Mesh Parameterization: Theory and Practice

Article · December 2008

DOI: 10.1145/1508044.1508091 · Source: OAI

3 authors:

Kai Hormann
University of Lugano
136 PUBLICATIONS **5,686** CITATIONS
SEE PROFILE

Bruno Eric Levy
National Institute for Research in Computer Science and Control
167 PUBLICATIONS **9,946** CITATIONS
SEE PROFILE

Alla Sheffer
University of British Columbia
123 PUBLICATIONS **7,634** CITATIONS
SEE PROFILE

Siggraph Course Notes

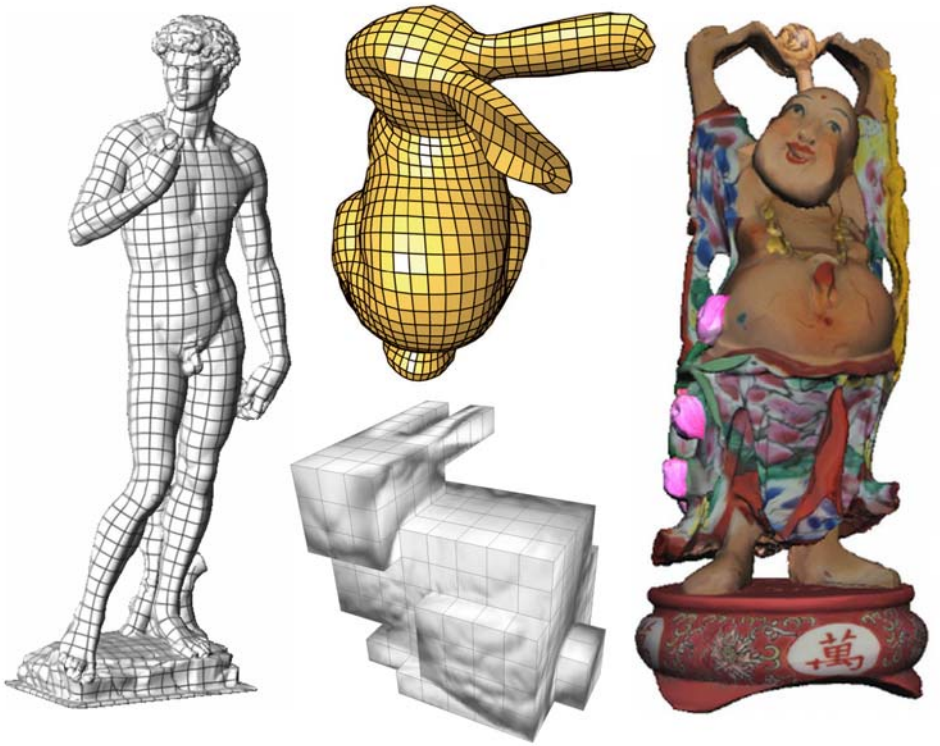
Mesh Parameterization: Theory and Practice

Kai Hormann
Clausthal University of Technology

Bruno Lévy
INRIA

Alla Sheffer
The University of British Columbia

August 2007



Summary

Mesh parameterization is a powerful geometry processing tool with numerous computer graphics applications, from texture mapping to animation transfer. This course outlines its mathematical foundations, describes recent methods for parameterizing meshes over various domains, discusses emerging tools like global parameterization and inter-surface mapping, and demonstrates a variety of parameterization applications.

Prerequisites

The audience should have had some prior exposure to mesh representation of geometric models and a working knowledge of vector calculus, elementary linear algebra, and the fundamentals of computer graphics. Optional pre-requisites: some lectures may also assume some familiarity with differential geometry, graph theory, harmonic functions, and numerical optimization.

Intended Audience

Graduate students, researchers, and application developers who seek to understand the concepts and technologies used in mesh parameterization and wish to utilize them. Listeners get an overview of the spectrum of processing applications that benefit from parameterization and learn how to evaluate different methods in terms of specific application requirements.

Sources

These notes are largely based on the following survey papers, which can be found on the author webpages:

- M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer, 2005.
- A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision*, 2(2):105–171, 2006.
- B. Lévy. Parameterization and deformation analysis on a manifold. Technical report, ALICE, 2007. `http://alice.loria.fr/publications`.
- Source code (Graphite, OpenNL, ...) can be found from `http://alice.loria.fr/software`.

Supplemental material and presentation slides are also available on the presenter webpages.

Course Overview

For any two surfaces with similar topology, there exists a bijective mapping between them. If one of these surfaces is a triangular mesh, the problem of computing such a mapping is referred to as mesh parameterization. The surface that the mesh is mapped to is typically called the parameter domain. Parameterization was introduced to computer graphics for mapping textures onto surfaces. Over the last decade, it has gradually become a ubiquitous tool for many mesh processing applications, including detail-mapping, detail-transfer, morphing, mesh-editing, mesh-completion, remeshing, compression, surface-fitting, and shape-analysis. In parallel to the increased interest in applying parameterization, various methods were developed for different kinds of parameter domains and parameterization properties.

The goal of this course is to familiarize the audience with the theoretical and practical aspects of mesh parameterization. We aim to provide the skills needed to implement or improve existing methods, to investigate new approaches, and to critically evaluate the suitability of the techniques for a particular application.

The course starts with an introduction to the general concept of parameterization and an overview of its applications. The first half of the course then focuses on planar parameterizations while the second addresses more recent approaches for alternative domains. The course covers the mathematical background, including intuitive explanations of parameterization properties like bijectivity, conformality, stretch, and area-preservation. The state-of-the-art is reviewed by explaining the main ideas of several approaches, summarizing their properties, and illustrating them using live demos. The course addresses practical aspects of implementing mesh parameterization discussing numerical algorithms for robustly and efficiently parameterizing large meshes and providing tips on how to handle the variety of often contradicting criteria when choosing an appropriate parameterization method for a specific target application. We conclude by presenting a list of open research problems and potential applications that can benefit from parameterization.

Speakers

Mathieu Desbrun, Caltech, USA

Mathieu Desbrun is an associate professor at the California Institute of Technology (Caltech) in the Computer Science department. After receiving his Ph.D. from the National Polytechnic Institute of Grenoble (INPG), he spent two years as a post-doctoral researcher at Caltech before starting a research group at the University of Southern California (USC). His research interests revolve around applying discrete differential geometry (differential, yet readily-discretizable computational foundations) to a wide range of fields and applications such as meshing, parameterization, smoothing, fluid and solid mechanics, etc.

Kai Hormann, Clausthal University of Technology, Germany

Kai Hormann is an assistant professor for Computer Graphics in the Department of Informatics at Clausthal University of Technology in Germany. His research interests are focussed on the mathematical foundations of geometry processing algorithms as well as their applications in computer graphics and related fields. Dr. Hormann has co-authored several papers on parameterization methods, surface reconstruction, and barycentric coordinates. Kai Hormann received his PhD from the University of Erlangen in 2002 and spent two years as a postdoctoral research fellow at the Multi-Res Modeling Group at Caltech, Pasadena and the CNR Institute of Information Science and Technologies in Pisa, Italy.

Bruno Lévy, INRIA, France

Bruno Lévy is a Researcher with INRIA. He is the head of the ALICE INRIA Project-Team. His main contributions concern texture mapping and parameterization methods for triangulated surfaces, and are now used by some 3D modeling software (including Maya, Silo, Blender, Gocad and Catia). He obtained his Ph.D in 1999, from the INPL (Institut National Polytechnique de Lorraine). His work, entitled "Computational Topology: Combinatorics and Embedding", was awarded the SPECIF price in 2000 (best French Ph.D. thesis in Computer Sciences).

Alla Sheffer, University of British Columbia, Canada

Alla Sheffer is an assistant professor in the Computer Science department at the University of British Columbia (Canada). She conducts research in the areas of computer graphics and computer aided engineering. Dr. Sheffer is predominantly interested in the algorithmic aspects of digital geometry processing, focusing on several fundamental problems of mesh manipulation and editing. Her recent research addresses algorithms for mesh parameterization, processing of developable surfaces, mesh editing and segmentation. She co-authored several parameterization methods which are used in popular 3D modelers including Blender and Catia. Alla Sheffer received her PhD from the Hebrew University of Jerusalem in 1999. She spent two years as a postdoctoral researcher in the University of Illinois at Urbana-Champaign, and then two years as an assistant professor at Technion, Israel.

Kun Zhou, Microsoft, USA

Kun Zhou is a Lead Researcher of the graphics group at Microsoft Research Asia. He received his B.S. and Ph.D degrees in Computer Science from Zhejiang University in 1997 and 2002 respectively. Kun s research interests include geometry processing, texture processing and real-time rendering. He holds over ten granted and pending US patents. Some of these techniques have been integrated in Windows Vista, DirectX and XBOX SDK.

Syllabus

Morning session

- Introduction [15 min, Alla]
- Differential Geometry Primer [30 min, Kai]
- Barycentric Mappings [30 min, Kai]
- Setting the Boundary Free [30 min, Bruno]
- Indirect methods - ABF and Circle Patterns [30 min, Alla]
- Making it work in practice - Segmentation and Constraints [45 min, Kun Zhou]
- Comparison and Applications of Planar Methods [30 min, Kai]

Afternoon session

- Spherical Parameterization [30 min, Alla]
- Discrete Exterior Calculus in a Nutshell [30 min, Mathieu]
- Global Parameterization [45 min, Mathieu]
- Cross-Parameterization/Inter-surface mapping [30 min, Alla]
- Making it work in practice - Numerical Aspects [30 min, Bruno]
- Comparison and Applications of Global Methods [30 min, Bruno]
- Open Problems and Q/A [15 min, all]

Contents

Chapter 1

Introduction

Given any two surfaces with similar topology it is possible to compute a one-to-one and onto mapping between them. If one of these surfaces is represented by a triangular mesh, the problem of computing such a mapping is referred to as mesh parameterization. The surface that the mesh is mapped to is typically referred to as the parameter domain. Parameterizations between surface meshes and a variety of domains have numerous applications in computer graphics and geometry processing as described below. In recent years numerous methods for parameterizing meshes were developed, targeting diverse parameter domains and focusing on different parameterization properties.

This course reviews the various parameterization methods, summarizing the main ideas of each technique and focusing on the practical aspects of the methods. It also provides examples of the results generated by many of the more popular methods. When several methods address the same parameterization problem, the survey strives to provide an objective comparison between them based on criteria such as parameterization quality, efficiency and robustness. We also discuss in detail the applications that benefit from parameterization and the practical issues involved in implementing the different techniques.

1.1 Applications

Surface parameterization was introduced to computer graphics as a method for mapping textures onto surfaces [Bennis et al., 1991; Maillot et al., 1993]. Over the last decade, it has gradually become a ubiquitous tool, useful for many mesh processing applications, discussed below.

Detail Mapping

Detailed objects can be efficiently represented by a coarse geometric shape (polygonal mesh or subdivision surface) with the details corresponding to each triangle stored in a separate 2D array. In traditional texture mapping the details are the colors of the respective pixels. Models can be further enriched by storing bump, normal, or displacement maps. Recent techniques [Peng et al., 2004; Porumbescu et al., 2005] model a thick region of space in the neighborhood of the surface by using a volumetric texture, rather

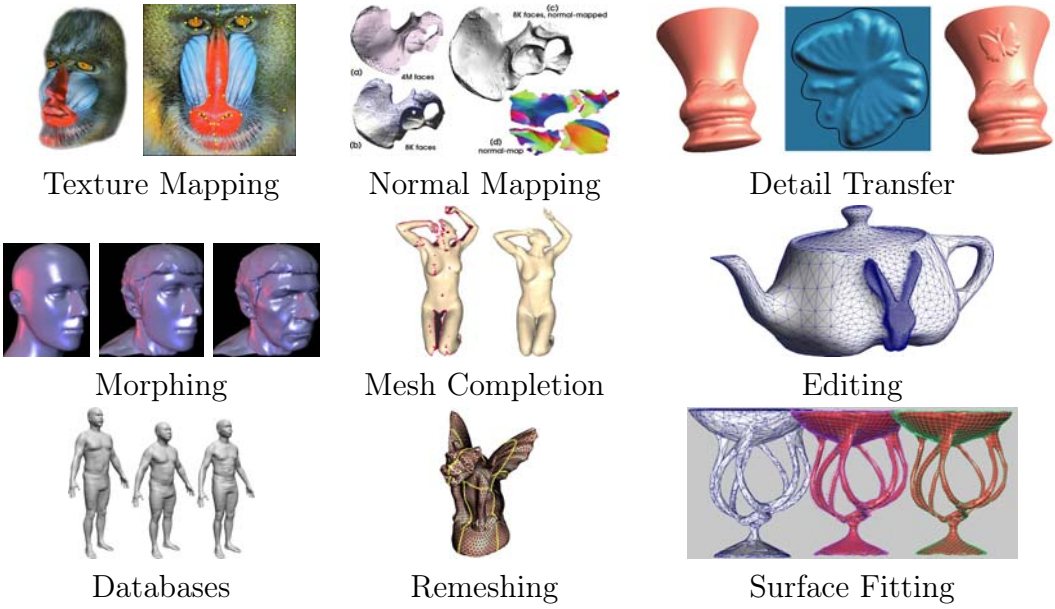


Figure 1.1: Parameterization Applications.

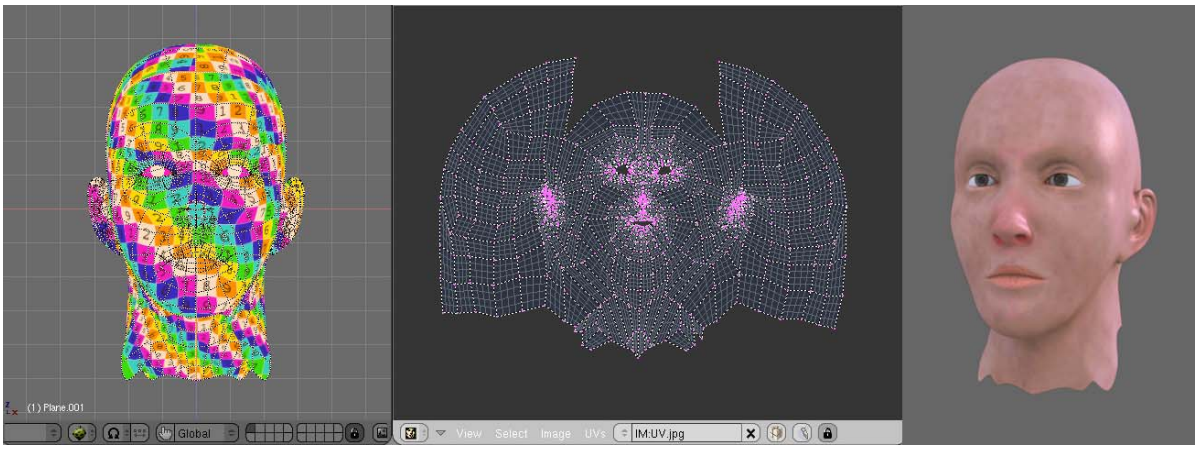


Figure 1.2: Application of parameterization: texture mapping (Least Squares Conformal Maps implemented in the Open-Source Blender modeler).

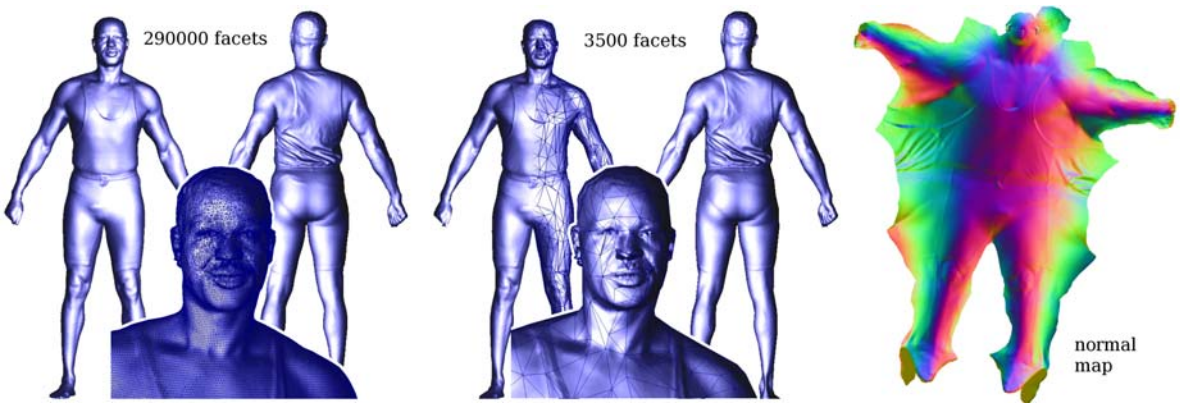


Figure 1.3: Application of parameterization: appearance-preserving simplification. All the details are encoded in a normal map, applied onto a dramatically simplified version of the model (1.5% of the original size).

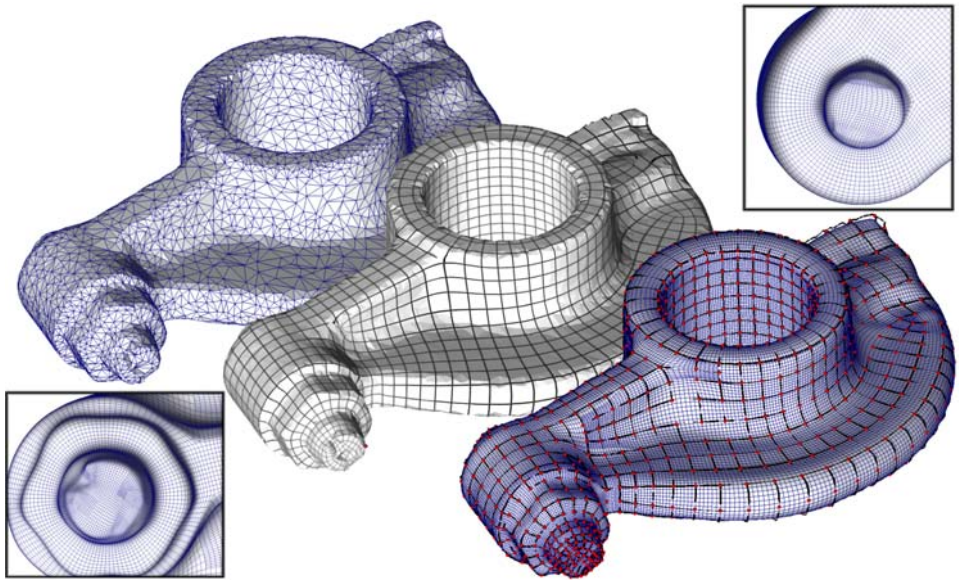


Figure 1.4: A global parameterization realizes an abstraction of the initial geometry. This abstraction can then be re-instanciated into alternative shape representations.

than a 2D one. Such techniques are needed in order to model detail with complicated topology or detail that cannot be easily approximated locally by a height field, such as sparsely interwoven structures or animal fur. The natural way to map details to surfaces is using planar parameterization.

Detail Synthesis

While the goal of texture mapping is to represent the complicated appearance of 3D objects, several methods make use of mesh parameterization to create the local detail necessary for a rich appearance. Such techniques can use as input flat patches with sample detail, e.g. [Soler et al., 2002]; parametric or procedural models; or direct user input and editing [Carr and Hart, 2004]. The type of detail can be quite varied and the intermediate representations used to create it parallel the final representations used to store it.

Morphing and Detail Transfer

A map between the surfaces of two objects allows the transfer of detail from one object to another (e.g. [Praun et al., 2001]), or the interpolation between the shape and appearance of several objects [Alexa, 2000; Kraevoy and Sheffer, 2004; Schreiner et al., 2004]. By varying the interpolation ratios over time, one can produce morphing animations. In spatially-varying and frequency-varying morphs, the rate of change can be different for different parts of the objects, or different frequency bands (coarseness of the features being transformed) [Allen et al., 2003; Kraevoy and Sheffer, 2004]. Such a map can either be computed directly or, as more commonly done, computed by mapping both object surfaces to a common domain. In addition to transferring the static appearance of surfaces, inter-surface parameterizations allow the transfer of animation data between shapes, either by transferring the local surface influence from bones of an animation rig, or by directly transferring the local affine transformation of each triangle in the mesh [Sumner and Popović, 2004].

Mesh Completion

Meshes from range scans often contain holes and multiple components. Lévy [2003] uses planar parameterization to obtain the natural shape for hole boundaries and to triangulate those. In many cases, prior knowledge about the overall shape of the scanned models exists. For instance, for human scans, templates of a generic human shape are readily available. Allen et al. [2003] and Anguelov et al. [2005] use this prior knowledge to facilitate completion of scans by computing a mapping between the scan and a template human model. Kraevoy and Sheffer [2005] develop a more generic and robust template-based approach for completion of any type of scans. The techniques typically use an inter-surface parameterization between the template and the scan.

Mesh Editing

Editing operations often benefit from a local parameterization between pairs of models. Biermann et al. [2002] use local parameterization to facilitate cut-and-paste transfer of details between models. They locally parameterize the regions of interest on the two models in 2D and overlap the two parameterizations. They use the parameterization to transfer shape properties from one model to the other. Lévy [2003] uses local parameterization for mesh composition in a similar manner. They compute an overlapping planar parameterization of the regions near the composition boundary on the input models and use it to extract and smoothly blend shape information from the two models.

Creation of Object Databases

Once a large number of models are parameterized on a common domain one can perform an analysis determining the common factors between objects and their distinguishing traits. For example on a database of human shapes [Allen et al., 2003] the distinguishing traits may be gender, height, and weight. Objects can be compared against the database and scored against each of these dimensions, and the database can be used to create new plausible object instances by interpolation or extrapolation of existing ones.

Remeshing

There are many possible triangulations that represent the same shape with similar levels of accuracy. Some triangulation may be more desirable than others for different applications. For example, for numerical simulations on surfaces, triangles with a good aspect ratio (that are not too small or too skinny are important for convergence and numerical accuracy. One common way to remesh surfaces, or to replace one triangulation by another, is to parameterize the surface, then map a desirable, well-understood, and easy to create triangulation of the domain back to the original surface. For example, Gu et al. [2002] use a regular grid sampling of a planar square domain, while other methods, e.g. [Guskov et al., 2000] use regular subdivision (usually 1-to-4 triangle splits) on the faces of a simplicial domain. Such locally regular meshes can usually support the creation of smooth surfaces as the limit process of applying subdivision rules. To generate high quality triangulations, Desbrun et al. [2002] parameterize the input mesh in the plane and then use planar Delaunay triangulation to obtain a high quality remeshing of the surface. One problem these methods face is the appearance of visible discontinuities along the cuts created to facilitate the parameterization. Surazhsky and Gotsman [2003] avoid global parameterization, and instead use local parameterization to move vertices along the mesh as part of an explicit remeshing scheme. Recent methods such as [Ray et al., 2006] use global parameterization to generate a predominantly quadrilateral mesh directly on the 3D surface.

