten Teil Methoden vorgeschlagen, um eine symmetrische geschlossene Oberfläche zu bauen und diese symmetrisch einzubetten. Oberflächepflasterung hat eine breite Reihe von Anwendung in der Computergrafik und Geometrieverarbeitung.

Für eine gegebene Oberfläche schlagen wir einen flexiblen Algorithmus vor, welcher automatisch Vierecksgitter von hoher Qualität auf dieser Oberfläche erzeugt. Strukturierte Darstellung einer Oberfläche durch Vierecksgitter hat praktische Anwendungen in unterschiedlichen Bereichen, wie Unterteilungsflächen, Approximation von Oberflächen, Kompression und Hierarchien in der finiten Element Methode. Unser Ansatz besteht darin, einen Singularitäten-Graph eines Rahmenfeldes zu konstruieren. Dieses Feld dekodiert die lokalen Strukturen der Geometrie. Das Problem, ein Vierecksgitter auf der Oberfläche zu erzeugen ist damit äquivalent zur Konstruktion eines minimalen gewichteten Matchings mit disjunkten Nebenbedingungen. Das resultierende Vierecksgitter ist von hoher Qualität in dem Sinne, als das es trotz grober Auflösung den lokalen Strukturen der Geometrie folgt. Anwendungen unserer Methode auf Dreiecks- und Vierecksgitter zeigen, dass die Ergebnisse sich mit denen anderer aktueller Methoden messen können.

Für eine gegebene reguläre Pflasterung, präziser eine reguläre Karte, führen wir die entsprechende reguläre Oberfläche ein. Diese ist die passendste Oberfläche von hohem Genus, welche die gegebene Pflasterung realisiert. Das Visualisieren regulärer Karten ist ein schwieriges Problem. Alle regulären Karten, bzw. symmetrischen Pflasterungen einer Oberfläche von Genus bis 302 sind algebraisch bereits bekannt. Sie liegen in Form von Symmetriegruppen vor, welche auf den entsprechenden universellen Abdeckungsräumen agieren. Jedoch ist wenig über die geometrischen Realisierungen bekannt, d.h. über das Finden hoch-symmetrischer Einbettungen abgeschlossener Oberflächen und die passenden hoch-symmetrischen Pflasterungen. In dieser Arbeit führen wir einen Algorithmus ein, welcher automatisch räumliche Modelle für einige dieser regulären Karten erstellt und dabei schöne Oberflächen mit sehr hoher Symmetrie erzeugt.