Mesh Compression

Mesh compression is used to compactly store or transmit geometric models. As with other data, compression rates are inversely proportional to the data entropy. Thus higher compression rates can be obtained when models are represented by meshes that are as regular as possible, both topologically and geometrically. Topological regularity refers to meshes where almost all vertices have the same degree. Geometric regularity implies that triangles are similar to each other in terms of shape and size and vertices are close to the centroid of their neighbors. Such meshes can be obtained by parameterizing the original objects and then remeshing with regular sampling patterns [Gu et al., 2002]. The quality of the parameterization directly impacts the compression efficiency.

Surface Fitting

One of the earlier applications of mesh parameterization is surface fitting [Floater, 2000]. Many applications in geometry processing require a smooth analytical surface to be constructed from an input mesh. A parameterization of the mesh over a base domain significantly simplifies this task. Earlier methods either parameterized the entire mesh in the plane or segmented it and parameterized each patch independently. More recent methods, e.g. [Li et al., 2006] focus on constructing smooth global parameterizations and use those for fitting, achieving global continuity of the constructed surfaces.

Modeling from Material Sheets

While computer graphics focuses on virtual models, geometry processing has numerous real-world engineering applications. Particularly, planar mesh parameterization is an important tool when modeling 3D objects from sheets of material, ranging from garment modeling to metal forming or forging [Bennis et al., 1991; Julius et al., 2005]. All of these applications require the computation of planar patterns to form the desired 3D shapes. Typically, models are first segmented into nearly developable charts, and these charts are then parameterized in the plane.

Medical Visualization

Complex geometric structures are often better visualized and analyzed by mapping the surface normal-map, color, and other properties to a simpler, canonical domain. One of the structures for which such mapping is particularly useful is the human brain [Hurdal et al., 1999; Haker et al., 2000]. Most methods for brain mapping use the fact that the brain has genus zero, and visualize it through spherical [Haker et al., 2000] or planar [Hurdal et al., 1999] parameterization.

Chapter 2

Differential Geometry Primer

Before we go into the details of how to compute a mesh parameterization and what to do with it, let us quickly review some of the basic properties from differential geometry that will be essential for understanding the motivation behind the methods described later. For more details and proofs of these properties, we refer the interested reader to the standard literature on differential geometry and in particular to the books by do Carmo [1976], Klingenberg [1978], Kreyszig [1991], and Morgan [1998].

2.1 Basic Definitions

Suppose that $\Omega \subset \mathbb{R}^2$ is some simply connected region (i.e., without any holes), for example,

$$\begin{aligned} \text{the } \textit{unit square}: &\quad \Omega = \{(u,v) \in \mathbb{R}^2 : u, v \in [0,1]\}, \quad \text{or} \\ \text{the } \textit{unit disk}: &\quad \Omega = \{(u,v) \in \mathbb{R}^2 : u^2 + v^2 \leq 1\}, \end{aligned}$$

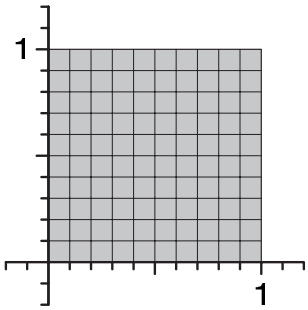
and that the function $f : \Omega \to \mathbb{R}^3$ is *continuous* and an *injection* (i.e., no two distinct points in Ω are mapped to the same point in $\mathbb{R}^3$). We then call the image S of Ω under f a *surface*,

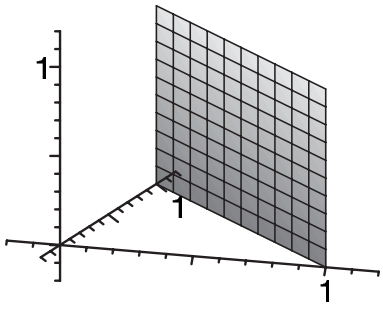
$$S = f(\Omega) = \{f(u,v) : (u,v) \in \Omega\},$$

and say that f is a *parameterization* of S over the *parameter domain* Ω. It follows from the definition of S that f is actually a *bijection* between Ω and S and thus admits to define its inverse $f^{-1} : S \to \Omega$. Here are some examples:

1. simple linear function:

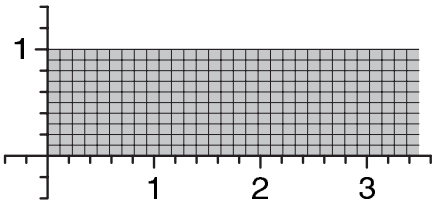
$$\begin{aligned} \textit{parameter domain:} &\quad \Omega = \{(u,v) \in \mathbb{R}^2 : u, v \in [0,1]\} \\ \textit{surface:} &\quad S = \{(x,y,z) \in \mathbb{R}^3 : x, y, z \in [0,1], x + y = 1\} \\ \textit{parameterization:} &\quad f(u,v) = (u, 1-u, v) \\ \textit{inverse:} &\quad f^{-1}(x,y,z) = (x,z) \end{aligned}$$

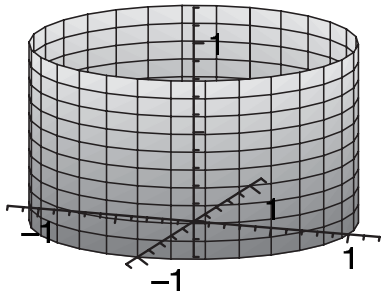




2. cylinder:

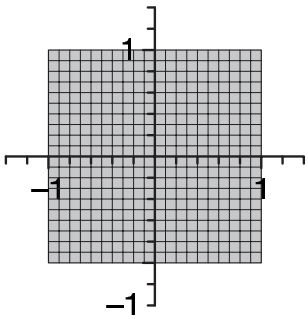
$$\begin{aligned} \textit{parameter domain:} \quad & \Omega = \{(u,v) \in \mathbb{R}^2 : u \in [0, 2\pi), v \in [0,1]\} \\ \textit{surface:} \quad & S = \{(x,y,z) \in \mathbb{R}^3 : x^2 + y^2 = 1, z \in [0,1]\} \\ \textit{parameterization:} \quad & f(u,v) = (\cos u, \sin u, v) \\ \textit{inverse:} \quad & f^{-1}(x,y,z) = (\arccos x, z) \end{aligned}$$

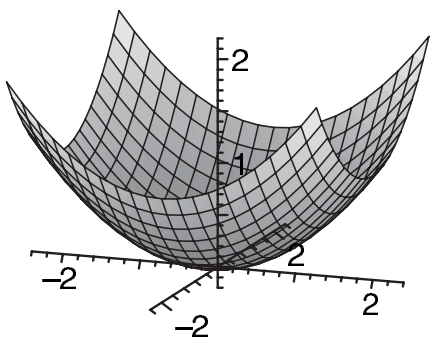




3. paraboloid:

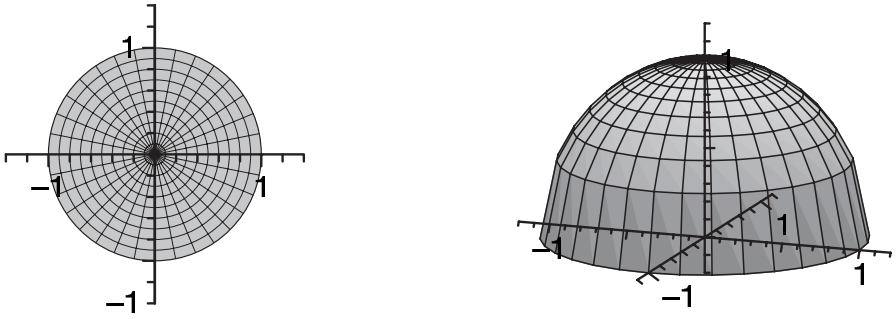
$$\begin{aligned} \textit{parameter domain:} \quad & \Omega = \{(u,v) \in \mathbb{R}^2 : u, v \in [-1,1]\} \\ \textit{surface:} \quad & S = \{(x,y,z) \in \mathbb{R}^3 : x, y \in [-2,2], z = \tfrac{1}{4}(x^2 + y^2)\} \\ \textit{parameterization:} \quad & f(u,v) = (2u, 2v, u^2 + v^2) \\ \textit{inverse:} \quad & f^{-1}(x,y,z) = (\tfrac{x}{2}, \tfrac{y}{2}) \end{aligned}$$





4. hemisphere (orthographic):

$$\begin{aligned} \textit{parameter domain:} &\quad \Omega = \{(u,v) \in \mathbb{R}^2 : u^2 + v^2 \leq 1\} \\ \textit{surface:} &\quad S = \{(x,y,z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 = 1, z \geq 0\} \\ \textit{parameterization:} &\quad f(u,v) = (u, v, \sqrt{1-u^2-v^2}) \\ \textit{inverse:} &\quad f^{-1}(x,y,z) = (x,y) \end{aligned}$$



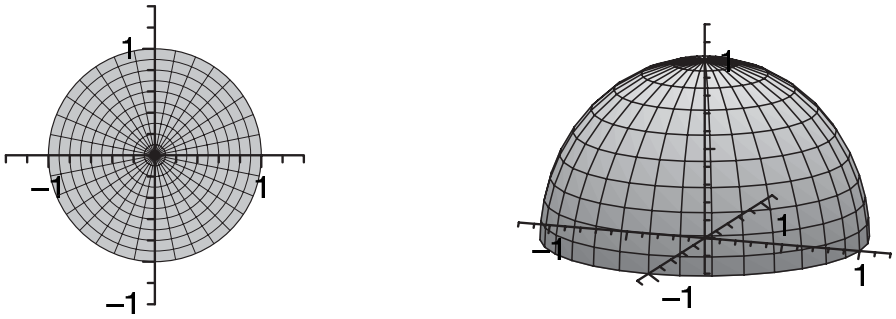
Having defined a surface S like that, we should note that the function f is by no means the only parameterization of S over Ω. In fact, given any bijection $\varphi : \Omega \to \Omega$, it is easy to verify that the *composition* of f and φ, i.e., the function $g = f \circ \varphi$, is a parameterization of S over Ω, too. For example, we can easily construct such a *reparameterization* φ from any bijection $\rho : [0,1] \to [0,1]$ by defining

$$\begin{aligned} \text{for the unit square:} &\quad \varphi(u,v) = (\rho(u), \rho(v)), \quad \text{or} \\ \text{for the unit disk:} &\quad \varphi(u,v) = (u\rho(u^2+v^2), v\rho(u^2+v^2)). \end{aligned}$$

In particular, taking the function $\rho(x) = \frac{2}{1+x}$ and applying this reparameterization of the unit disk to the parameterization of the hemisphere in the example above gives the following alternative parameterization:

5. hemisphere (stereographic):

$$\begin{aligned} \textit{parameter domain:} &\quad \Omega = \{(u,v) \in \mathbb{R}^2 : u^2 + v^2 \leq 1\} \\ \textit{surface:} &\quad S = \{(x,y,z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 = 1, z \geq 0\} \\ \textit{parameterization:} &\quad f(u,v) = (\tfrac{2u}{1+u^2+v^2}, \tfrac{2v}{1+u^2+v^2}, \tfrac{1-u^2-v^2}{1+u^2+v^2}) \\ \textit{inverse:} &\quad f^{-1}(x,y,z) = (\tfrac{x}{1+z}, \tfrac{y}{1+z}) \end{aligned}$$



2.2 Intrinsic Surface Properties

Although the parameterization of a surface is not unique—and we will later discuss how to get the "best" parameterization with respect to certain criteria—it nevertheless is a very handy thing to have as it allows to compute a variety of properties of the surface. For example, if f is differentiable, then its *partial derivatives*

$$f_u = \frac{\partial f}{\partial u} \qquad \text{and} \qquad f_v = \frac{\partial f}{\partial v}$$

span the local *tangent plane* and by simply taking their cross product and normalizing the result we get the *surface normal*

$$n_f = \frac{f_u \times f_v}{\|f_u \times f_v\|}.$$

To simplify the notation, we will often speak of f_u and f_v as *the* derivatives and of n_f as *the* surface normal, but we should keep in mind that formally all three are functions from $\mathbb{R}^2$ to $\mathbb{R}^3$. In other words, for any point $(u, v) \in \Omega$ in the parameter domain, the tangent plane at the surface point $f(u, v) \in S$ is spanned by the two vectors $f_u(u, v)$ and $f_v(u, v)$, and $n_f(u, v)$ is the normal vector at this point[1]. Again, let us clarify this by considering two examples:

1. For the simple linear function $f(u, v) = (u, 1 - u, v)$ we get

   $$f_u(u, v) = (1, -1, 0) \qquad \text{and} \qquad f_v(u, v) = (0, 0, 1)$$

   and further

   $$n_f(u, v) = (\tfrac{-1}{\sqrt{2}}, \tfrac{1}{\sqrt{2}}, 0),$$

   showing that the normal vector is constant for all points on S.

2. For the parameterization of the cylinder, $f(u, v) = (\cos u, \sin u, v)$, we get

   $$f_u(u, v) = (-\sin u, \cos u, 0) \qquad \text{and} \qquad f_v(u, v) = (0, 0, 1)$$

   and further

   $$n_f(u, v) = (\cos u, \sin u, 0),$$

   showing that the normal vector at any point $(x, y, z) \in S$ is just $(x, y, 0)$.

Note that in both examples the surface normal is *independent* of the parameterization. In fact, this holds for all surfaces and is therefore called an *intrinsic* property of the surface. Formally, we can also say that the surface normal is a function $n : S \to \mathbb{S}^2$, where $\mathbb{S}^2 = \{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 = 1\}$ is the unit sphere in $\mathbb{R}^3$, so that

$$n(\boldsymbol{p}) = n_f(f^{-1}(\boldsymbol{p}))$$

[1] We tacitly assume that the parameterization is *regular*, i.e., f_u and f_v are always linearly independent and therefore n_f is non-zero.

for any $\boldsymbol{p} \in S$ and any parameterization f. As an exercise, you may want to verify this for the two alternative parameterizations of the hemisphere given above. Other intrinsic surface properties are the *Gaussian curvature* $K(\boldsymbol{p})$ and the *mean curvature* $H(\boldsymbol{p})$ as well as the total *area* of the surface $A(S)$. To compute the latter, we need the *first fundamental form*

$$\mathbf{I}_f = \begin{pmatrix} f_u \cdot f_u & f_u \cdot f_v \\ f_v \cdot f_u & f_v \cdot f_v \end{pmatrix} = \begin{pmatrix} E & F \\ F & G \end{pmatrix},$$

where the product between the partial derivatives is the usual dot product in $\mathbb{R}^3$. It follows immediately from the Cauchy-Schwarz inequality that the determinant of this symmetric 2×2 matrix is always non-negative, so that its square root is always real. The area of the surface is then defined as

$$A(S) = \int_\Omega \sqrt{\det \mathbf{I}_f}\, du\, dv.$$

Take, for example, the orthographic parameterization $f(u, v) = (u, v, \sqrt{1 - u^2 - v^2})$ of the hemisphere over the unit disk. After some simplifications we find that

$$\det \mathbf{I}_f = \frac{1}{1 - u^2 - v^2}$$

and can compute the area of the hemisphere as follows:

$$\begin{aligned} A(S) &= \int_{-1}^{1} \int_{-\sqrt{1-v^2}}^{\sqrt{1-v^2}} \frac{1}{\sqrt{1 - u^2 - v^2}}\, du\, dv \\ &= \int_{-1}^{1} \left[\arcsin \frac{u}{\sqrt{1 - v^2}} \right]_{-\sqrt{1-v^2}}^{\sqrt{1-v^2}} dv \\ &= \int_{-1}^{1} \pi\, dv \\ &= 2\pi, \end{aligned}$$

as expected. Of course we get the same result if we use the stereographic parameterization, and you may want to try that as an exercise.

In order to compute the curvatures we must first assume the parameterization to be twice differentiable, so that its *second order* partial derivatives

$$f_{uu} = \frac{\partial^2 f}{\partial u^2}, \qquad f_{uv} = \frac{\partial^2 f}{\partial u \partial v}, \qquad \text{and} \qquad f_{vv} = \frac{\partial^2 f}{\partial v^2}$$

are well defined. Taking the dot products of these derivatives with the surface normal then gives the symmetric 2×2 matrix that is known as the *second fundamental form*

$$\mathbf{II}_f = \begin{pmatrix} f_{uu} \cdot n_f & f_{uv} \cdot n_f \\ f_{uv} \cdot n_f & f_{vv} \cdot n_f \end{pmatrix} = \begin{pmatrix} L & M \\ M & N \end{pmatrix}.$$

Gaussian and mean curvature are finally defined as the determinant and half the trace of the matrix $\mathbf{I}_f^{-1}\mathbf{II}_f$, respectively:

$$K = \det(\mathbf{I}_f^{-1}\mathbf{II}_f) = \frac{\det \mathbf{II}_f}{\det \mathbf{I}_f} = \frac{LN - M^2}{EG - F^2}$$

and

$$H = \tfrac{1}{2}\operatorname{trace}(\mathbf{I}_f^{-1}\mathbf{II}_f) = \frac{LG - 2MF + NE}{2(EG - F^2)}.$$

For example, carrying out these computations reveals that the curvatures are constant for most of the surfaces from above:

$$\begin{aligned} \text{simple linear function:} \quad & K = 0, \quad H = 0, \\ \text{cylinder:} \quad & K = 0, \quad H = \tfrac{1}{2}, \\ \text{hemisphere:} \quad & K = 1, \quad H = -1. \end{aligned}$$

As an exercise, show that the curvatures at any point $\boldsymbol{p} = (x, y, z)$ of the paraboloid from above are $K(\boldsymbol{p}) = \frac{1}{4(1+z)^2}$ and $H(\boldsymbol{p}) = \frac{2+z}{4(1+z)^{3/2}}$.

2.3 Metric Distortion

Apart from these intrinsic surface properties, there are others that depend on the parameterization, most importantly the *metric distortion*. Consider, for example, the two parameterizations of the hemisphere above. In both cases, the image of the surface on the right is overlaid by a regular grid, which actually is the image of the corresponding grid in the parameter domain shown on the left. You will notice that the surface grid looks more regular for the stereographic than for the orthographic projection and that the latter considerably stretches the grid in the radial direction near the boundary.

To better understand this kind of stretching, let us see what happens to the surface point $f(u, v)$ as we move a tiny little bit away from (u, v) in the parameter domain. If we denote this infinitesimal parameter displacement by $(\Delta u, \Delta v)$, then the new surface point $f(u + \Delta u, v + \Delta v)$ is approximately given by the first order Taylor expansion $\tilde{f}$ of f around (u, v),

$$\tilde{f}(u + \Delta u, v + \Delta v) = f(u, v) + f_u(u, v)\Delta u + f_v(u, v)\Delta v.$$

This linear function maps all points in the vicinity of $\boldsymbol{u} = (u, v)$ into the tangent plane $T_{\boldsymbol{p}}$ at $\boldsymbol{p} = f(u, v) \in S$ and transforms circles around $\boldsymbol{u}$ into ellipses around $\boldsymbol{p}$ (see Figure 2.1). The latter property becomes obvious if we write the Taylor expansion more compactly as

$$\tilde{f}(u + \Delta u, v + \Delta v) = \boldsymbol{p} + J_f(\boldsymbol{u})\tbinom{\Delta u}{\Delta v},$$

where $J_f = (f_u \; f_v)$ is the *Jacobian* of f, i.e. the 3×2 matrix with the partial derivatives of f as column vectors. Then using the *singular value decomposition* of the Jacobian,

$$J_f = U\Sigma V^T = U \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{pmatrix} V^T,$$

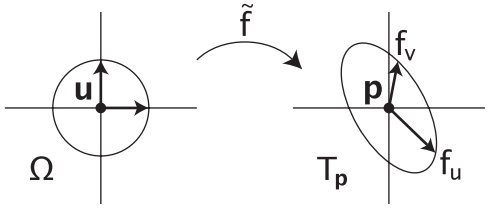


Figure 2.1: First order Taylor expansion $\tilde{f}$ of the parameterization f.

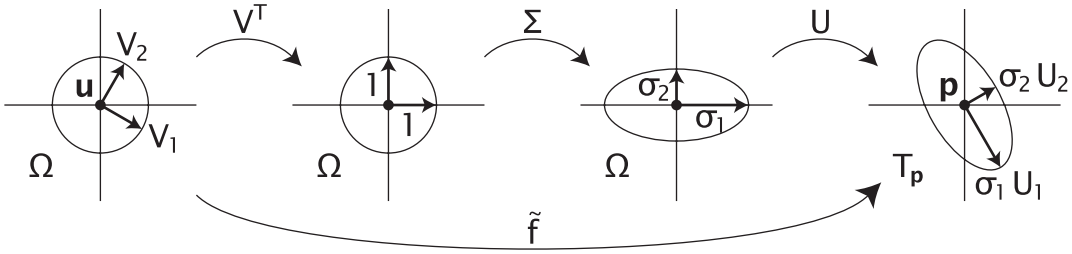


Figure 2.2: SVD decomposition of the mapping $\tilde{f}$.

with *singular values* $\sigma_1 \geq \sigma_2 > 0$ and *orthonormal* matrices $U \in \mathbb{R}^{3\times 3}$ and $V \in \mathbb{R}^{2\times 2}$ with column vectors U_1, U_2, U_3, and V_1, V_2, respectively, we can split up the linear transformation $\tilde{f}$ as shown in Figure 2.2:

1. The transformation V^T first rotates all points around $\boldsymbol{u}$ such that the vectors V_1 and V_2 are in alignment with the u- and the v-axes afterwards.

2. The transformation Σ then stretches everything by the factor σ_1 in the u- and by σ_2 in the v-direction.

3. The transformation U finally maps the unit vectors $(1, 0)$ and $(0, 1)$ to the vectors U_1 and U_2 in the tangent plane $T_{\boldsymbol{p}}$ at $\boldsymbol{p}$.

As a consequence, any circle of radius r around $\boldsymbol{u}$ will be mapped to an ellipse with semi-axes of length $r\sigma_1$ and $r\sigma_2$ around $\boldsymbol{p}$ and the orthonormal frame $[V_1, V_2]$ is mapped to the orthogonal frame $[\sigma_1 U_1, \sigma_2 U_2]$.

This transformation of circles into ellipses is called *local metric distortion* of the parameterization as it shows how f behaves locally around some parameter point $\boldsymbol{u} \in \Omega$ and the corresponding surface point $p = f(\boldsymbol{u}) \in S$. Moreover, all information about this local metric distortion is hidden in the singular values σ_1 and σ_2. For example, if both values are identical, then J_f is just a rotation plus uniform scaling and f does not distort angles around $\boldsymbol{u}$. Likewise, if the product of the singular values is 1, then the area of any circle in the parameter domain is identical to the area of the corresponding ellipse in the tangent plane and we say that f is locally area-preserving.

Computing the singular values directly is a bit tedious, so that we better resort to the fact that the singular values of any matrix A are the square roots of the *eigenvalues* of the matrix $A^T A$. In our case, the matrix ${J_f}^T J_f$ is an old acquaintance, namely the

first fundamental form,

$$J_f{}^T J_f = \begin{pmatrix} f_u{}^T \\ f_v{}^T \end{pmatrix} (f_u \; f_v) = \mathbf{I}_f = \begin{pmatrix} E & F \\ F & G \end{pmatrix},$$

and we can easily compute the two eigenvalues λ_1 and λ_2 of this symmetric matrix by using the nifty little formula

$$\lambda_{1,2} = \tfrac{1}{2}\big((E+G) \pm \sqrt{4F^2 + (E-G)^2}\big).$$

We now summarize the main properties that a parameterization can have locally:

$$\begin{aligned} f \text{ is } \textit{isometric} \text{ or } \textit{length-preserving} &\iff \sigma_1 = \sigma_2 = 1 \iff \lambda_1 = \lambda_2 = 1, \\ f \text{ is } \textit{conformal} \text{ or } \textit{angle-preserving} &\iff \sigma_1 = \sigma_2 \iff \lambda_1 = \lambda_2, \\ f \text{ is } \textit{equiareal} \text{ or } \textit{area-preserving} &\iff \sigma_1\sigma_2 = 1 \iff \lambda_1\lambda_2 = 1. \end{aligned}$$

Obviously, any isometric mapping is conformal and equiareal, and every mapping that is conformal and equiareal is also isometric, in short,

$$\text{isometric} \iff \text{conformal} + \text{equiareal}.$$

Thus equipped, let us go back to the examples above and check their properties:

1. simple linear function:

$$\begin{aligned} \textit{parameterization:} \quad & f(u,v) = (u, 1-u, v) \\ \textit{Jacobian:} \quad & J_f = \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \end{pmatrix} \\ \textit{first fundamental form:} \quad & \mathbf{I}_f = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \\ \textit{eigenvalues:} \quad & \lambda_1 = 2, \quad \lambda_2 = 1 \end{aligned}$$

   This parameterization is neither conformal nor equiareal.

2. cylinder:

$$\begin{aligned} \textit{parameterization:} \quad & f(u,v) = (\cos u, \sin u, v) \\ \textit{Jacobian:} \quad & J_f = \begin{pmatrix} \cos u & 0 \\ -\sin u & 0 \\ 0 & 1 \end{pmatrix} \\ \textit{first fundamental form:} \quad & \mathbf{I}_f = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ \textit{eigenvalues:} \quad & \lambda_1 = 1, \quad \lambda_2 = 1 \end{aligned}$$

   This parameterization is isometric.

3. paraboloid:

$$\begin{aligned} \textit{parameterization:} \quad & f(u,v) = (2u, 2v, u^2+v^2) \\ \textit{Jacobian:} \quad & J_f = \begin{pmatrix} 2 & 0 \\ 0 & 2 \\ 2u & 2v \end{pmatrix} \\ \textit{first fundamental form:} \quad & \mathbf{I}_f = \begin{pmatrix} 4+4u^2 & 4uv \\ 4uv & 4+4v^2 \end{pmatrix} \\ \textit{eigenvalues:} \quad & \lambda_1 = 4, \quad \lambda_2 = 4(1+u^2+v^2) \end{aligned}$$

This mapping is not equiareal and conformal only at $(u, v) = (0, 0)$.

4. hemisphere (orthographic):

$$\begin{aligned} \textit{parameterization:} \quad & f(u,v) = (u, v, \tfrac{1}{d}) \quad \textit{with} \quad d = \tfrac{1}{\sqrt{1-u^2-v^2}} \\ \textit{Jacobian:} \quad & J_f = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -ud & -vd \end{pmatrix} \\ \textit{first fundamental form:} \quad & \mathbf{I}_f = \begin{pmatrix} 1+u^2d^2 & uvd^2 \\ uvd^2 & 1+v^2d^2 \end{pmatrix} \\ \textit{eigenvalues:} \quad & \lambda_1 = 1, \quad \lambda_2 = d^2 \end{aligned}$$

This mapping is isometric at $(u, v) = (0, 0)$, but neither conformal nor equiareal elsewhere.

5. hemisphere (stereographic):

$$\begin{aligned} \textit{parameterization:} \quad & f(u,v) = (2ud, 2vd, (1-u^2-v^2)d) \quad \textit{with} \quad d = \tfrac{1}{1+u^2+v^2} \\ \textit{Jacobian:} \quad & J_f = \begin{pmatrix} 2d-4u^2d^2 & -4uvd^2 \\ -4uvd^2 & 2d-4v^2d^2 \\ -4ud^2 & -4vd^2 \end{pmatrix} \\ \textit{first fundamental form:} \quad & \mathbf{I}_f = \begin{pmatrix} 4d^2 & 0 \\ 0 & 4d^2 \end{pmatrix} \\ \textit{eigenvalues:} \quad & \lambda_1 = 4d^2, \quad \lambda_2 = 4d^2 \end{aligned}$$

This mapping is always conformal, but equiareal and thus isometric only at the boundary of Ω, i.e., for $u^2 + v^2 = 1$.

It turns out that the only parameterization that is optimal in the sense that it is isometric everywhere and thus does not introduce any distortion at all is the one for the cylinder. In fact, it was shown by Gauß [1827] that a globally isometric parameterization exists only for *developable* surfaces like planes, cones, and cylinders with vanishing Gaussian curvature $K(\boldsymbol{p}) = 0$ at all surface points $\boldsymbol{p} \in S$. As an exercise, you can try to find such a globally isometric parameterization for the planar surface patch from the first example.

Other interesting parameterizations are those that are globally conformal like the stereographic projection for the hemisphere, and it was shown by Riemann [1851] that

such a parameterization exists for any surface that is topologically equivalent to a disk and any simply connected parameter domain.

More generally, the "best" parameterization f of a surface S over a parameter domain Ω is found as follows. We first need a bivariate non-negative function $E : \mathbb{R}_+^2 \to \mathbb{R}_+$ that measures the local distortion of a parameterization with singular values σ_1 and σ_2. Usually, this function has a global minimum at $(1, 1)$ so as to favour isometry, but depending on the application, it may also be defined such that the minimal value is taken along the whole line (x, x) for $x \in \mathbb{R}_+$, for example, if conformal mappings shall be preferred. The overall distortion of a particular parameterization f is then measured by simply averaging the local distortion over the whole domain,

$$E(f) = \int_\Omega E(\sigma_1(u, v), \sigma_2(u, v))\, du\, dv \Big/ A(\Omega),$$

and the best parameterization with respect to E is then found by minimizing $E(f)$ over the space of all admissible parameterizations.

Chapter 3

Barycentric Mappings

In many applications, and in particular in computer graphics, it is nowadays common to work with piecewise linear surfaces in the form of triangle meshes, and we will mainly stick to this type of surface for the remainder of these course notes.

3.1 Triangle Meshes

As in the previous chapter, let us denote points in $\mathbb{R}^3$ by $\boldsymbol{p} = (x, y, z)$ and points in $\mathbb{R}^2$ by $\boldsymbol{u} = (u, v)$. An *edge* is then defined as the convex hull of (or, equivalently, the line segment between) two distinct points and a *triangle* as the convex hull of three non-collinear points. We will denote edges and triangles in $\mathbb{R}^3$ with capital letters and those in $\mathbb{R}^2$ with small letters, for example, $e = [\boldsymbol{u}_1, \boldsymbol{u}_2]$ and $T = [\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3]$.

A *triangle mesh* $S_{\mathcal{T}}$ is the union of a set of *surface triangles* $\mathcal{T} = \{T_1, \ldots, T_m\}$ which intersect only at common edges $\mathcal{E} = \{E_1, \ldots, E_l\}$ and vertices $\mathcal{V} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{n+b}\}$. More specifically, the set of vertices consists of n *interior* vertices $\mathcal{V}_I = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ and b *boundary* vertices $\mathcal{V}_B = \{\boldsymbol{p}_{n+1}, \ldots, \boldsymbol{p}_{n+b}\}$. Two distinct vertices $\boldsymbol{p}_i, \boldsymbol{p}_j \in \mathcal{V}$ are called *neighbours*, if they are the end points of some edge $E = [\boldsymbol{p}_i, \boldsymbol{p}_j] \in \mathcal{E}$, and for any $\boldsymbol{p}_i \in \mathcal{V}$ we let $N_i = \{j : [\boldsymbol{p}_i, \boldsymbol{p}_j] \in \mathcal{E}\}$ be the set of indices of all neighbours of $\boldsymbol{p}_i$.

A parameterization f of $S_{\mathcal{T}}$ is usually specified the other way around, that is, by defining the inverse parameterization $g = f^{-1}$. This mapping g is uniquely determined by specifying the *parameter points* $\boldsymbol{u}_i = g(\boldsymbol{p}_i)$ for each vertex $\boldsymbol{p}_i \in \mathcal{V}$ and demanding that g is continuous and linear for each triangle. In this setting, $g|_T$ is the linear map from a surface triangle $T = [\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_k]$ to the corresponding *parameter triangle* $t = [\boldsymbol{u}_i, \boldsymbol{u}_j, \boldsymbol{u}_k]$ and $f|_t = (g|_T)^{-1}$ is the inverse linear map from t to T. The parameter domain Ω finally is the union of all parameter triangles (see Figure 3.1).

3.2 Parameterization by Affine Combinations

A rather simple idea for constructing a parameterization of a triangle mesh is based on the following physical model. Imagine that the edges of the triangle mesh are springs that are connected at the vertices. If we now fix the boundary of this spring network somewhere in the plane, then the interior of this network will relax in the energetically

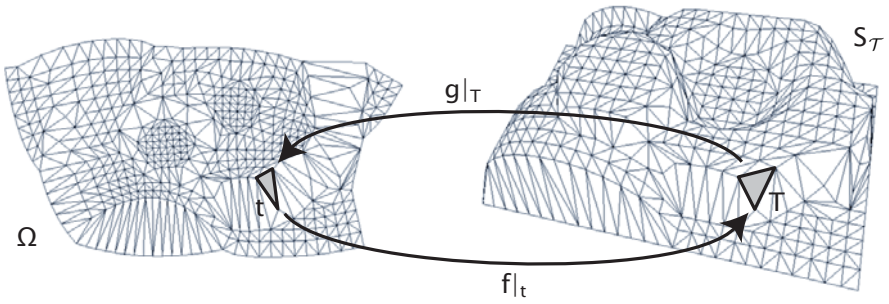


Figure 3.1: Parameterization of a triangle mesh.

most efficient configuration, and we can simply assign the positions where the joints of the network have come to rest as parameter points.

If we assume each spring to be ideal in the sense that the rest length is zero and the potential energy is just $\frac{1}{2}Ds^2$, where D is the spring constant and s the length of the spring, then we can formalize this approach as follows. We first specify the parameter points $\boldsymbol{u}_i = (u_i, v_i)$, $i = n+1, \ldots, n+b$ for the boundary vertices $\boldsymbol{p}_i \in \mathcal{V}_B$ of the mesh in some way (see Section 3.4). Then we minimize the overall spring energy

$$E = \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j \in N_i} \tfrac{1}{2} D_{ij} \|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2,$$

where $D_{ij} = D_{ji}$ is the spring constant of the spring between $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, with respect to the unknown parameter positions $\boldsymbol{u}_i = (u_i, v_i)$ for the interior points[1]. As the partial derivative of E with respect to $\boldsymbol{u}_i$ is

$$\frac{\partial E}{\partial \boldsymbol{u}_i} = \sum_{j \in N_i} D_{ij}(\boldsymbol{u}_i - \boldsymbol{u}_j),$$

the minimum of E is obtained if

$$\sum_{j \in N_i} D_{ij} \boldsymbol{u}_i = \sum_{j \in N_i} D_{ij} \boldsymbol{u}_j$$

holds for all $i = 1, \ldots, n$. This is equivalent to saying that each interior parameter point $\boldsymbol{u}_i$ is an *affine combination* of its neighbours,

$$\boldsymbol{u}_i = \sum_{j \in N_i} \lambda_{ij} \boldsymbol{u}_j, \tag{3.1}$$

with normalized coefficients

$$\lambda_{ij} = D_{ij} \Big/ \sum_{k \in N_i} D_{ik}$$

that obviously sum to 1.

[1]The additional factor $\frac{1}{2}$ appears because summing up the edges in this way counts every edge twice.

By separating the parameter points for the interior and the boundary vertices in the sum on the right hand side of (3.1) we get

$$\boldsymbol{u}_i - \sum_{j \in N_i, j \le n} \lambda_{ij} \boldsymbol{u}_j = \sum_{j \in N_i, j > n} \lambda_{ij} \boldsymbol{u}_j,$$

and see that computing the coordinates u_i and v_i of the interior parameter points $\boldsymbol{u}_i$ requires to solve the linear systems

$$AU = \bar{U} \qquad \text{and} \qquad AV = \bar{V}, \tag{3.2}$$

where $U = (u_1, \ldots, u_n)$ and $V = (v_1, \ldots, v_n)$ are the column vectors of unknown coordinates, $\bar{U} = (\bar{u}_1, \ldots, \bar{u}_n)$ and $\bar{V} = (\bar{v}_1, \ldots, \bar{v}_n)$ are the column vectors with coefficients

$$\bar{u}_i = \sum_{j \in N_i, j > n} \lambda_{ij} u_j \qquad \text{and} \qquad \bar{v}_i = \sum_{j \in N_i, j > n} \lambda_{ij} v_j$$

and $A = (a_{ij})_{i,j=1,\ldots,n}$ is the $n \times n$ matrix with elements

$$a_{ij} = \begin{cases} 1 & \text{if } i = j, \\ -\lambda_{ij} & \text{if } j \in N_i, \\ 0 & \text{otherwise.} \end{cases}$$

Methods for efficiently solving these systems are described in Chapter 10 of these course notes.

3.3 Barycentric Coordinates

The question remains how to choose the spring constants D_{ij} in the spring model, or more generally, the normalized coefficients λ_{ij} in (3.1). The simplest choice of constant spring constants $D_{ij} = 1$ goes back to the work of Tutte [1960, 1963] who used it in a more abstract graph-theoretic setting to compute straight line embeddings of planar graphs, and the idea of taking spring constants that are proportional to the lengths of the corresponding edges in the triangle mesh was used by Greiner and Hormann [1997]. A main drawback of both approaches is that they do not fulfill the following minimum requirement that we should expect from any parameterization method.

Linear reproduction: Suppose that $S_{\mathcal{T}}$ is contained in a plane so that its vertices have coordinates $\boldsymbol{p}_i = (x_i, y_i, 0)$ with respect to some appropriately chosen orthonormal coordinate frame. Then a globally isometric (and thus optimal) parameterization can be defined by just using the local coordinates $\boldsymbol{x}_i = (x_i, y_i)$ as parameter points themselves, that is, by setting $\boldsymbol{u}_i = \boldsymbol{x}_i$ for $i = 1, \ldots, n + b$. As the overall parameterization then is a linear function, we say that a parameterization method has *linear reproduction* if it produces such an isometric mapping in this setting.

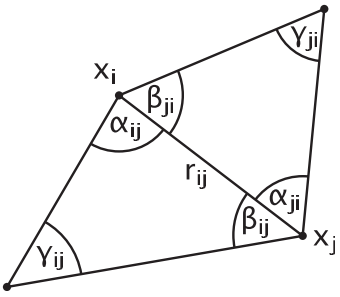


Figure 3.2: Notation for the construction of barycentric coordinates.

In the setting from the previous section, linear reproduction can be achieved if the parameter points for the boundary vertices are set correctly and the values λ_{ij} are chosen such that

$$\boldsymbol{x}_i = \sum_{j \in N_i} \lambda_{ij} \boldsymbol{x}_j \qquad \text{and} \qquad \sum_{j \in N_i} \lambda_{ij} = 1$$

for all interior vertices. Values λ_{ij} with both these properties are also called *barycentric coordinates* of $\boldsymbol{x}_i$ with respect to its neighbours $\boldsymbol{x}_j$, $j \in N_i$. If some $\boldsymbol{x}_i$ has exactly three neighbours, then the λ_{ij} are uniquely defined and these barycentric coordinates inside triangles actually have many useful applications in computer graphics (e.g., Gouraud and Phong shading, ray-triangle-intersection), geometric modelling (e.g., triangular Bézier patches, splines over triangulations), and many other fields (e.g., the finite element method, terrain modelling).

For polygons with more than three vertices, the barycentric coordinates of a point in the interior are, however, not unique anymore and there are several ways of defining them. The most popular of them can all be described in a common framework [Floater et al., 2006] that we shall briefly review. For any interior point $\boldsymbol{x}_i$ and one of its neighbours $\boldsymbol{x}_j$ let $r_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ be the length of the edge $e_{ij} = [\boldsymbol{x}_i, \boldsymbol{x}_j]$ between the two points and let the angles at the corners of the triangles adjacent to e_{ij} be denoted as shown in Figure 3.2. The barycentric coordinates λ_{ij} of $\boldsymbol{x}_i$ with respect its neighbours $\boldsymbol{x}_j$, $j \in N_i$ can then be computed by the normalization $\lambda_{ij} = w_{ij} / \sum_{k \in N_i} w_{ik}$ from any of the following *homogeneous coordinates* w_{ij}.

- *Wachspress coordinates*: The earliest generalization of barycentric coordinates goes back to Wachspress [1975] who suggested to set

  $$w_{ij} = \frac{\cot \alpha_{ji} + \cot \beta_{ij}}{{r_{ij}}^2}.$$

  While he was mainly interested in applying these coordinates in finite element methods, Desbrun et al. [2002] used them for parameterizing triangle meshes and Meyer et al. [2002] for interpolating e.g. colour values inside convex polygons. Moreover, a simple geometric construction of these coordinates was given by Ju et al. [2005b].

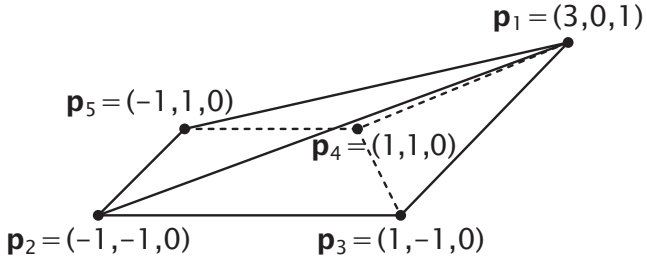


Figure 3.3: Example of a triangle mesh for which only the barycentric mapping with mean value coordinates is a bijection.

- *Discrete harmonic coordinates*: Another type of barycentric coordinates that stem from finite element methods and actually arise from the standard piecewise linear approximation to the Laplace equation are given by

$$w_{ij} = \cot \gamma_{ij} + \cot \gamma_{ji}.$$

  In the context of mesh parameterization, these coordinates were first used by Eck et al. [1995], but they have also been used to compute discrete minimal surfaces [Pinkall and Polthier, 1993].

- *Mean value coordinates*: By discretizing the mean value theorem, Floater [2003a] found yet another set of barycentric coordinates with

$$w_{ij} = \frac{\tan \frac{\alpha_{ij}}{2} + \tan \frac{\beta_{ji}}{2}}{r_{ij}}.$$

  While his main application was mesh parameterization, Hormann and Tarini [2004] and Hormann and Floater [2006] later showed that they have many other useful applications, in particular in computer graphics.

The beauty of all three choices is that the weights w_{ij} depend on angles and distances only, so that they can not only be computed if $\boldsymbol{x}_i$ and its neighbours are coplanar, but more generally for any interior vertex $\boldsymbol{p}_i \in \mathcal{V}_I$ of a triangle mesh if these angles and distances are just taken from the triangles around $\boldsymbol{p}_i$. Of course, an alternative approach that was introduced by Floater [1997] is to locally flatten the one-ring of triangles around $\boldsymbol{p}_i$ into the plane, e.g. with an exponential map, and then to compute the weights w_{ij} from this planar configuration.

A triangle mesh parameterization that is computed by solving the linear systems (3.2) with any set of barycentric coordinates λ_{ij} is called a *barycentric mapping* and obviously has the linear reproduction property, provided that an appropriate method for computing the parameter points for the boundary vertices, e.g. mapping them to the least squares plane (see Section 3.4), is used.

Despite this property, it may happen that a barycentric mapping, when constructed for a non-planar mesh, gives an unexpected result, as the simple example in Figure 3.3

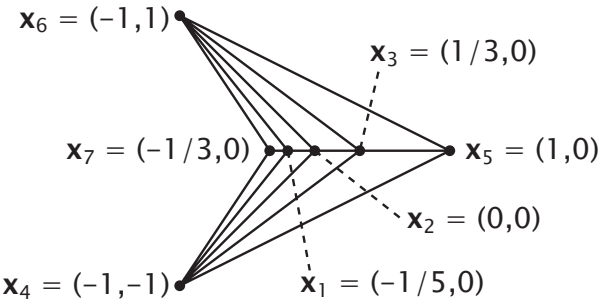


Figure 3.4: Example of a triangle mesh for which the linear system with Wachspress coordinates is singular.

illustrates. If we use $\boldsymbol{u}_2 = (-1,-1)$, $\boldsymbol{u}_3 = (1,-1)$, $\boldsymbol{u}_4 = (1,1)$, $\boldsymbol{u}_5 = (-1,1)$ as parameter points for the four boundary vertices and compute the barycentric weights λ_{12}, λ_{13}, λ_{14}, λ_{15} with the formulas described above, then we get the following positions for $\boldsymbol{u}_1$:

$$\begin{aligned} \text{Wachspress coordinates:} \quad & \boldsymbol{u}_1 = (-35.1369, 0), \\ \text{discrete harmonic coordinates:} \quad & \boldsymbol{u}_1 = (2.1138, 0), \\ \text{mean value coordinates:} \quad & \boldsymbol{u}_1 = (0.4538, 0). \end{aligned}$$

That is, only the mean value coordinates yield a position for $\boldsymbol{u}_1$ that is contained in the convex hull of the other four parameter points, and using the other coordinates will create parameter triangles that overlap, thus violating the bijectivity property that any parameterization should have.

The reason behind this behaviour is that the Wachspress and discrete harmonic coordinates can assume *negative* values in certain configurations like the one in Figure 3.3, whereas the mean values coordinates are always *positive*. And while overlapping triangles may occur for negative weights, this never happens if all weights are positive *and* the parameter points of the boundary vertices form a convex shape. The latter fact has first been proven by Tutte [1963] for the special case of $\lambda_{ij} = 1/\eta_i$ where $\eta_i = \#N_i$ is the number of $\boldsymbol{p}_i$'s neighbours, which are not true barycentric coordinates, but Floater [1997] observed that the proof carries over to arbitrary positive weights λ_{ij}. Recently, Gortler et al. [2006] could even show that the restriction to a convex boundary can be considerably relaxed, but this requires to solve a non-linear problem.

Another important aspect concerns the solvability of the linear systems (3.2) and it has been shown that the matrix A is always guaranteed to be non-singular for discrete harmonic [Pinkall and Polthier, 1993] and mean value coordinates [Floater, 1997]. For Wachspress coordinates, however, it may happen that the sum of homogeneous coordinates $W_i = \sum_{k \in N_i} w_{ik}$ is zero so that the normalized coordinates λ_{ij} and thus the matrix A are not even well-defined. In the example shown in Figure 3.4 this actually happens for all interior vertices $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$. But even if we skip the normalization and try to solve the equivalent and well-defined homogeneous systems $WAU = W\bar{U}$ and $WAV = W\bar{V}$ with $W = \operatorname{diag}(W_1, \ldots, W_n)$ instead, we find that the matrix WA is singular in this particular example, namely $WA = \begin{pmatrix} 0 & -50 & 0 \\ 40 & 0 & -24 \\ 0 & 18 & 0 \end{pmatrix}$.

3.4 The Boundary Mapping

The first step in constructing a barycentric mapping is to choose the parameter points for the boundary vertices and the simplest way of doing it is to just project the boundary vertices into the plane that fits the boundary vertices best in a least squares sense. However, for meshes with a complex boundary, this simple procedure may lead to undesirable fold-overs in the boundary polygon and cannot be used. In general, there are two issues to take into account here: (1) choosing the *shape* of the boundary of the parameter domain and (2) choosing the *distribution* of the parameter points around the boundary.

Choosing the shape

In many applications, it is sufficient (or even desirable) to take a rectangle or a circle as parameter domain, with the advantage that such a convex shape guarantees the bijectivity of the parameterization if positive barycentric coordinates like the mean value coordinates are used to compute the parameter points for the interior vertices. The convexity restriction may, however, generate big distortions near the boundary when the boundary of the triangle mesh $S_{\mathcal{T}}$ does not resemble a convex shape. One practical solution to avoid such distortions is to build a "virtual" boundary, i.e., to augment the given mesh with extra triangles around the boundary so as to construct an extended mesh with a "nice" boundary. This approach has been successfully used by Lee et al. [2002], and Kós and Várady [2003].

Choosing the distribution

The usual procedure mentioned in the literature is to use a simple univariate parameterization method such as *chord length* [Ahlberg et al., 1967] or *centripetal* parameterization [Lee, 1989] for placing the parameter points either around the whole boundary, or along each side of the boundary when working with a rectangular domain [Hormann, 2001, Section 1.2.5].

Despite these heuristics working pretty well in some cases, having to fix the boundary vertices may be a severe limitation in others and the next chapter studies parameterization methods that can include the position of the boundary parameter points in the optimization process and thus yield parameterizations with less distortion.

Chapter 4

Setting the Boundary Free

As explained in the previous section, Tutte's theorem combined with mean value weights provides a provably correct way of constructing a valid parameterization for a disk-like surface. However, for some surfaces, the necessity to fix the boundary on a convex polygon may be problematic (c.f. Figure 4.1), for the following reasons : (1) in general, it is difficult to find a "natural" way of fixing the border on a convex polygon, and (2) for some surfaces, the shape of the boundary is far from convex. Therefore the obtained parameterization shows high deformations. Even if one can imagine different ways of improving the result shown in the Figure, the so-obtained parameterization will be probably not as good as the one shown in Figure 4.1-C, that better matches what a tanner would expect for such a mesh. For these reasons, the next section studies the methods that can construct parameterizations with free boundaries, that minimize deformations in a similar way. We start by giving an intuition of how to use the notions from differential geometry explained in Chapter 2 in our context of parameterization with free boundaries.

4.1 Deformation analysis

To see how to apply the theoretic concepts explained in Chapter 2, we first need to grasp their intuitive meaning.

4.1.1 The Jacobian matrix

The first derivatives of the parameterization are involved in deformation analysis, it is then necessary to have an intuition of their geometric meaning. In physics, material point mechanics studies the movement of an object, approximated by a point $\mathbf{p}$, when forces are applied to it. The *trajectory* is the curve described by the point $\mathbf{p}$ when t varies from t_0 to t_1, where t denotes time. The function putting a given time t in correspondence with the position $\mathbf{p}(t) = \{x(t), y(t), z(t)\}$ of the point $\mathbf{p}$ is a parameterization of the trajectory, i.e., a parameterization of a *curve*. It is well known that the vector of the derivatives $\mathbf{v}(t) = \partial\mathbf{p}/\partial t = \{\partial x/\partial t, \partial y/\partial t, \partial z/\partial t\}$ corresponds to the *speed* of $\mathbf{p}$ at time t.

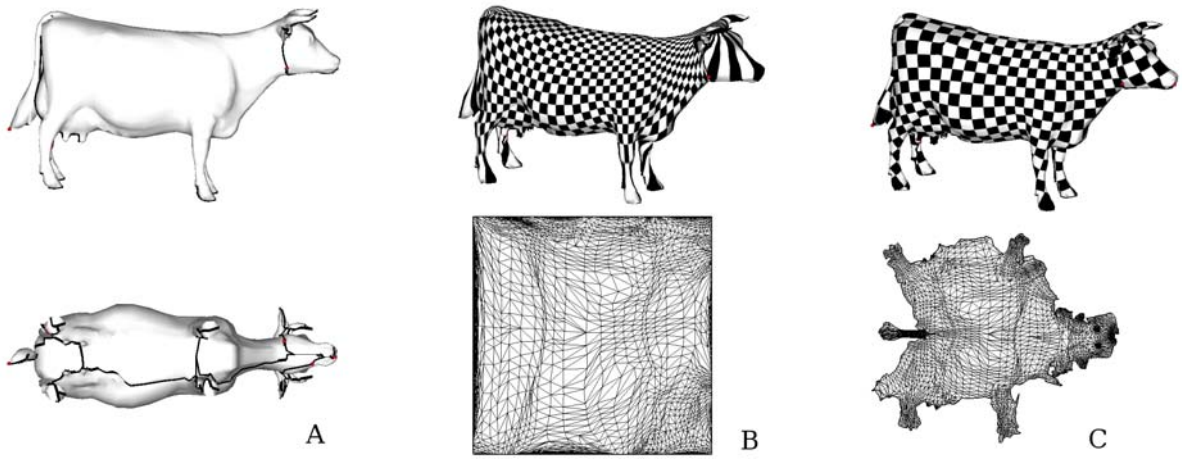


Figure 4.1: A: a mesh cut in a way that makes it homeomorphic to a disk, using the *seamster* algorithm [Sheffer and Hart, 2002]; B: Tutte-Floater parameterization obtained by fixing the border on a square; C: parameterization obtained with a free-boundary parameterization [Sheffer and de Sturler, 2001].

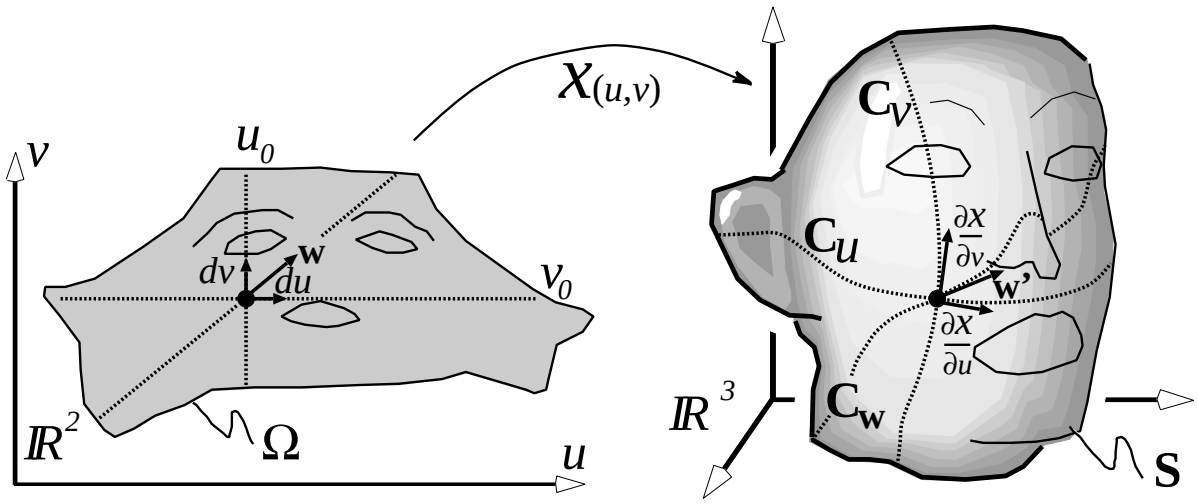


Figure 4.2: Elementary displacements from a point (u, v) of Ω along the u and the v axes are transformed into the tangent vectors to the iso-u and iso-v curves passing through the point $f(u, v)$

As shown in Figure 4.2, we consider now a function $f : (u, v) \mapsto (x, y, z)$, putting a subspace Ω of $\mathbb{R}^2$ in one-to-one correspondence with a surface $\mathbf{S}$ of $\mathbb{R}^3$. The scalars (u, v) are the coordinates in parameter space. In the case of a curve parameterization, the curve is described by a single parameter t. In contrast, in our case, we consider a surface parameterization $f(u, v) = \{x(u, v), y(u, v), z(u, v)\}$, and there are **two** parameters, u and v. Therefore, at a given point (u_0, v_0) of the parameter space Ω, there are two "speed" vectors to consider: $f_u = (\partial f/\partial u)(u_0, v_0)$ and $f_v = (\partial f/\partial v)(u_0, v_0)$. It is easy to check that f_u is the "speed" vector of the curve $\mathbf{C}_u : t \mapsto f(u_0 + t, v_0)$ at $\mathcal{X}(u_0, v_0)$ and that f_v is the "speed" vector of the curve $\mathbf{C}_v : t \mapsto f(u_0, v_0 + t)$. The curve $\mathbf{C}_u$ (resp. $\mathbf{C}_v$) is the iso-u (resp. the iso-v) curve passing through $f(u_0, v_0)$, i.e. the image through f of the line of equation $u = u_0$ (resp. $v = v_0$).

4.1.2 The 1^{st} fundamental form and the anisotropy ellipse

At that point, one may think that the information provided by the two vectors $f_u(u_0, v_0)$ and $f_v(u_0, v_0)$ is not sufficient to characterize the distortions between Ω and $\mathcal{S}$ in the neighborhood of (u_0, v_0) and $f(u_0, v_0)$. In fact, they can be used to compute how an arbitrary vector $\mathbf{w} = (a, b)$ in parameter space is transformed into a vector $\mathbf{w}'$ in the neighborhood of (u_0, v_0). In other words, we want to compute the "speed" vector $\mathbf{w}' = \partial f(u_0 + t.a, v_0 + t.b)/\partial t$ of the curve corresponding to the image of the straight line $(u, v) = (u_0, v_0) + t.w$. The vector $\mathbf{w}'$, i.e. the tangent to the curve $\mathcal{C}_{\mathbf{w}}$, can be simply computed by applying the chain rule, and one can check that it can be computed from the derivatives of f as follows : $\mathbf{w}' = af_u(u_0, v_0) + bf_v(u_0, v_0)$. The vector $\mathbf{w}'$ is referred to as the *directional derivative* of f at (u_0, v_0) relative to the direction $\mathbf{w}$.

In matrix form, $\mathbf{w}'$ is obtained by $\mathbf{w}' = \mathbf{J}(u_0, v_0)\mathbf{w}$, where $\mathbf{J}(u_0, v_0)$ is the matrix of all the partial derivatives of f:

$$\mathbf{J}(u_0, v_0) = \begin{bmatrix} \frac{\partial x}{\partial u}(u_0, v_0) & \frac{\partial x}{\partial v}(u_0, v_0) \\ \\ \frac{\partial y}{\partial u}(u_0, v_0) & \frac{\partial y}{\partial v}(u_0, v_0) \\ \\ \frac{\partial z}{\partial u}(u_0, v_0) & \frac{\partial z}{\partial v}(u_0, v_0) \end{bmatrix} = \Big[f_u(u_0, v_0) \vdots f_v(u_0, v_0) \Big] \tag{4.1}$$

As already said in Chapter 2, the matrix $\mathbf{J}(u_0, v_0)$ is referred to as the *Jacobian matrix* of f at (u_0, v_0).

The notion of directional derivative makes it possible to know what an elementary displacement $\mathbf{w}$ from a point (u_0, v_0) in parameter space becomes when it is transformed by the function f. The Jacobian matrix helps also computing dot products and vector norms onto the surface $\mathcal{S}$. This can be done using the matrix $\mathbf{J}^T\mathbf{J}$, referred to as the 1^{st} *fundamental form* of f, also described in the differential geometry section. This matrix is denoted by $\mathbf{I}$, and defined by :

$$\mathbf{I}(u_0, v_0) = \mathbf{J}^T\mathbf{J} = \begin{bmatrix} f_u \cdot f_u & f_u \cdot f_v \\ \\ f_v \cdot f_u & f_v \cdot f_v \end{bmatrix} \tag{4.2}$$

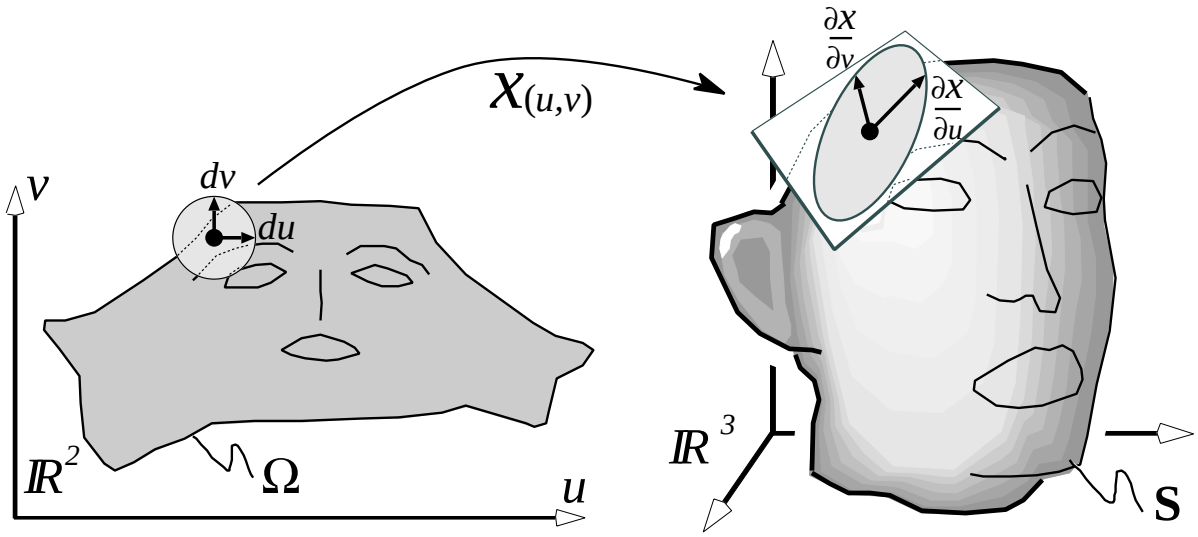


Figure 4.3: Anisotropy: an elementary circle is transformed into an elementary ellipse.

The 1^{st} fundamental form $\mathbf{I}(u_0, v_0)$ is also referred to as the *metric tensor* of f, since it makes it possible to measure how distances and angles are transformed in the neighborhood of (u_0, v_0). The squared norm of the image $\mathbf{w}'$ of a vector $\mathbf{w}$ is given by $||\mathbf{w}'||^2 = \mathbf{w}^T\mathbf{I}\mathbf{w}$, and the dot product $\mathbf{w}_1'^T\mathbf{w}_2' = \mathbf{w}_1^T\mathbf{I}\mathbf{w}_2$ determines how the angle between $\mathbf{w}_1$ and $\mathbf{w}_2$ is transformed. The next section gives a geometric interpretation of the 1^{st} fundamental form and its eigenvalues.

The previous section has studied how an elementary *displacement* from a parameter-space location (u_0, v_0) is transformed through the parameterization f. As shown in Figure 4.3, our goal is now to determine what an elementary *circle* becomes.

Let us consider the two eigenvalues λ_1, λ_2 of G, and two associated unit eigenvectors $\mathbf{w}_1, \mathbf{w}_2$. Note that since $\mathbf{I}$ is symmetric, $\mathbf{w}_1$ and $\mathbf{w}_2$ are orthogonal. An arbitrary unit vector $\mathbf{w}$ can be written as $\mathbf{w} = \cos(\theta)\mathbf{w}_1 + \sin(\theta)\mathbf{w}_2$. The squared norm of $\mathbf{w}' = \mathbf{J}\mathbf{w}$ is then given by:

$$\begin{aligned} ||\mathbf{w}'||^2 &= \mathbf{w}^T\mathbf{I}\mathbf{w} \\ &= (\cos(\theta)\mathbf{w}_1 + \sin(\theta)\mathbf{w}_2)^T\mathbf{I}(\cos(\theta).\mathbf{w}_1 + \sin(\theta).\mathbf{w}_2) \\ &= \cos^2(\theta)||\mathbf{w}_1||^2\lambda_1 + \sin^2(\theta)||\mathbf{w}_1||^2\lambda_2 + \\ &\quad \sin(\theta)\cos(\theta)(\lambda_1\mathbf{w}_2^T\mathbf{w}_1 + \lambda_2\mathbf{w}_1^T\mathbf{w}_2) \\ &= \cos^2(\theta).\lambda_1 + \sin^2(\theta).\lambda_2 \end{aligned} \tag{4.3}$$

In Equation 4.3, the cross terms $\mathbf{w}_1^T\mathbf{w}_2$ and $\mathbf{w}_2^T\mathbf{w}_1$ vanish since $\mathbf{w}_1$ and $\mathbf{w}_2$ are orthogonal. Let us see now what are the extrema of $||\mathbf{w}'||^2$ in function of θ.

$$\begin{aligned} \frac{\partial||\mathbf{w}'(\theta)||^2}{\partial\theta} &= 2\sin(\theta)\cos(\theta)(\lambda_2 - \lambda_1) \\ &= \sin(2\theta)(\lambda_2 - \lambda_1) \end{aligned} \tag{4.4}$$

The extrema of $||\mathbf{w}'(\theta)||^2$ are then obtained for $\theta \in \{0, \pi/2, \pi, 3\pi/2\}$, i.e. for $\mathbf{w} = \mathbf{w}_1$ or $\mathbf{w} = \mathbf{w}_2$. Therefore, the maximum and minimum values of $||\mathbf{w}'(\theta)||^2$ are λ_1 and λ_2,

and:

- The axes of the anisotropy ellipse are $\mathbf{J}\mathbf{w}_1$ and $\mathbf{J}\mathbf{w}_2$;
- The lengths of the axes are $\sqrt{\lambda_1}$ and $\sqrt{\lambda_2}$.

Note: as mentioned in Chapter 2, the lengths of the axes $\sqrt{\lambda_1}$ and $\sqrt{\lambda_2}$ also correspond to the singular values of the matrix $\mathbf{J}$. A geometric interpretation of the SVD is also explained in that chapter. We remind that the singular value decomposition (SVD) of a matrix $\mathbf{J}$ writes:

$$\mathbf{J} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{U} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{bmatrix} \mathbf{V}^T$$

where $\mathbf{U} : 3 \times 3$ and $\mathbf{V} : 2 \times 2$ are such that their column vectors form an orthonormal basis (we also say that they are *unit* matrices), and $\mathbf{\Sigma}$ is a matrix such that only its diagonal elementsσ_1, σ_2 are non-zero. The scalars σ_1, σ_2 are called the *singular values* of J. In our case, by substituting the Jacobian matrix J with its SVD, we obtain:

$$\begin{aligned} \mathbf{I} &= \mathbf{J}^T\mathbf{J} \\ &= (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T) \\ &= \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad = \quad \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T \\ &= \mathbf{V} \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \mathbf{V}^T \end{aligned}$$

Since $\mathbf{U}$ is a unit matrix, the central term $\mathbf{U}^T\mathbf{U}$ of the third line is equal to the identity matrix and vanishes. We then obtain SVD of the matrix $\mathbf{I}$, that is also a diagonalization of $\mathbf{I}$. By unicity of the SVD, we deduce the relation between the eigenvalues λ_1, λ_2 of G and the singular values σ_1, σ_2 of J: $\lambda_1 = \sigma_1^2$ and $\lambda_2 = \sigma_2^2$.

We can now give the expression of the lengths of the anisotropy ellipse σ_1 and σ_2. We first recall the expression of the Jacobian matrix as a function of the gradient vectors:

$$\mathbf{I} = \mathbf{J}^T\mathbf{J} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} \qquad \text{with} \qquad \begin{cases} E = f_u^2 \\ F = f_u \cdot f_v \\ G = f_v^2 \end{cases} \tag{4.5}$$

The lengths of the axis of the anisotropy ellipse correspond to the eigenvalues of $\mathbf{I}$. Their expression can be found by computing the square roots of the zeros of the characteristic polynomial $|\mathbf{I} - \sigma\mathbf{Id}|$ (that is a second order equation in σ) :

$$\begin{aligned} \sigma_1 &= \sqrt{1/2(E+G) + \sqrt{(E-G)^2 + 4F^2}} \\ \sigma_2 &= \sqrt{1/2(E+G) - \sqrt{(E-G)^2 + 4F^2}} \end{aligned} \tag{4.6}$$

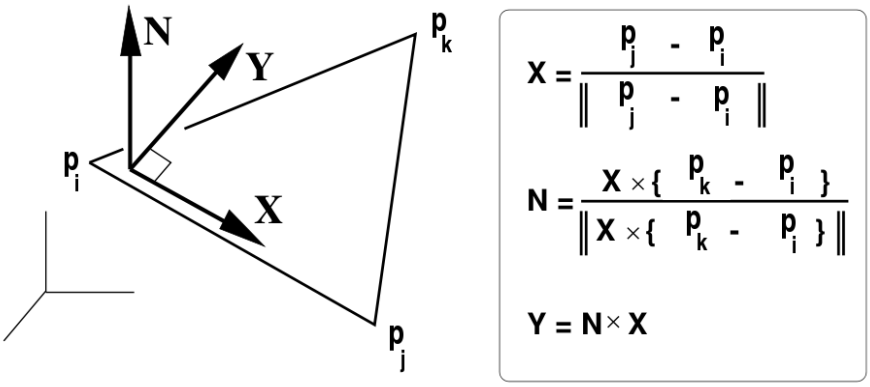


Figure 4.4: Local X, Y basis in a triangle.

Deformation analysis for triangulated surfaces

Deformation analysis, introduced in the previous section, involves the computation of the gradients of the parameterization as a function of the parameters u and v. In the case of a triangulated surface, the parameterization is a piecewise linear function. Therefore, the gradients are constant in each triangle.

Before studying the computation of these gradients, we need to mention that our setting is slightly different from the previous section. In our case, as previously mentioned, the 3D surface is given, and our goal is to construct the parameterization. In this setting, it seems more natural to characterize the inverse of the parameterization, i.e. the function that goes from the 3D surface (known) to the parametric space (unknown). This function is also piecewise linear. To port deformation analysis to this setting, it is possible to provide each triangle with an orthonormal basis X, Y, as shown in Figure 4.4 (and we can use one of the vertices $\mathbf{p}_i$ of the triangle as the origin). In this basis, we can study the inverse of the parameterization, that is to say the function that maps a point (X, Y) of the triangle to a point (u, v) in parameter space. This function writes :

$$\begin{cases} u(X,Y) &= \lambda_1 u_i + \lambda_2 u_j + \lambda_3 u_k \\ v(X,Y) &= \lambda_1 v_i + \lambda_2 v_j + \lambda_3 v_k \end{cases}$$

where $(\lambda_1, \lambda_2, \lambda_3)$ denote the barycentric coordinates at the point (x, y) in the triangle, computed as before :

$$\begin{pmatrix} \lambda_i \\ \lambda_j \\ \lambda_k \end{pmatrix} = \frac{1}{2|T|_{X,Y}} \begin{pmatrix} Y_j - Y_k & X_k - X_j & X_j Y_k - X_k Y_j \\ Y_k - Y_i & X_i - X_k & X_k Y_i - X_i Y_k \\ Y_i - Y_j & X_j - X_i & X_i Y_j - X_j Y_i \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

where $2|T|_{X,Y} = (X_i Y_j - Y_i X_j) + (X_j Y_k - Y_j X_k) + (X_k Y_i - Y_k X_i)$ denotes the double area of the triangle (in 3D space this time).

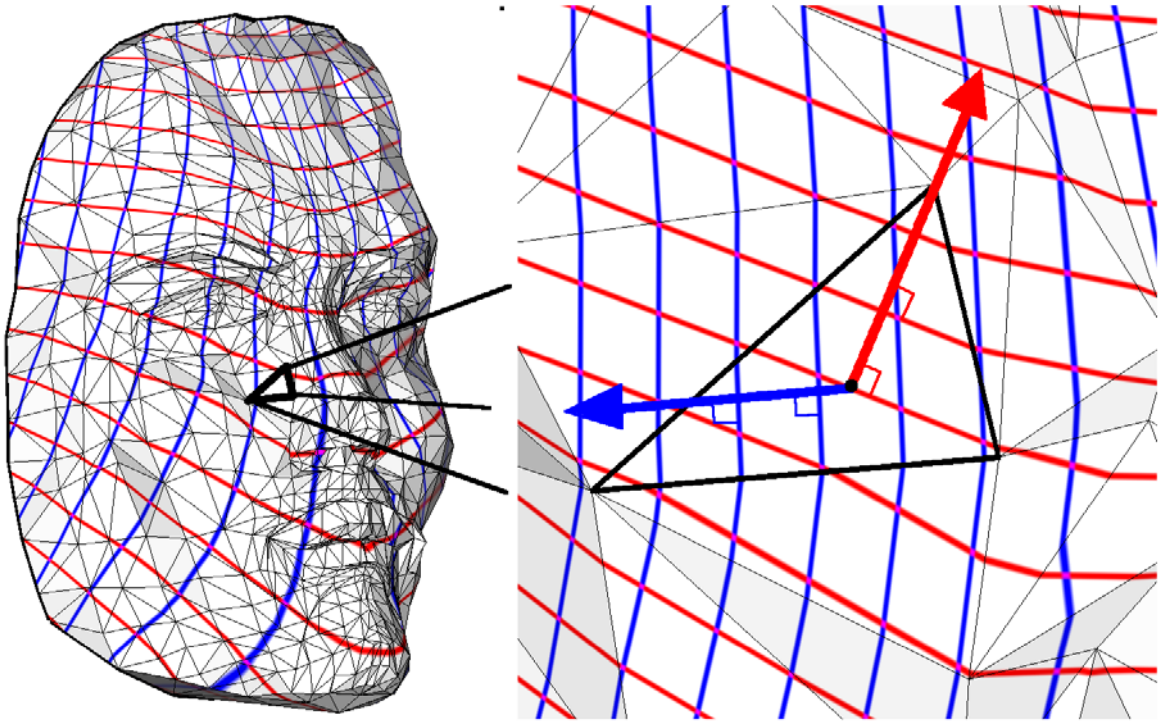


Figure 4.5: Iso-u,v curves and associated gradients.

By substituting the values of λ_1, λ_2 and λ_3 in $u = \lambda_1 u_i + \lambda_2 u_j + \lambda_3 u_k$ (resp. v), we obtain:

$$\begin{pmatrix} \partial u/\partial X \\ \partial u/\partial Y \end{pmatrix} = \mathbf{M}_T \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} = \frac{1}{2|T|_{X,Y}} \begin{pmatrix} Y_j - Y_k & Y_k - Y_i & Y_i - Y_j \\ X_k - X_j & X_i - X_k & X_j - X_i \end{pmatrix} \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} \tag{4.7}$$

where the matrix $\mathbf{M}_T$ solely depends on the geometry of the triangle T.

As shown in Figure 4.5, these gradients are different (but strongly related with) the gradients of the inverse function, computed in the previous section. The gradient of u (resp. v) intersects the iso-us (resp. the iso-vs) with a right angle (instead of being tangent to them), and its norm is the inverse of the one computed in the previous section.

These gradients can then be used to deduce the expression of the Jacobian matrix $\mathbf{J}_T$, as follows :

$$\begin{aligned}
\mathbf{J}_T &= \begin{pmatrix} \frac{\partial u}{\partial X} & \frac{\partial v}{\partial X} \\ \frac{\partial u}{\partial Y} & \frac{\partial v}{\partial Y} \end{pmatrix} \\
&= \frac{1}{2|T|_{x,y}} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} X_3 - X_2 & X_1 - X_3 & X_2 - X_1 \\ Y_3 - Y_2 & Y_1 - Y_3 & Y_2 - Y_1 \end{pmatrix} \begin{pmatrix} u_1 & v_1 \\ u_2 & v_2 \\ u_3 & v_3 \end{pmatrix} \\
&= \frac{1}{2|T|_{x,y}} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \end{pmatrix} \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} u_1 & v_1 \\ u_2 & v_2 \\ u_3 & v_3 \end{pmatrix}
\end{aligned} \tag{4.8}$$

The second line of this equation shows how the Jacobian matrix combines the vectors of the edges of the triangle with the (u, v) coordinates. The third line is a more symmetric expression, that uses the coordinates of the points directly. The square matrix swaps the coordinates of the triangle edges shows an interesting similarity with the imaginary number i (in fact, this is a representation of the imaginary number i). We will elaborate more on the links with complex analysis in Section 4.3 that deals with conformal methods.

4.2 Parameterization methods based on deformation analysis

This section reviews these methods, using the formalism introduced in Chapter 2 and Section 4.1. However, before going further, we need to warn the reader about a possible source of confusion :

- half of the methods study the function that goes from the surface to the parametric space (as in the previous Section). This is justified by the fact that the (u, v) coordinates are unknown. Therefore, it is more natural to go from the known world (the surface) to the unknown world (the parameter space);
- the other half of the methods use the inverse convention, and study the function that goes from parameter space to the surface (as in Section 4.1). This is justified by the fact that it makes the formalism compatible with classical differential geometry books [do Carmo, 1976] that use this convention.

Since both conventions are justified, both are used by different authors. However, as will be shown, deformation analysis for one convention can be easily deduced from the other one. Therefore, except the risk of confusion, this does not introduce much difficulty to understand these methods. So far, the notations we used in this section are related with the function that maps (X, Y) local coordinates in the triangle to (u, v) coordinates in parameter space Ω.

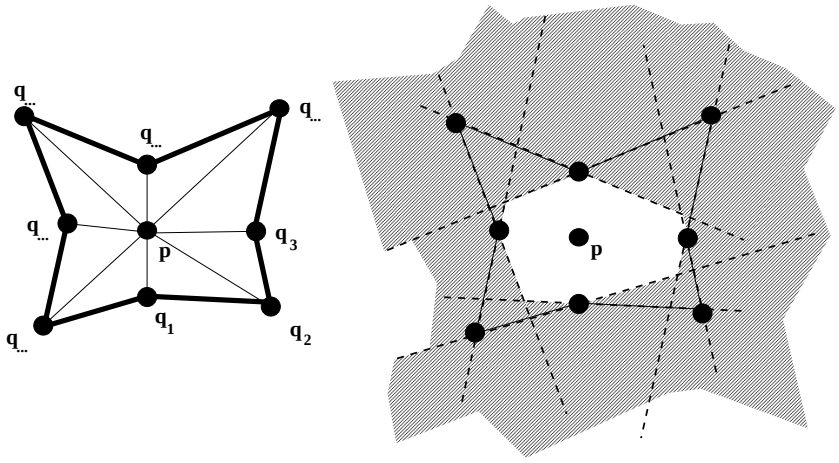


Figure 4.6: Left: to avoid triangle flips, each vertex $\mathbf{p}$ is constrained to remain in the kernel of the polygon defined by its neighbors $\mathbf{q}_i$; Right: the kernel of a polygon (white) is defined by the intersection of the half-planes defined by the support lines of its edges (dashed).

Notations for the convention $(u, v) \mapsto (x, y, z)$

The symbols $\mathbf{J}', \mathbf{I}', \sigma'_1, \sigma'_2$ are related with the function that goes from the parametric space to the surface, and denote respectively the Jacobian matrix, the first fundamental form, and the lengths of the two axes of the anisotropy ellipse.

We now study the relations between these values and those associated with the inverse function. The Jacobian matrix of the inverse of a function f is equal to the inverse of the Jacobian matrix of f^{-1}. Therefore, we have $\mathbf{J}' = \mathbf{J}^{-1}$.

Moreover, it is easy to check that if the SVD of $\mathbf{J}$ writes $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, with $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2)$ and with $\mathbf{U}, \mathbf{V}$ two unit matrices, i.e. $\mathbf{U}^T\mathbf{U} = Id$ and $\mathbf{V}^T\mathbf{V} = Id$, then the SVD of $\mathbf{J}'$ writes $\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T$ (one can check that the product of the two matrices gives the identity matrix). Therefore, the lengths of the smallest and largest axis of the anisotropy ellipse of the inverse function are given by :

$$\sigma'_1 = \frac{1}{\sigma_2} \quad ; \quad \sigma'_2 = \frac{1}{\sigma_1}$$

Armed with these definitions, we can now proceed to review several methods based on deformation analysis, and express them in a common formalism. As explained before, we need to take care of identifying whether the surface $\rightarrow$ parametric-space function or parametric-space $\rightarrow$ surface function is used. Before evoking these methods, we give two more precisions :

- to avoid triangle flips, some of the methods constrain each vertex $\mathbf{p}$ to remain in the kernel of the polygon defined by its neighbors $\mathbf{q}_i$. This notion is illustrated in Figure 4.6. To compute the kernel of a polygon, it is for instance possible to apply Sutherland and Hogdman's re-entrant polygon clipping algorithm to the polygon

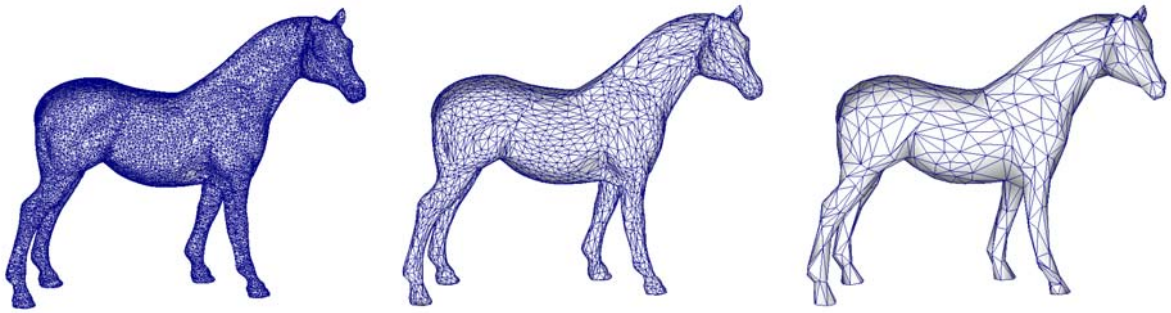


Figure 4.7: Parameterization methods based on deformation analysis often need to minimize non-linear objective functions. To accelerate the computations, these methods often use a multiresolution approach, based on Hoppe et.al's Progressive Mesh data structure.

(clipped by itself). The algorithm is described in most general computer graphics books [Foley et al., 1995];

- since they are based on the eigenvalues of the first fundamental form, the objective functions involved in deformation analysis are often non-linear, and therefore difficult to minimize in an efficient way. To accelerate the computations, a commonly used technique consists in representing the surface in a multi-resolution manner, based on Hoppe's *Progressive Mesh* data structure [Hoppe, 1996]. The algorithm starts by optimizing a simplified version of the object, then introduces the additional vertices and optimizes them by iterative refinements.

Now that we have seen the general notions related with deformation analysis and the particular aspects that concern the optimization of objective functions involved in deformation analysis, we can review several classical methods that belong to this category.

4.2.1 Green-Lagrange deformation tensor

Historically, to minimize the deformations of a parameterization, one of the first methods was developed by Maillot et al. [1993]. The main idea behind their approach consists in minimizing a matrix norm of the Green-Lagrange deformation tensor. This notion comes from mechanics, and measures the deformation of a material. Intuitively, we know that if the metric tensor G is equal to the identity matrix, then we have an isometric parameterization. The Green-Lagrange deformation tensor is given by $G - Id$, and measures the "non-isometry" of the parameterization. Maillot *et. al* minimize the Froebenius norm of this matrix, given by :

$$\|\mathbf{I} - \mathbf{Id}\|_F^2 = (\lambda_1 - 1)^2 + (\lambda_2 - 1)^2$$

where λ_1 and λ_2 denote the eigenvalues of the first fundamental form $\mathbf{I}$.

4.2.2 MIPS

Hormann and Greiner's MIPS (Mostly Isometric Parameterization of Surfaces) method [Hormann and Greiner, 2000a] was to our knowledge the first mesh parameterization method that computes a natural boundary. This method is based on the minimization of the ratio between the two lengths of the axes of the anisotropy ellipse. This corresponds to the 2-norm of the Jacobian matrix :

$$K_2(\mathbf{J}_T) = \|\mathbf{J}_T\|_2 \|\mathbf{J}_T^{-1}\|_2 = \sigma_1/\sigma_2$$

Since minimizing this energy is a difficult numerical problems, Hormann and Greiner have replaced the two norm $\|.\|_2$ by the Froebenius $\|.\|_F$, that is to say the square root of the sum of the squared singular values :

$$\begin{aligned} K_F(\mathbf{J}_T) &= \|\mathbf{J}_T\|_F \|\mathbf{J}_T^{-1}\|_F \\ &= \sqrt{\sigma_1^2+\sigma_2^2}\sqrt{\left(\tfrac{1}{\sigma_1}\right)^2+\left(\tfrac{1}{\sigma_2}\right)^2} \\ &= \tfrac{\sigma_1^2+\sigma_2^2}{\sigma_1\sigma_2} \quad = \quad \frac{\text{trace}(\mathbf{I}_T)}{\det(\mathbf{J}_T)} \end{aligned}$$

As can be seen, fortunate cancelations of terms yield a simple expression at the end. The final expression corresponds to the ratio between the trace of the metric tensor and the determinant of the Jacobian matrix. As indicated in the original article, this value can also be interpreted as the *Dirichlet energy per parameter-space area*: the term trace($\mathbf{I}$) corresponds to the Dirichlet energy, and the Jacobian det($\mathbf{J}$) to the ratio between triangle's area in 3D and in parameter space.

Interestingly, as explained by the authors, $K_F(\mathbf{J}_T)$ also writes :

$$K_F(\mathbf{J}_T) = \sigma_1/\sigma_2 + \sigma_2/\sigma_1 = K_2(\mathbf{J}_T) + K_2(\mathbf{J}_T^{-1})$$

This alternative expression reveals that the anisotropy in *both directions* (from Ω to $\mathcal{S}$ and from $\mathcal{S}$ to Ω) is taken into account.

4.2.3 Stretch minimization

Motivated by texture mapping applications, Sander et al. [2001] studied the way a signal stored in parameter space is deformed when it is texture-mapped onto the surface (by applying the parameterization). For this reason, their formalism uses the inverse function, that maps the parametric space onto the surface. Therefore, we use the notations J', G', σ_1', σ_2', as explained at the beginning of the section.

A possible way of characterizing the deformations of a texture is to consider a point and a direction in parameter space and analyze how the texture is deformed along that direction. Sander *et. al* called this value the "stretch". This exactly corresponds to the notion of directional derivative, that we introduced in Section 4.1. For a triangle T, they

Figure 4.8: Some results computed by stretch L_2 minimization (parameterized models courtesy of Pedro Sander and Alla Sheffer).

defined two energies, that correspond to the average value of the stretch for all directions (stretch $L_2(T)$), and to the maximum stretch (stretch $L_\infty(T)$) :

$$\begin{array}{rcccl} L_2(T) & = & \sqrt{(\sigma_1'^2 + \sigma_2'^2)/2} & = & \sqrt{((1/\sigma_1)^2 + (1/\sigma_2)^2)/2} \\ L_\infty(T) & = & \sigma_2' & = & 1/\sigma_1 \end{array}$$

where the expression of $\sigma_1' = 1/\sigma_2$ and $\sigma_2' = 1/\sigma_1$ are given at the beginning of the section. The local energies of each triangle T are combined into a global energy $L_2(S)$ and $L_\infty(S)$ defined as follows :

$$L_2(S) = \sqrt{\frac{\sum_T |T| L_2(T)}{\sum_T |T|}}$$

$$L_\infty(S) = \max_T L_\infty(T)$$

Figure 4.8 shows some results computed with this approach. This formalism is particularly well suited to texture mapping applications, since it minimizes the deformations that are responsible of the visual artifacts that this type of application wants to avoid. Moreover, a simple modification of this method allows the contents of the texture to be taken into account, and therefore to define a signal-adapted parameterization [Sander et al., 2002].

4.2.4 Symmetric energy

A similar method was proposed in [Sorkine et al., 2002]. Based on the remark that shrinking and stretching should be treated the same, they replace the L_2 and L_∞ energy with the following one, more symmetric with this respect:

$$D_T = \max(\sigma_2', 1/\sigma_1') = \max(1/\sigma_1, \sigma_2)$$

4.2.5 Combined energy

To introduce more flexibility in these methods, Degener et al. [2003] proposed to use a combined energy, with a term $\sigma'_1\sigma'_2$ that penalizes area deformations[1], and a term σ'_1/σ'_2 that penalizes angular deformations. To facilitate the numerical optimization of the objective function, each term x is replaced by the expression $x + 1/x$, which gives :

$$E_{combined} = E_{angle} \times (E_{area})^{\theta}$$

with :

$$E_{area} = \sigma'_1\sigma'_2 + \frac{1}{\sigma'_1\sigma'_2} = \det(\mathbf{J}'_T) + \frac{1}{\det(\mathbf{J}'_T)}$$

$$E_{angle} = \frac{\sigma'_1}{\sigma'_2} + \frac{\sigma'_2}{\sigma'_1} = \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1}{\sigma_2} = E_{MIPS}$$

where the parameter θ makes it possible to choose the relative importance of both terms. One can notice that the angular term corresponds to the energy minimized by the MIPS method (see above).

4.3 Conformal methods

Conformal methods are related with the formalism of complex analysis. The involved conformality condition defines a criterion with sufficient "rigidity" to offer good extrapolation capabilities, that can compute natural boundaries. The reader interested with this formalism may read the excellent book by Needham [1997].

As seen in this section, deformation analysis, introduced in Section 4.1, plays a central role in the definition of (non-distorted) parameterization methods. We now focus on a particular family of methods, for which the anisotropy ellipse is a circle for all point of the surface. As shown in Figure 4.9, this also means that the two gradient vectors f_u and f_v are orthogonal and have the same norm. The condition can also be written as $f_v = \mathbf{n} \times f_u$, where $\mathbf{n}$ denotes the normal vector. Interestingly, if a parameterization is conformal, this is also the case of the inverse function (since the Jacobian matrix of the inverse is equal to the inverse of the Jacobian matrix). The relation can also be seen in Figure 4.5, if the iso-u,v curves are orthogonal, it is also the case of their normal vectors. Finally, conformality also means that the Jacobian matrix is composed of a rotation and a scaling (in other words, a *similarity*). Therefore, conformal applications locally correspond to similarities. In terms of differential geometry (Chapter 2), this means that the lengths of the anisotropy ellipse are the same ($\sigma_1 = \sigma_2$). We now review different methods that compute a conformal parameterization.

[1]This corresponds to the Jacobian of the parameterization, i.e. the determinant of the Jacobian matrix, that defines the differential element for areas.

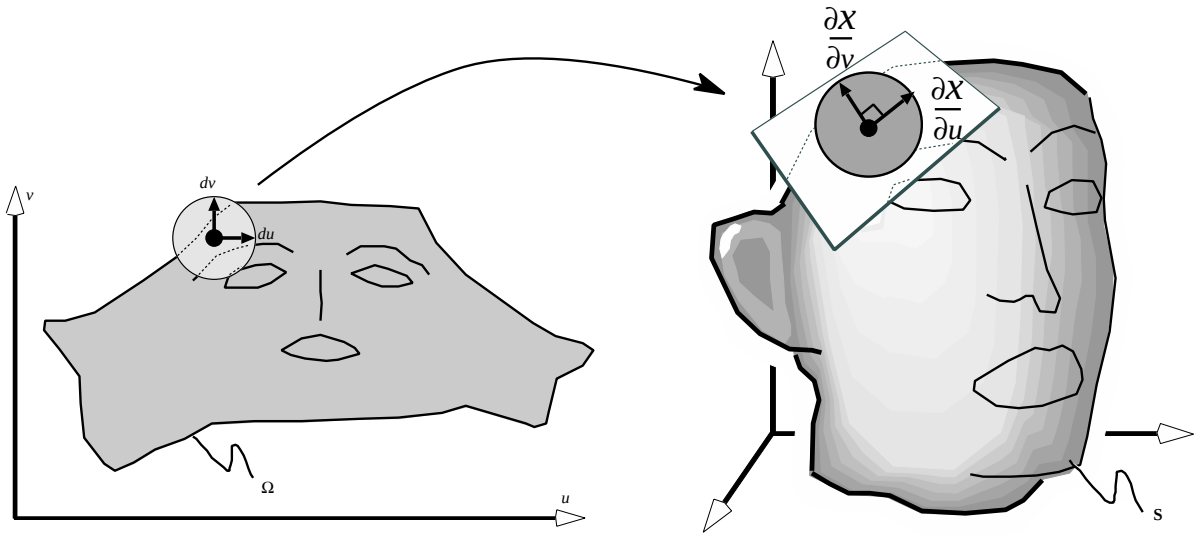


Figure 4.9: A conformal parameterization transforms an elementary circle into an elementary circle.

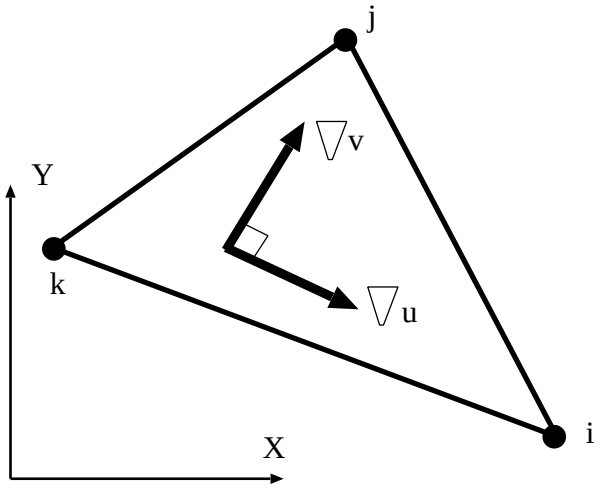


Figure 4.10: In a triangle provided with a local (X, Y) basis, it is easy to express the condition that characterizes conformal maps.

LSCM

In contrast with the exposition of the initial paper [Lévy et al., 2002], we will present the method in terms of simple geometric relations between the gradients. We will then elaborate with the complex analysis formalism, and establish the relation with other methods.

The LSCM method (Least Squares Conformal Maps) simply expresses the conformality condition of the functions that maps the surface to parameter space. We now consider one of the triangles of the surface, provided with an orthonormal basis (X, Y) of its support plane (c.f. Figure 4.10). In this context, conformality writes :

$$\nabla v = \mathrm{rot}_{90}(\nabla u) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \nabla u \tag{4.9}$$

where rot_{90} denotes the anticlockwise rotation of 90 degrees.

Using the expression of the gradient in a triangle (derived at the end of Section 4.1.2), Equation 4.9, that characterizes piecewise linear conformal maps rewrites :

$$\mathbf{M}_T \begin{pmatrix} v_i \\ v_j \\ v_k \end{pmatrix} - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \mathbf{M}_T \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

where the matrix $\mathbf{M}_T$ is given by Equation 4.7.

In the continuous setting, Riemann proved that any surface admits a conformal parameterization. However, in our specific case of piecewise linear functions, only developable surfaces admit a conformal parameterization. For a general (non-developable) surface, we minimize an energy E_{LSCM} that corresponds to the "non-conformality" of the application, and called the *discrete conformal energy* :

$$E_{LSCM} = \sum_{T=(i,j,k)} |T| \left\| \mathbf{M}_T \begin{pmatrix} v_i \\ v_j \\ v_k \end{pmatrix} - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \mathbf{M}_T \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} \right\|^2 \tag{4.10}$$

Note that conformality is invariant through similarities applied in parameter space. For this reason, the quadratic form E_{LSCM} is not definite positive, and its matrix is singular. However, it is possible to define the 4 degrees of freedom of the similarity by removing two vertices from the set of variables, using the technique presented in Section 10.4.3. The so-defined quadratic form is provably positive definite.

We have considered conformal maps from the point of view of the gradients. In the next section, we exhibit relations between conformal maps and harmonic functions. This also shows some connections with Floater's barycentric mapping method and with its more recent generalizations.

Conformal maps and harmonic maps

Conformal maps play a particular role in complex analysis and Riemannian geometry. The following system of equations characterizes conformal maps :

$$\begin{aligned} \frac{\partial v}{\partial x} &= -\frac{\partial u}{\partial y} \\ \\ \frac{\partial v}{\partial y} &= \frac{\partial u}{\partial x} \end{aligned}$$

This system of equations is known as Cauchy-Riemann equations They play a central role in complex analysis, since they characterize differentiable complex functions (also called analytic functions). We focus on the complex function $U(X) = u(X) + iv(X)$ with $X = x + iy$. It is then possible to prove that the derivative $U'(X)$ of U at point X defined by :

$$U'(X) = \lim_{A \to 0} \frac{U(X) - U(A)}{X - A}$$

exists if and only if Cauchy-Riemann equations are satisfied.

Another way of understanding this relation is given by the order 1 Taylor expansion. In the case of a function f from $\mathbb{R}$ to $\mathbb{R}$, the Taylor expansion of f writes $f(x_0 + a) \simeq f(x_0) + f'(x_0)a$. This realizes a local linear approximation of the function f around the point x_0.

If we now consider a complex function $U(X)$, when the complex derivative is defined, the Taylor expansion writes $U(X_0 + A) \simeq U(X_0) + U'(X_0)A$. If we consider the complex function U as a 2D geometric transform of the plane, and by writing $U'(X_0) = Re^{i\theta}$ in polar form, we get $U(X_0 + A) \simeq U(X_0) + Re^{i\theta}A$. In other words, differentiable complex functions behave locally like a similarity of the plane (composed of a translation of vector $U(X_0)$, a rotation of angle θ and a scaling of factor R). This gives another explanation for the Cauchy-Riemann equations : the rotation and the scaling applied to the two gradient vectors need to match.

Note that with complex numbers, the energy minimized by LSCM takes a simple form, that reveals the symmetric roles played by the coordinates (X_i, Y_i) on the surface and the coordinates (u_i, v_i) in parameter-space :

$$E_{LSCM} = \frac{1}{2} \left| \begin{pmatrix} P_1 & P_2 & P_3 \end{pmatrix} \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} \right|^2$$

where $U_1 = (u_1 + iv_1)$ (resp. U_2, U_3) and where $P_1 = (X_1 + iY_1)$ (resp. P_2, P_3).

Another interesting property of complex differentiable functions is that their order-1 differentiability makes them differentiable at any order. We can then use the Cauchy-Riemann equations to compute the order 2 derivatives of u and v.

This establishes an interesting relation with the Laplacian operator Δ :

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} &= \frac{\partial^2 v}{\partial x \partial y} \\ \\ \frac{\partial^2 u}{\partial y^2} &= -\frac{\partial^2 v}{\partial x \partial y} \end{aligned}$$

or :

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

$$\Delta v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = 0$$

In other words, the real part and the imaginary part of a conformal map are two harmonic functions (i.e. two functions with zero Laplacian). This is the point of view adopted by Desbrun et al. [2002] to develop their conformal parameterization method, nearly equivalent to LSCM. Thus, they compute two harmonic functions while letting the border evolve. On the border, a set of constraints enforce the conformality of the parameterization, and introduce a coupling term between the u's and the v's.

Another way of considering both approaches, mentioned by Pinkall and Polthier [1993] and probably at the origin of Desbrun, Meyer, and Alliez's intuition, is given by Plateau's problem [Plateau, 1873; Meeks, 1981]. Given a closed curve, this problem concerns the existence of a surface with minimum area, such that its border matches the closed curve. To minimize the area of a surface, Douglas [1931] and Radó [1930], and later Courant [1950] considered Dirichlet's energy (i.e. the integral of the squared norm of the gradients), easier to manipulate. A discretization of this energy was proposed by Pinkall and Polthier [1993], with the aim of giving a practical solution to Plateau's problem in the discrete case. Dirichlet's energy differs from the area of the surface. The difference is a term that depends on the parameterization, called the *conformal energy*. The conformal energy is equal to zero if the parameterization is conformal. The relation between these three quantities is explained below :

$$\underbrace{\int_S \det(\mathbf{J}) ds}_{\text{area of the surface}} = \underbrace{\frac{1}{2}\int_S \|f_u\|^2 + \|f_v\|^2 ds}_{\text{Dirichlet's energy}} - \underbrace{\frac{1}{2}\int_S \|f_v - \text{rot}_{90}(f_u X)\|^2}_{\text{conformal energy}}$$

This relation is easy to prove, by expanding the integrated terms in a local basis X, Y of the tangent plane:

$$\underbrace{\frac{\partial X}{\partial u}\frac{\partial Y}{\partial v} - \frac{\partial X}{\partial v}\frac{\partial Y}{\partial u}}_{\det(\mathbf{J})} = \underbrace{\frac{1}{2}\left(\left\|\begin{pmatrix}\frac{\partial X}{\partial u}\\ \frac{\partial Y}{\partial u}\end{pmatrix}\right\|^2 + \left\|\begin{pmatrix}\frac{\partial X}{\partial v}\\ \frac{\partial Y}{\partial v}\end{pmatrix}\right\|^2\right)}_{\|f_u\|^2+\|f_v\|^2} - \underbrace{\frac{1}{2}\left(\left\|\begin{pmatrix}\frac{\partial X}{\partial v}+\frac{\partial Y}{\partial u}\\ \frac{\partial Y}{\partial v}-\frac{\partial X}{\partial u}\end{pmatrix}\right\|^2\right)}_{\|f_v-\text{rot}_{90}(f_u)\|^2}$$

Therefore, LSCM minimizes the conformal energy, and Desbrun *et.al*'s method minimize Dirichlet's energy. Since the difference between these two quantities corresponds to the (constant) area of the surface, both methods are equivalent.

All the methods mentioned above are based on relations between the gradients, the Jacobian or the first fundamental form of the parameterization. For this reason, they can be qualified as *analytical* methods. In the next chapter, we focus on *geometrical* methods, that consider the shape of the triangles (and more specifically their **angles**).

Chapter 5

Angle-Space Methods

Instead of defining a planar parameterization in terms of vertex coordinates, both the ABF/ABF++ method [Sheffer and de Sturler, 2000, 2001; Sheffer et al., 2005] and the circle-patterns algorithm [Kharevych et al., 2006] define it in terms of the angles of the planar triangles. Figure 7.3 provides a comparison between angle-space and direct conformal methods. As demonstrated, angle-space methods introduce significantly less stretch into the parameterization on models that have regions of high Gaussian curvature.

The ABF method (Angle Based Flattening) [Sheffer and de Sturler, 2000, 2001] is based on the following observation: a planar triangulation is uniquely defined by the corner angles of its triangles (modulo a similarity transformation). Based on this observation the authors reformulate the parameterization problem – finding (u_i, v_i) coordinates – in terms of angles, that is to say finding the angles α_k^t, where α_i^t denotes the angle at the corner of triangle t incident to vertex k.

To ensure that the 2D angles define a valid triangulation, a set of constraints needs to be satisfied.

- Triangle validity (for each triangle t):

$$\forall t \in T, \quad \alpha_1^t + \alpha_2^t + \alpha_3^t - \pi = 0; \tag{5.1}$$

- Planarity (for each internal vertex v):

$$\forall v \in V_{int}, \quad \sum_{(t,k)\in v^*} \alpha_k^t - 2\pi = 0, \tag{5.2}$$

  where V_{int} denotes the set of internal vertices, and where v^* denotes the set of angles incident to vertex v.

- Positivity : $\alpha_k^t > 0$ for all angles. We note that this constraint can be ignored in most practical setups [Sheffer et al., 2005], simplifying the solution process.

- Reconstruction (for each internal vertex) - this constraints ensures that edges shared by pairs of triangles has the same length:

$$\forall v \in V_{int}, \quad \prod_{(t,k)\in v^*} \sin \alpha_{k\oplus 1}^t - \prod_{(t,k)\in v^*} \sin \alpha_{k\ominus 1}^t = 0. \tag{5.3}$$

The indices $k \oplus 1$ and $k \ominus 1$ denote the next and previous angle in the triangle. Intuitively, note that the product $\sin \alpha^t_{k\oplus 1} \sin \alpha^t_{k\ominus 1}$ corresponds to the product of the ratio between the lengths of two consecutive edges around vertex k. If they do not match, it is the possible to "turn around" vertex k without "landing" on the starting point.

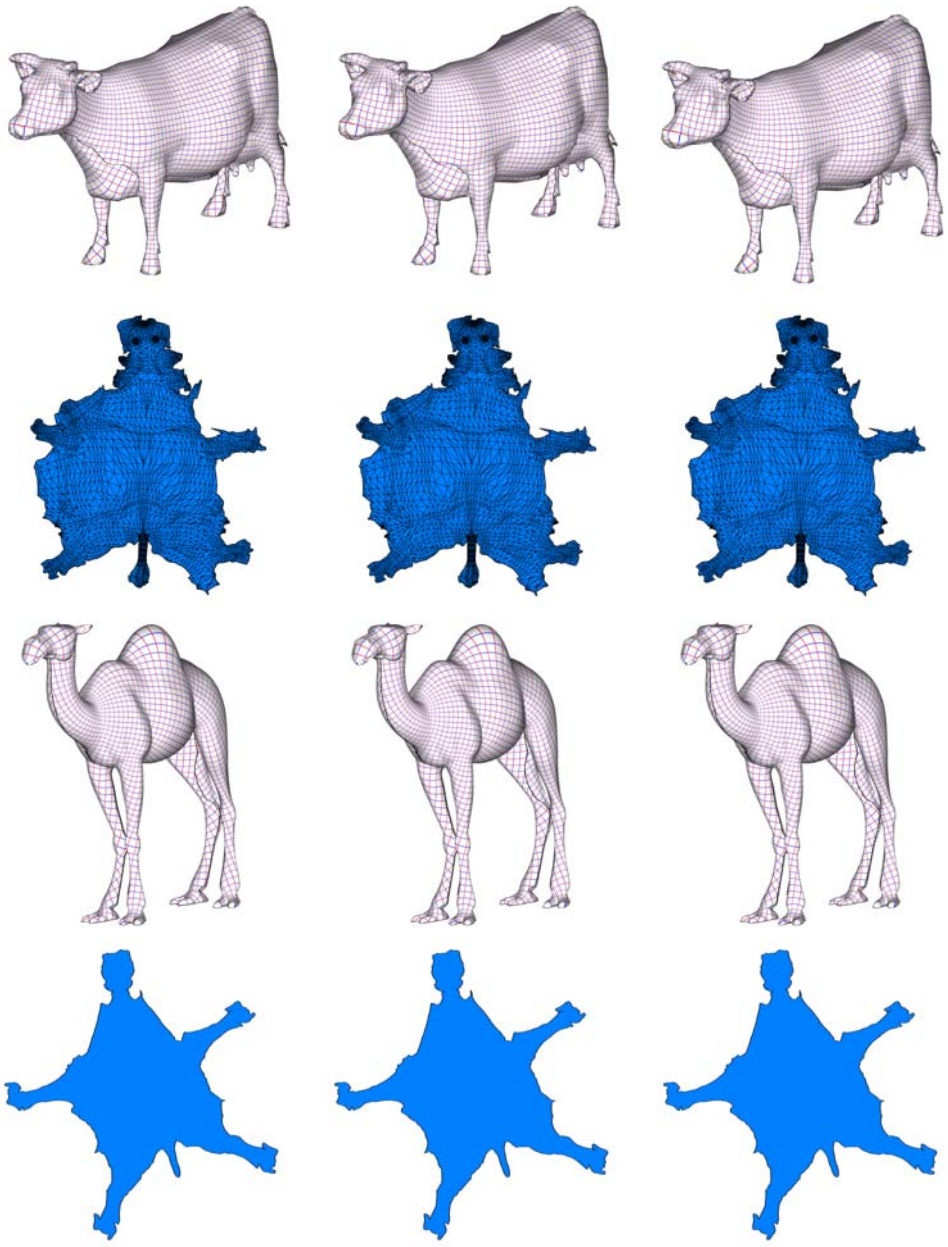
They search for angles that are as close as possible to the original 3D mesh angles β^t_k and which satisfy those constraints.

The constrained numerical optimization problem is solved using the Lagrange multipliers method (see section 10.6). The stationary point of the Lagrangian is computed using Newton's method (see Section 10.5). Each step requires to solve a linear system of size $2nv + 4nf$, where nv denotes the number of interior mesh vertices and nf the number of facets. They then convert the solution angles into actual (u, v) vertex coordinates using a propagation procedure. The resulting parameterizations are guaranteed to have no flipped triangles, i.e. be *locally* bijective, but can contain global overlaps. The authors provided a mechanism for resolving such overlaps, but it has no guarantees of convergence. The original ABF method is relatively slow and suffers from stability problems in the angle-to-uv conversion stage for large meshes.

ABF was augmented to yield ABF++ [Sheffer et al., 2005], a technique addressing both problems. ABF++ introduces a stable angle-to-uv conversion using the LSCM method to obtain (u, v) coordinates from the set of angles. Sheffer et al. [2005] also drastically speeds up the solution by introducing both direct and hierarchical solution approaches. For the direct solver they switch from Newton to Gauss-Newton solution setup, simplifying the structure of the linear system solved at each step. They then use the structure of the Hessian matrix to solve the linear system without explicitly inverting the entire Hessian. Thus they manage to reduce the size of the explicitly inverted matrix at each step by a factor of five to about $2nv$ (note that on a manifold mesh $2nv \approx nf$). This is what one may expect for a parameterization problem since this corresponds to the number of degrees of freedom. We make an interesting observation about the convergence behaviour of ABF+ throughout the Gauss-Newton process. While it takes five to ten iterations to reduce the minimization error (the gradient of the Lagrangian) below $1e-6$, practically all the changes from iteration two and on occur in the Lagrange multipliers. Even after one iteration (Figure 5.1(a)) the computed angles are very close to the final ones and they completely converge within a couple of iterations (Figure 5.1).

Several modifications of the formulation were proposed over the last few years. For instance, Zayer et al. [2003] introduced additional constraints on the angles enforcing the parameter domain to have convex boundaries, thus guaranteeing global bijectivity.

Kharevych et al. [2006] use a circle patterns approach where each circle corresponds to a mesh face. In contrast to classical circle packing, they use intersecting circles, with prescribed intersection angles θ_e (Figure 5.2). Given these angles, the circle radii follow as the unique minimizer of a convex energy. The method first computes the intersection angles using non-linear constrained optimization and then finds the circle radii using unconstrained minimization. To find the intersection angles it first computes a set of feasible triangle angles α^k_{ij} which are close to the 3D angles β^k_{ij} and satisfy the following



(a) (b) (c))

Figure 5.1: Convergence of ABF++: (a) parameterization result after one iteration (error metrics cow: $L_2^{stretch} = 1.10634$, $L_2^{shear} = 4.69e^{-4}$, $AngD = 4.19e^{-4}$ camel $L_2^{stretch} = 1.09616$, $L_2^{shear} = 5.21e - 3$, $AngD = 5.09e^{-3}$); (b) two iterations (error metrics cow: $L_2^{stretch} = 1.10641$, $L_2^{shear} = 4.64e^{-4}$, $AngD = 3.81e^{-4}$ camel $L_2^{stretch} = 1.09645$, $L_2^{shear} = 5.15e - 3$, $AngD = 5.04e^{-3}$ (d) ten iterations (error metrics cow: $L_2^{stretch} = 1.10638$, $L_2^{shear} = 4.64e^{-4}$, $AngD = 3.81e^{-4}$ camel $L_2^{stretch} = 1.09639$, $L_2^{shear} = 5.15e^{-3}$, $AngD = 5.04e^{-3}$). The formulas for the stretch, shear and angular distortion are taken from [Sheffer et al., 2005].

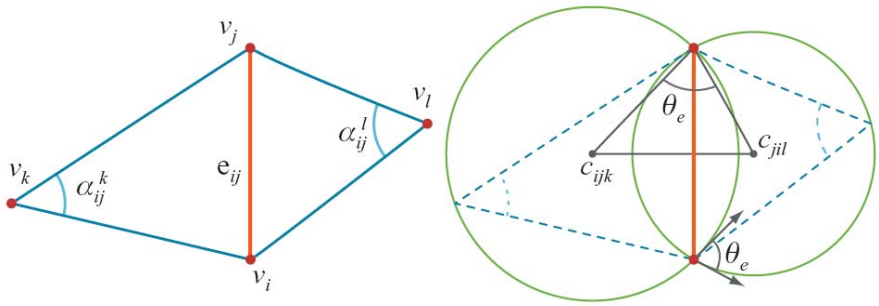


Figure 5.2: Circle patterns notations.

constraints:

- Triangle validity (for each triangle t):

$$\forall t_{ijk} \in T, \quad \alpha_{ij}^t + \alpha_{ik}^t + \alpha_{kj}^t - \pi = 0; \tag{5.4}$$

- Planarity (for each internal vertex v):

$$\forall v \in V_{int}, \quad \sum_{(t,ij) \in v^*} \alpha_{ij}^t - 2\pi = 0, \tag{5.5}$$

  where V_{int} denotes the set of internal vertices, and where v^* denotes the set of angles incident to vertex v.

- Positivity : $\alpha_{ij}^t > 0$ for all angles.

- Local Delaunay property (for each edge e_{ij}):

$$\forall e_{ij} \in E, \quad \alpha_{ij}^k + \alpha_{ij}^l < \pi \tag{5.6}$$

We note that three of these constraints are similar to those imposed by the ABF setup, with the non-linear reconstruction constraint on interior vertices replaced by the inequality local Delaunay constraint per edge. The intersection angles are computed from the feasible triangle angles using a simple formula. Since the solution for the intersection angles is conformal only for a Delaunay triangulation, the authors employ a pre-processing stage that involves "intrinsic" Delaunay triangulation. At the final stage of the method, the computed radii are converted to actual uv-coordinates. The method supports equality and inequality constraints on the angles along the boundary of the planar parameter domain. Similar to ABF, the parameterization is locally bijective, but can contain global overlaps. The amount of distortion introduced by the method is comparable with that of ABF/ABF++ techniques.

Kharevych et al. propose an extension of the method to global parameterization of meshes by introducing cone singularities. They observe that in angle space formulation

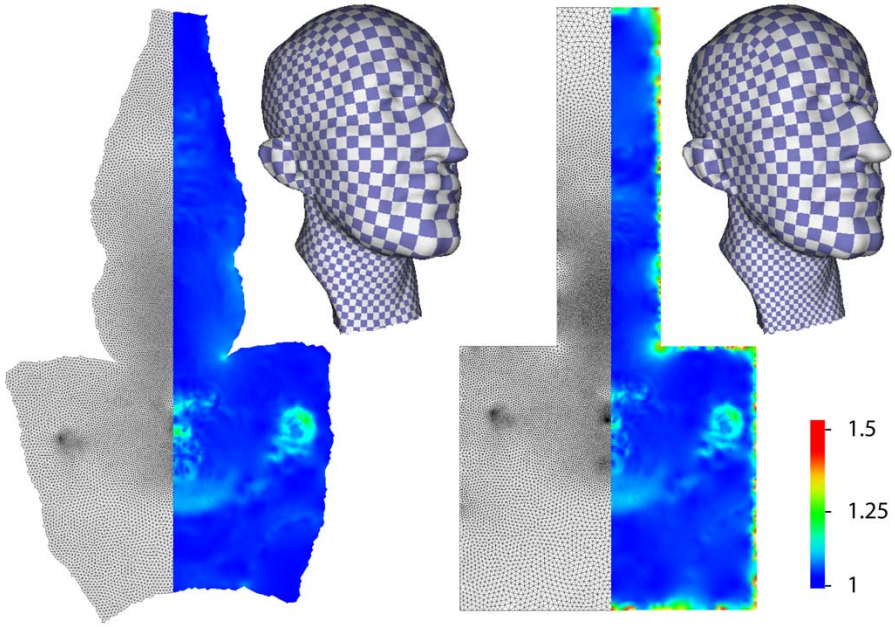


Figure 5.3: Fixed and free-boundary parameterizations computed using circle patterns.

the only difference between parameterizing the mesh boundary and its interior mesh is the constraints imposed on the interior vertices which are not imposed on boundary ones, and that it is possible to define a global parameterization by specifying an unconnected subset of mesh vertices as boundary vertices. Thus they first compute a solution in angle space with a set of cone singularity vertices specified by the user as boundary. For planar parameterization, to perform the angle-to-uv conversion they later compute edge paths between these vertices. The obtained parameterization is globally continuous up to translation and rotation, everywhere except at the cone singularities. The proposed approach can be directly applied to other angle-space methods such as ABF/ABF++.

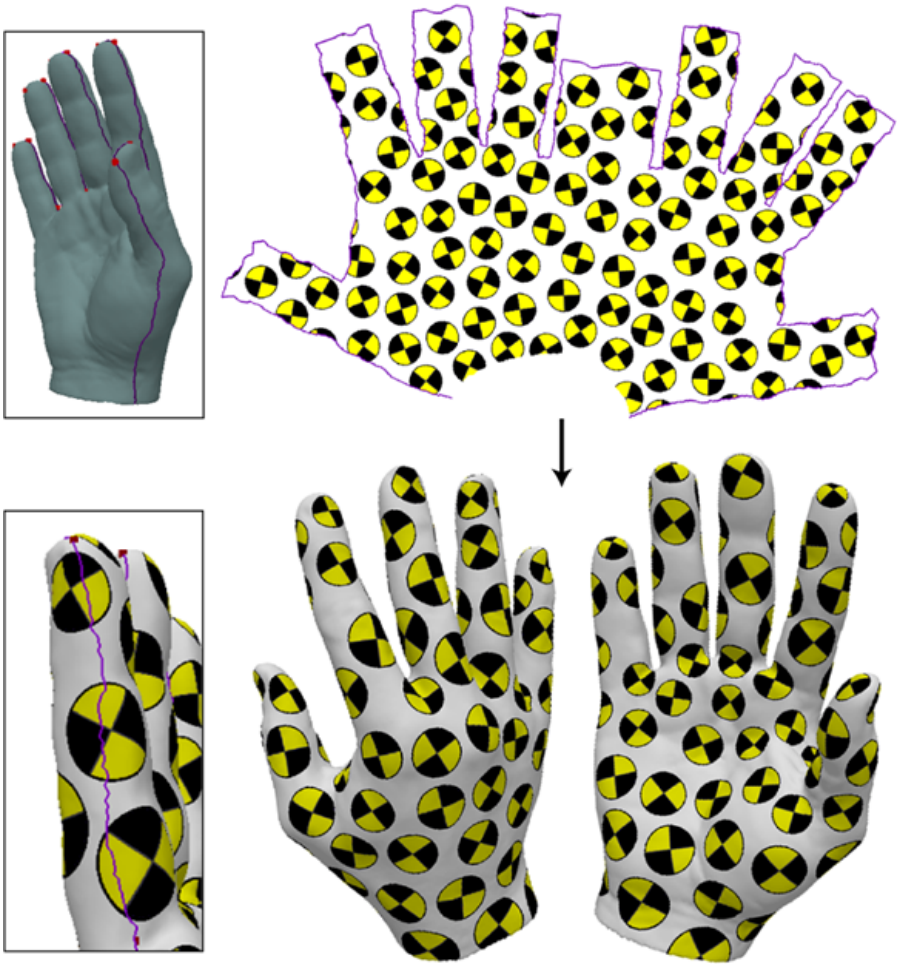


Figure 5.4: Global parameterization with cone singularities.

Chapter 6

Segmentation and Constraints[1]

For practical purposes planar mesh parameterization has to address two additional challenges: segmenting closed or high genus meshes to enable planar embedding and enforcing specific point-to-point correspondences during parameterization.

6.1 Segmentation

Planar parameterization is only applicable to surfaces with disk topology. Hence, closed surfaces and surfaces with genus greater than zero have to be cut prior to planar parameterization. As previously noted, greater surface complexity usually increases parameterization distortion, independent of the parameterization technique used. To allow parameterizations with low distortion, the surfaces must be cut to reduce the complexity. Since cuts introduce discontinuities into the parameterization, a delicate balance between the conflicting goals of small distortion and short cuts has to be achieved. It is possible to use constrained parameterization techniques to reduce cross-cut discontinuities.

Cutting and chart generation are most commonly used when computing parameterizations for mapping of textures and other signals onto the surface. They are also used for applications such as compression and remeshing. The techniques for cutting surfaces can be roughly divided into two categories: segmentation techniques which partition the surface into multiple charts (Section 6.1.1), and seam generation techniques which introduce cuts into the surface but keep it as a single chart (Section 6). Multiple charts created by segmentation typically have longer boundaries than those created by seam cutting. However, they can often be more efficiently packed into a compact planar domain.

6.1.1 Multi-Chart Segmentations

Depending on the application, mesh segmentation techniques use different criteria for creating charts. For parameterization, surfaces are broken into several charts such that the parametric distortion when parameterizing each chart is sufficiently low, while the number of charts remains small and their boundaries are kept as short as possible. Since

[1]This chapter is taken from [Sheffer et al., 2006] with minor changes.

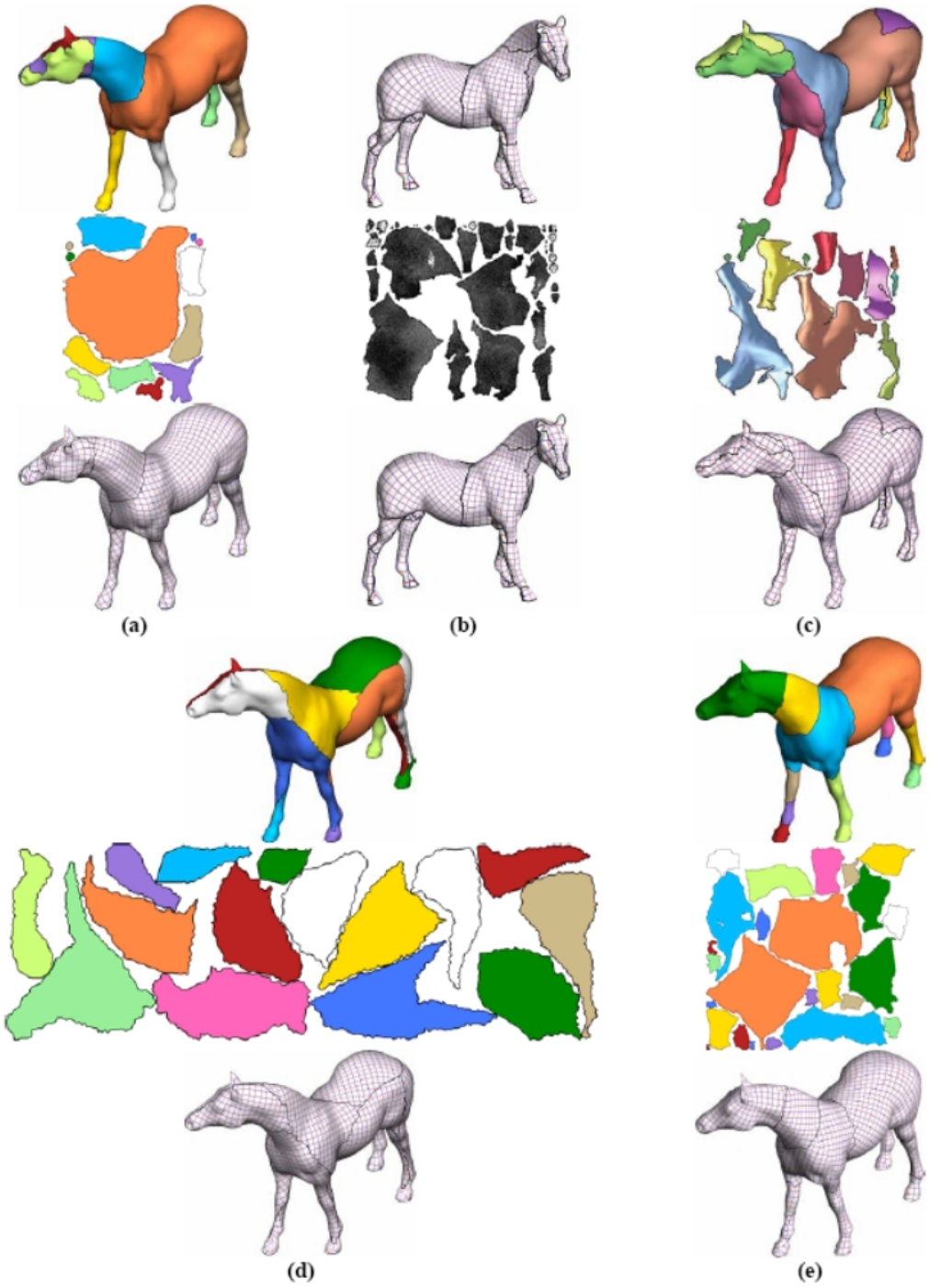


Figure 6.1: Multi-chart segmentations (a) Isocharts [Zhou et al., 2004] – 13 charts; (b) [Lévy et al., 2002] – 44 charts; (c) D-charts [Julius et al., 2005] – 12 charts; (d) multi-chart geometry images [Sander et al., 2003] – 15 charts; and (e) [Zhang et al., 2005] – 24 charts.

planes are developable by definition, one possible approach is to segment the surface into nearly planar charts [Maillot et al., 1993; Garland et al., 2001; Sander et al., 2003; Cohen-Steiner et al., 2004].

Recently, both Sander et al. [2003] and Cohen-Steiner et al. [2004] introduced planar segmentation methods inspired by Lloyd quantization. Their algorithms iterate between chart growing and reseeding stages. After each chart growing iteration the methods select the best seeds with respect to each chart and repeats the process. The authors demonstrate that by iterating, they obtain better results than single pass methods.

Planes are a special type of developable surfaces. Thus for parameterization purposes planar segmentation is over-restrictive and usually generates more charts than necessary. Several recent approaches focused on developable segmentation instead [Lévy et al., 2002; Zhou et al., 2004; Julius et al., 2005]. Lévy et al. [2002] proposed to detect high mean-curvature regions on the mesh and then generate charts starting from seeds which are farthest from those regions. This approach tends to capture many developable regions, but can also introduce charts which are far from developable.

Zhou et al. [2004] propose a segmentation method based on spectral analysis of the surface. They compute a matrix of geodesic mesh distances and then perform face clustering by growing charts around the n farthest points in a space defined by the dominant eigenvectors of the matrix.

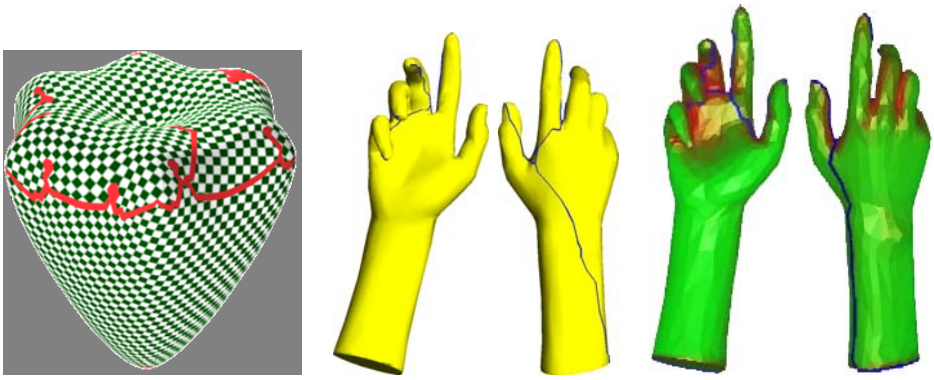
Zhang et al. [2005] segment the surface into feature regions by finding field iso-contours where the feature area increases significantly; then the features are classified as linear ellipsoids, flat ellipsoids, or spheres, and cut using a lengthwise, circular, or respectively a "baseball seam" cut. The resulting pieces are close to being developable and can be parameterized with little distortion.

Similar to [Sander et al., 2003], Julius et al. [2005] use Lloyd iterations of growing and reseeding. But instead of looking for planar regions they search for a larger subset of developable surfaces, the so called “developable surfaces of constant slope”, which are characterized by having a constant angle between the normal to the surface and some axis vector. They provide a metric to measure if a chart closely approximates such a surface, and use this metric in the chart growing and reseeding stages.

Chart packing: Chartification techniques raise an additional post-processing challenge. Following the parameterization of each individual chart, those charts need to be placed, or packed, in a common parameter domain. For efficient storage of the parameterized meshes, the packing has to be as compact as possible. The optimal packing problem is NP-hard, thus only heuristic or approximate packing algorithms exist. The Tetris algorithm [Lévy et al., 2002] introduces charts one by one, searching for the best fit along the active-front of the charts packed so far.

6.1.2 Seam Cutting

It is possible to reduce the parameterization distortion without cutting the surface into separate patches by introducing multiple partial cuts or seams inside a single patch. This typically leads to shorter cuts than those created by segmentation. Piponi and Borshukov [2000] generated such cuts manually using a network of edges.



(a) (b) (c)

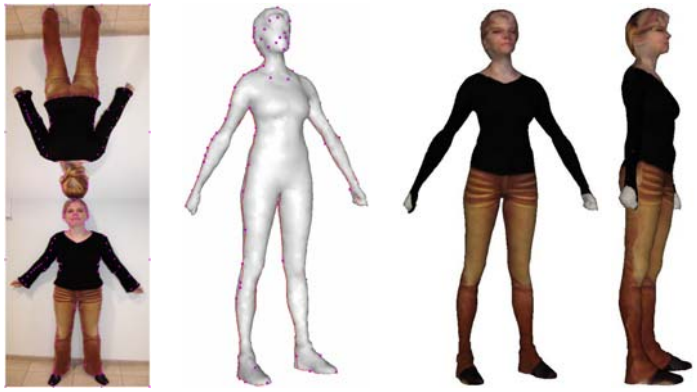
Figure 6.2: Seam cutting: (a) [Sorkine et al., 2002]; (b) [Gu et al., 2002]; (c) [Sheffer and Hart, 2002] - the color visualizes the surface visibility with more visible regions shown in green and less visible in red.

Sorkine et al. [2002] perform parameterization and cutting simultaneously. They unfold the mesh vertices onto the plane one after the other, optimizing the local mapping. Whenever the distortion of the mapping reaches a threshold, they cut the mesh to reduce it. As a result they have a hard bound on the distortion, but can end up with long and complicated boundaries. To measure distortion, they use the singular values of the per-triangle mapping.

Gu et al. [2002] use parameterization results to facilitate the cutting process. The authors first parameterize the surface using shape preserving parameterization [Floater, 1997]. They then find the point of maximal parametric distortion on the mapping, and generate the shortest cut from the surface boundary to that point. They repeat the process until the distortion falls below a certain threshold.

The Seamster algorithm [Sheffer and Hart, 2002] considers the differential geometry properties of the surface, independent of a particular parameterization technique. It first finds regions of high Gaussian curvature on the surface and then uses a minimal spanning tree of the mesh edges to connect those. Finally it cuts the mesh along the tree edges. Sheffer and Hart [2002] scale the edge length by a visibility metric when computing the minimal spanning tree. This way they are able to trace the cuts through the less visible parts of the surface hiding the potential cross-cut discontinuities in texture or other maps on the surface.

Genus reduction Seam cutting methods require an explicit preprocessing stage to convert surfaces with high genus into topological disks. The generation of minimal length cuts that convert a high genus surface into a topological disk is NP-hard [Erickson and Har-Peled, 2004]. This problem had received a lot of attention on the computational geometry community with a number of methods proposed which operate with differing levels of success. A fairly practical approach is taken by Gu et al. [2002], who trace a spanning graph of all the faces in the mesh and then prune this graph, obtaining a genus reducing cut.



(a) (b) (c)

Figure 6.3: Constraint enforcement with Matchmaker [Kraevoy et al., 2003]: (a) Texture (two photos) with feature points specified; (b) input model with corresponding vertices highlighted; (c) resulting texture.

(a) (b) (c)

Figure 6.4: Combining multiple textures [Zhou et al., 2005].

Figure 6.5: Two examples of constrained texture mapping [Lévy, 2001]

6.2 Constraints

Sometimes a parameterization needs to accommodate user constraints, specifying correspondences between vertices of the mesh. The most important application of constrained parameterization is texture mapping 3D models from photographs [Lévy, 2001; Kraevoy et al., 2003] (see Figures 6.5 and 6.3). Zhou et al. [2005] use more complex constraints to allow the user to combine several images to produce a complete texture for a mesh. Constrained parameterization can also be used to hide cross-seam discontinuities [Kraevoy et al., 2003; Zhou et al., 2005].

The methods for enforcing constraints can be split into two types, those that enforce soft or approximate constraints and that that enforce hard constraints. Methods based on energy minimization can accommodate soft constraints by adding a quadratic term to the energy function, measuring the distance between the constraint features in the current configuration and their desired location [Lévy, 2001]. Such constraints work reasonably well in practice, and can be solved efficiently, since they only add linear terms to the energy, but sometimes break theoretical guarantees about the original parameterization method, such as bijectivity. The degree of constraint approximation typically decreases with the distance between the constrained and unconstrained vertex locations. Some applications, such as hiding texture discontinuities along seams in the parameterization [Kraevoy et al., 2003; Zhou et al., 2005] require hard constraints to achieve perfect alignment of the texture along the seams.

One way to enforce hard constraints is by adding them into a regular parameterization formulation using Lagrange multipliers [Desbrun et al., 2002]. This approach allows constraints to be defined on points inside triangles and on arbitrary line segments (the vertices of the triangulation can be constrained more easily by taking the corresponding variables out of the system). However, it is easy to show that for a given mesh connectivity not every set of constraints can be satisfied. Thus methods like this that preserve the mesh connectivity will fail to generate bijective parameterizations for many inputs.

Methods that enforce hard-constraints robustly, introduce additional vertices into the mesh as they go along to ensure that a constrained solution exists. Eckstein et al. [2001] enforce hard constraints by deforming an existing embedding while adding new vertices when necessary. Theoretically, this method can handle large sets of constraints but is extremely complicated.

The Matchmaker algorithm [Kraevoy et al., 2003] compute the parameterization by establishing coarse patch correspondences between the input and the parameter domain. The provided feature points on the input model and the parameter domain are connected using a network of curves that partition the surface into patches that are then parameterized while trying to maintain continuity and smoothness between them. The curve tracing process is guided by a set of topological rules that ensure that the resulting patches will be consistent between the objects being parameterized and the domain. They compute the triangulations of the input and the parameter domain simultaneously. Continuity and smoothness between patches can be obtained by relaxing the parameterization.

Zhou et al. [2005] allow the user to combine several images to produce texture for a

mesh, by assigning some surface patches to different images, as well as using in-painting techniques to create texture for any unassigned transition patches between them. In addition to geometric smoothness of the map, they take into account the continuity of the texture signal being applied since it may come from different sources for two neighboring patches.

Chapter 7

Comparison of Planar Methods

The different methods reviewed minimize different types of distortion metrics. Ideally, most parameterization applications work best on zero distortion parameterizations, though most are tolerant to some amount of distortion, some being more tolerant to shear and others to stretch. In general, applications that depend on regular grids for sampling, such as different types of detail mapping and synthesis, as well as compression and regular resampling schemes (e.g. geometry images [Gu et al., 2002]), tend to perform better on stretch minimizing parameterizations, since stretch is directly related to under-sampling. In contrast, applications based on irregular sampling, such as remeshing [Desbrun et al., 2002], are very sensitive to shearing, but can handle quite significant stretch. When acceptable levels of shear or stretch are not attainable because a surface is too complex, the surface needs to be cut prior to parameterization in order to achieve acceptable distortion.

In addition to distortion, several other factors should be considered when choosing a parameterization method for an application at hand:

- *Free versus fixed boundary.* Many methods assume the boundary of the planar domain is pre-defined and convex. Fixed boundary methods typically use very simple formulations and are very fast. Such methods are well suited for some applications, for instance those that utilize a base mesh parameterization, see Section 5.1. Free-boundary techniques, which determine the boundary as part of the solution, are often slower, but typically introduce significantly less distortion.

- *Robustness.* Most applications of parameterization require it to be bijective. For some applications local bijectivity (no triangle flips) is sufficient while others require global bijectivity conditions (the boundary does not self-intersect). Only a subset of the parameterization methods can guarantee local or global bijectivity. Some of the others can guarantee bijectivity if the input meshes satisfy specific conditions.

- *Numerical Complexity.* The existing methods can be roughly classified according to the optimization mechanism they use into linear and non-linear methods. Linear methods are typically significantly faster and simpler to implement. However, as expected the simplicity usually comes at the cost of increased distortion.

Method	Distortion minimized	Boundary	Bijectivity	Complexity
Uniform [Tutte 1963]	None	Fixed, convex	Yes	Linear
Harmonic [Eck 1995]	Angles	Fixed, convex	No	Linear
Shape preserving [Floater 1997]	Angles	Fixed, convex	Yes	Linear
Mean-value [Floater 2003]	Angles	Fixed, convex	Yes	Linear
LSCM/DCP [Lévy et al. 2002, Desbrun et al. 2002]	Angles (&Area)	Free	No	Linear
ABF/ABF++ [Sheffer and de Sturler 2000, Sheffer et al. 2005]	Angles	Free	Local (no flips)	Non-linear
MIPS [Hormann and Greiner, 2000]	Angles	Free	Global	Non-linear
Circle Patterns [Kharevych et al. 2006]	Angles	Free	Local (no flips)	Non-linear (unique minimum)
Stretch minimizing [Sander et al. 2001]	Distances	Free	Global	Non-linear
MDS [Zigelman et al. 2002]	Distances	Free	No	Non-linear
[Degener et al. 2003]	Areas	Free	Yes	Non-linear

Table 1: Planar method summary.

Method	Cat Head (257 Δ)	Nefertiti (8071 Δ)	Cow (5804 Δ)	Camel (78K Δ)
Uniform [Tutte 1963]*	0.02 s.	0.23 s.	0.14 s.	2.91 s.
Harmonic [Eck et al. 1995]*	0.02 s.	0.26 s.	0.17 s.	3.21 s.
Mean-value [Floater 2003]*	0.02 s.	0.25 s.	0.16 s.	3.19 s.
LSCM [Lévy et al. 2002]*	0.03 s.	0.38 s.	0.20 s.	5.28 s.
ABF++ [Sheffer et al. 2005]*	0.06 s.	1.87 s.	0.77 s.	36.31 s.
MIPS [Hormann and Greiner, 2000]‡	1 s.	8 s.	5 s.	83 s.
Circle Patterns [Kharevych et al. 2006]*	0.1 s.	3.6 s.	2.1 s.	76.7 s.
Stretch minimizing [Sander et al. 2001]	5.17 s.†	12 s.	8.5 s.	127 s.

†1.2 GHz Pentium M *3.0 GHz Pentium 4 ‡3.2 GHz Pentium 4

Table 2. Timings of various parameterization techniques.

Table 1 summarizes the more commonly used methods in terms of these properties. Table 2 lists the runtimes for some of the more commonly used methods. We used the Ray et al. [2003] 3D modeling system to time the fixed boundary methods, LSCM and ABF++. For the other methods the timings were provided by the authors. As expected, linear techniques are about one order of magnitude faster than the non-linear ones. Nevertheless, even the non-linear methods are fairly fast taking less than two minutes to process average size models. Figures 7.1 to 7.3 show some typical parameterization results.

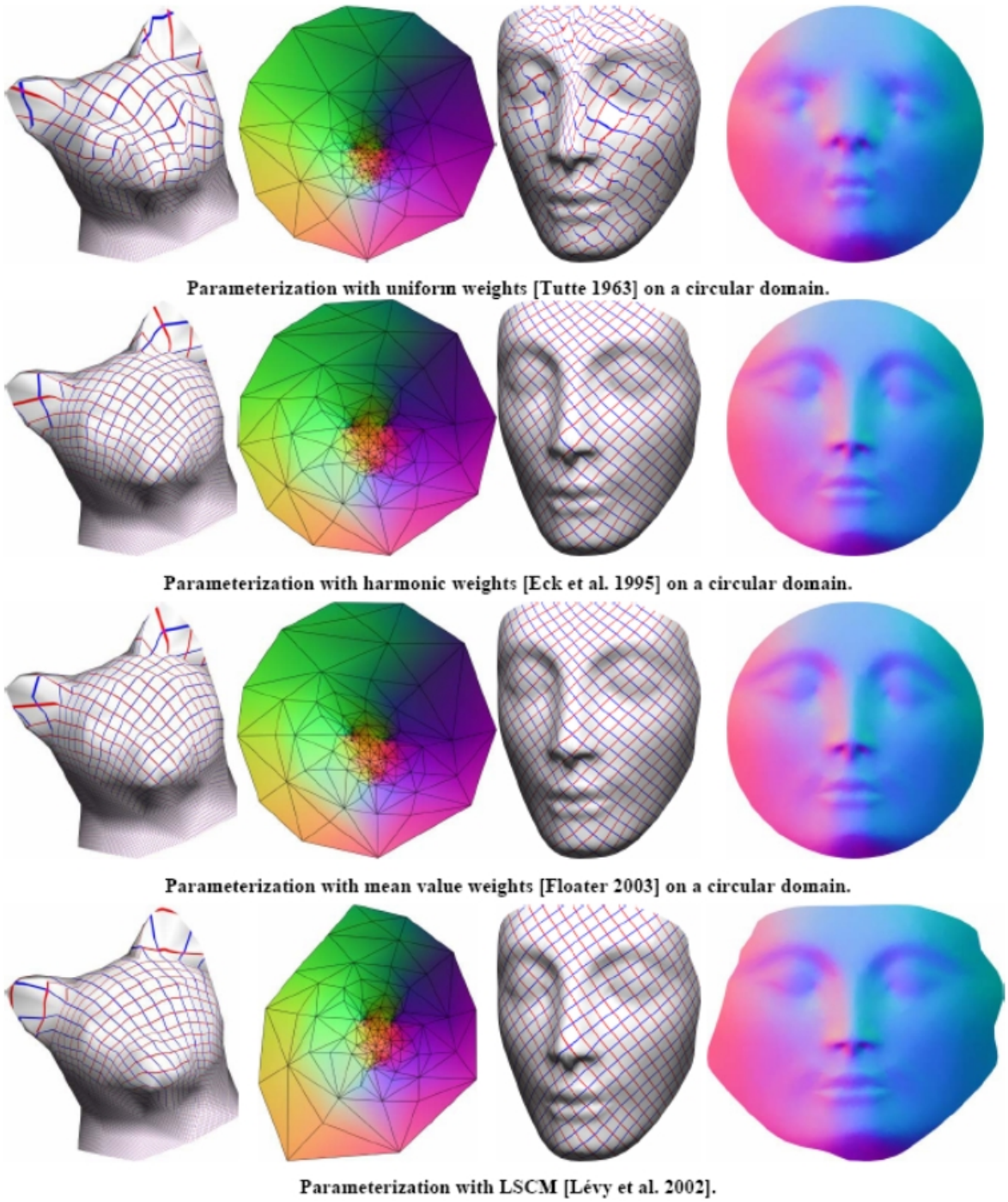


Figure 7.1: Surfaces with nearly convex boundaries parameterized with linear methods (images made with Graphite, http://alice.loria.fr/software).

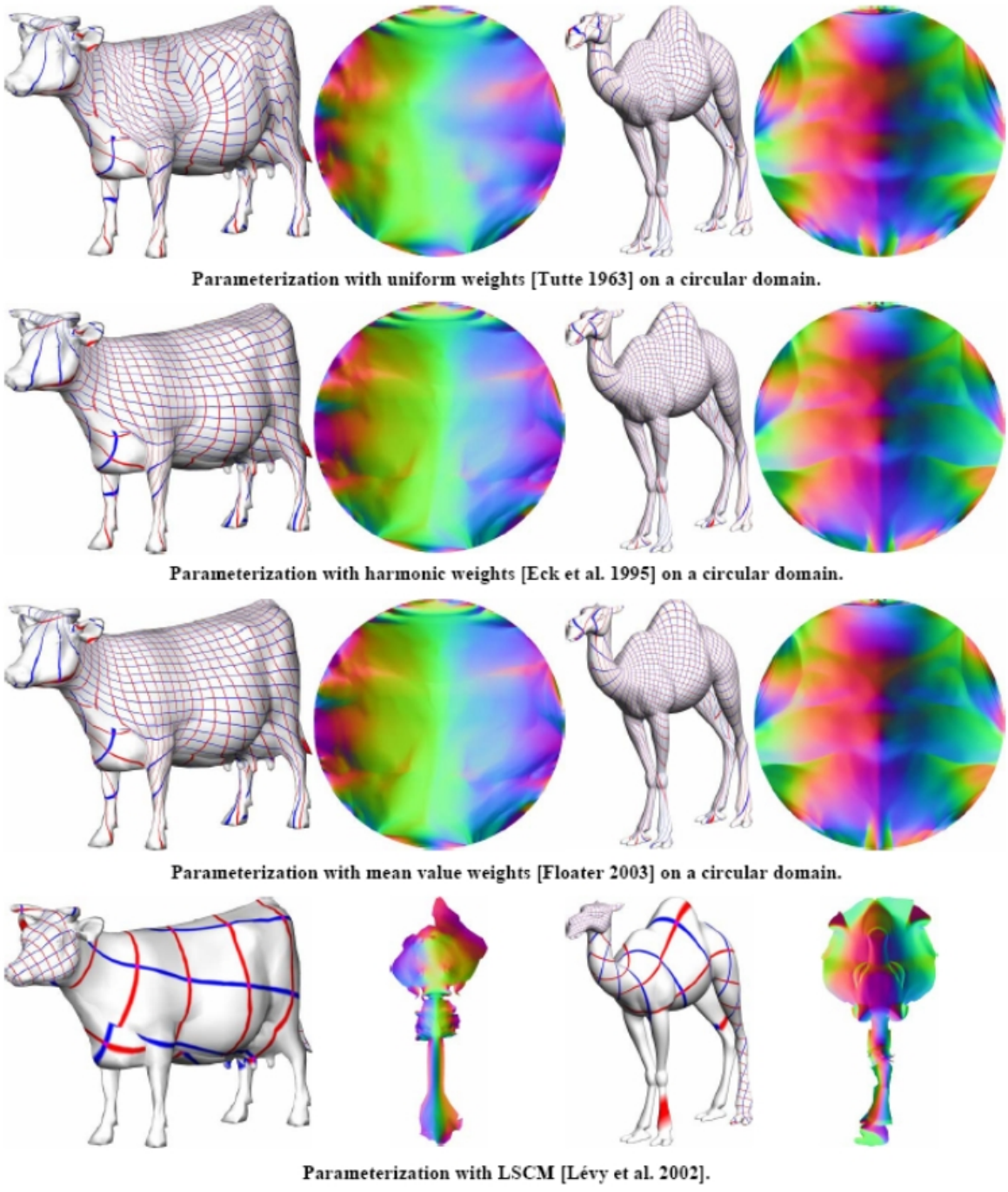


Figure 7.2: Surfaces with non-convex boundaries parameterized with linear methods (images made with Graphite, http://alice.loria.fr/software).

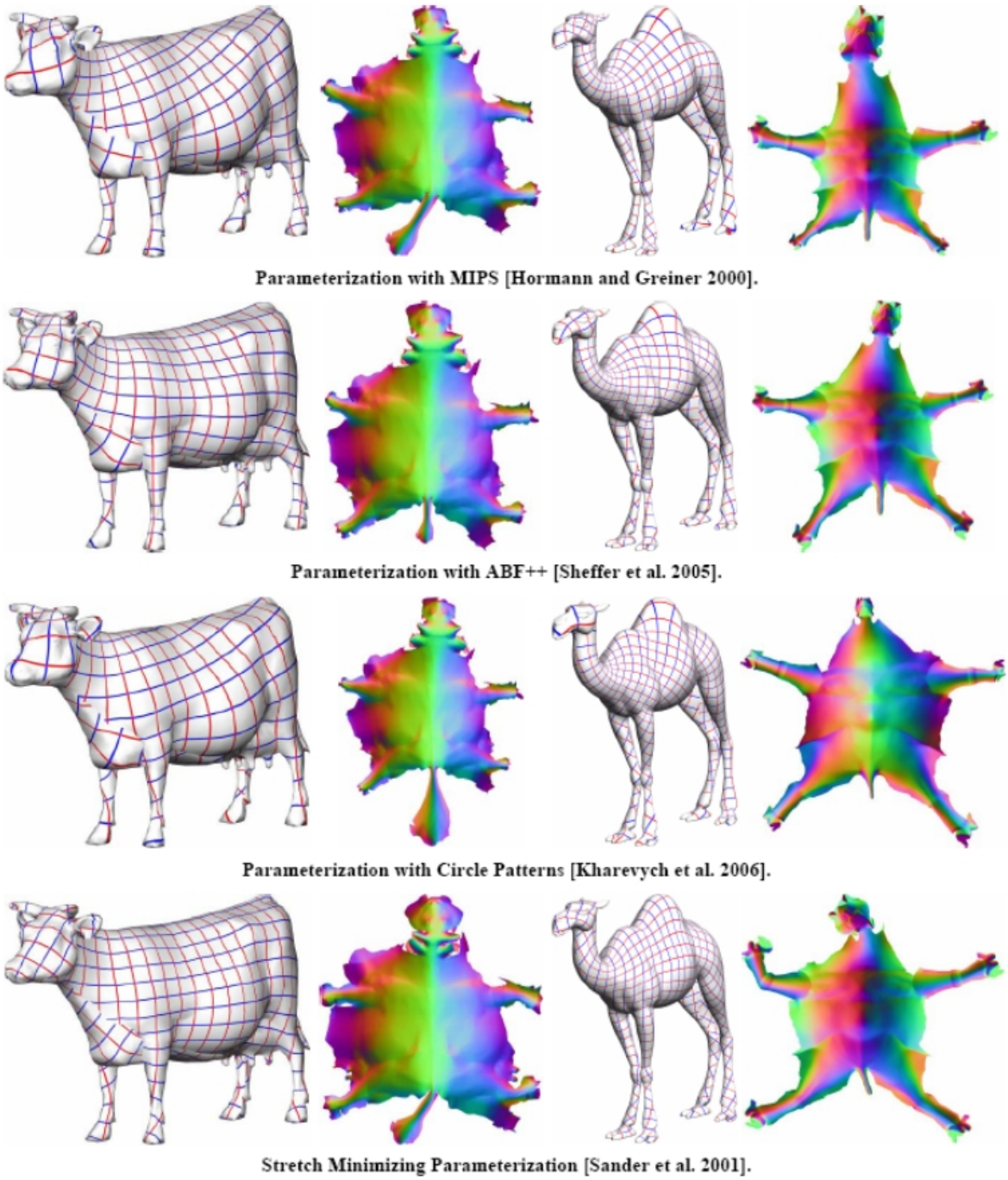


Figure 7.3: Surfaces with non-convex boundaries parameterized with non-linear methods (images made with Graphite, http://alice.loria.fr/software).

Chapter 8

Alternate base domains

Some applications are quite sensitive to discontinuities in the parameterization, or cannot tolerate them at all. In such cases, when the object to be parameterized is not a topological disc, it is worthwhile to use a different base domain for the parameterization. Examples of such domains that have been investigated include simplicial complexes, spheres, and periodic planar regions with transition curves.

In addition, numerous applications of parameterization require cross-parameterization or intersurface mapping between multiple models. Pair-wise mapping between models can be used for the transfer of different properties between the models, including straightforward ones, such as texture, and less obvious ones such as deformation and animation. It can also be used for blending and morphing, as well as mesh completion and repair. The most common approach for pair-wise mapping is to parameterize both models on a common base domain. Free-boundary planar parameterization is clearly unsuitable for this purpose. Instead alternate domains such as a simplicial complex or a sphere are commonly used.

8.1 The Unit Sphere

The big advantage of the spherical domain over the planar one is that it allows for seamless, continuous parameterization of genus-0 models, and there are a large number of such models in use. Thus, the spherical domain has received much attention in the last few years, with several papers published about this topic. Some rigorous theory is being developed, getting close to the level of understanding we have of planar parameterizations. These notes cover four main types of spherical parameterization approaches: Gauss-Seidel iterative extension of planar barycentric methods; stereographic projection; spherical generalization of barycentric coordinates; and multi-resolution embedding.

One attractive approach for spherical parameterization is to extend the barycentric, convex boundary planar methods to the sphere. Several methods [Alexa, 2000; Gu and Yau, 2002; Kobbelt et al., 1999] used Gauss-Seidel iterations to obtain such parameterization. They start by computing an initial guess and then moving the vertices one at a time, first computing a 3D position for the vertex using a barycentric formulation [Eck et al., 1995], and then projecting the vertex to the unit sphere. Isenburg et al. [2001]

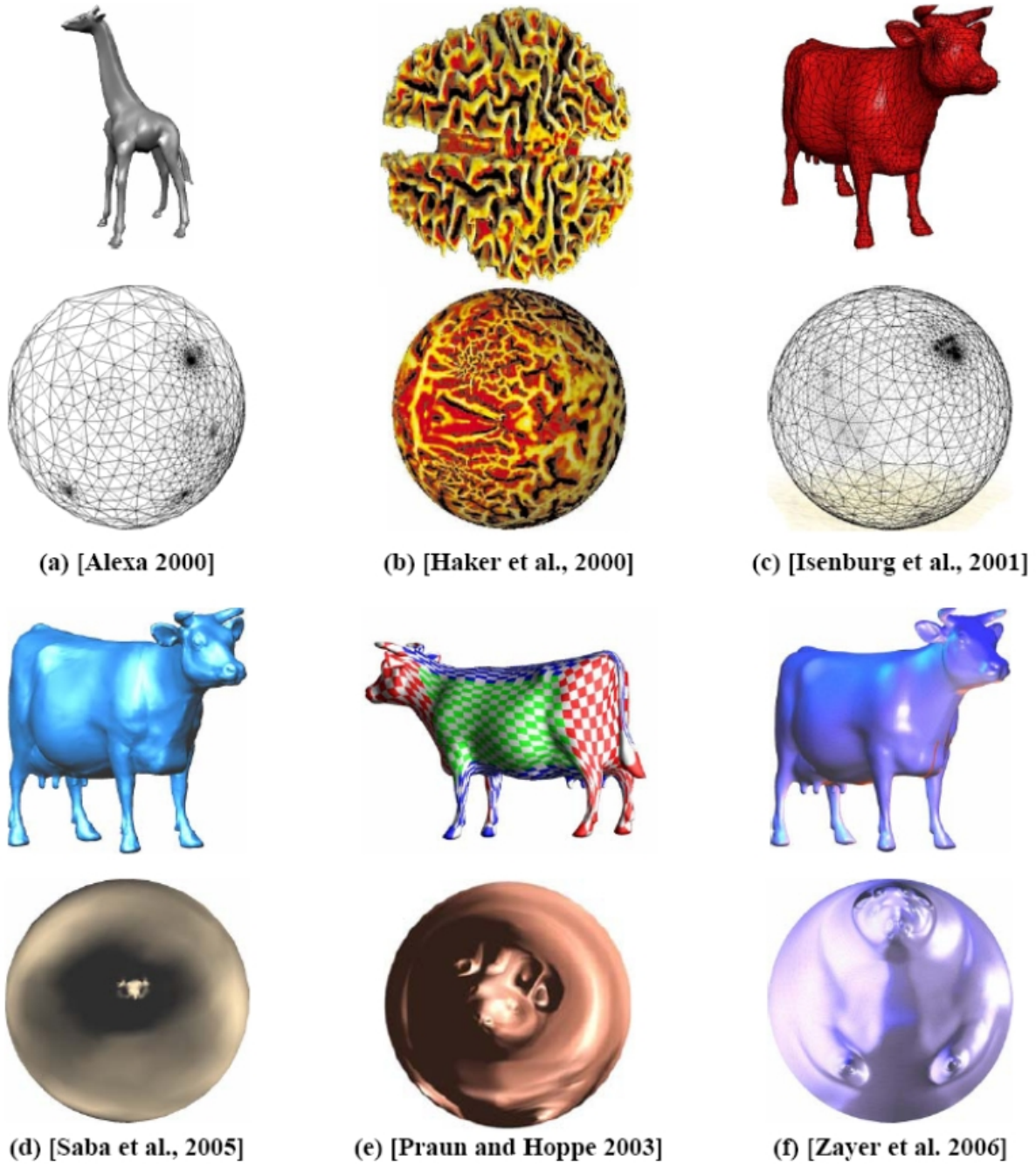


Figure 8.1: Spherical Parameterization.

split the mesh in two, map the cut onto a great circle and embed each half-mesh onto a hemisphere using a modified Tutte procedure. Regrettably, as proven by Saba et al. [2005] projected Gauss-Seidel iterations decrease the residual for only a finite number of iterations. As the result approaches a bijective solution, the scheme ultimately becomes unstable, the residual increases, and the system collapses to a degenerate solution. Saba et al. [2005] note that this behaviour is independent of step size.

Haker et al. [2000] compute a planar parameterization of the mesh first, using one of the triangles as a boundary. They then use the stereographic projection to obtain the spherical mapping. The result depends quite heavily on the choice of the boundary triangle. This approach works quite well in practice, however it doesn't offer any theoretical guarantees since the stereographic projection is bijective only for the continuous case, and can produce triangle flips in the discrete case. A simple proof by example of this statement can be obtained by imagining the great circle supporting the edge AB of a mapped spherical triangle ABC. The (continuous) stereographic projection maps this great circle to a circle in the original plane. The third vertex C can be perturbed in the plane to cross from the interior to the exterior of the circle, without changing the triangle orientation. The spherical triangle ABC will flip however as a result of this perturbation, as the image of C on the sphere will cross from one side to the other of the spherical edge AB.

Gotsman et al. [2003] showed how to correctly generalize the method of barycentric coordinates, with all its advantages, to the sphere. The generalization is based on results from spectral graph theory due to de Verdière [1993] and extensions due to Lovász and Schrijver [1999]. They provide a quadratic system of equations which is a spherical equivalent of the barycentric formulation. The authors do not provide an efficient way to solve the resulting system, and thus their method is limited to very small meshes. Saba et al. [2005] introduce a method for efficiently solving the system, by providing a good initial guess and using a robust solver. First, similar to [Isenburg et al., 2001], they partition the mesh in two, and embed each half on a hemisphere using a planar parameterization followed by a stereographic projection. They then use a numerical solution mechanism which combines Gauss-Seidel iteration with nonlinear minimization to obtain the final solution.

An efficient and bijective alternative is suggested by multi-resolution techniques. These methods obtain an initial guess by simplifying the model until it becomes a tetrahedron (or at least, convex), trivially embed it on the sphere, and then progressively add back the vertices [Shapiro and Tal, 1998; Praun and Hoppe, 2003]. Shapiro and Tal [1998] compute the embedding using purely topological operations and do not attempt to minimize any type of distortion. Praun and Hoppe [2003] obtain a spherical parameterization by alternating refining steps that add vertices from a multi-resolution decomposition of the object with relaxation of single vertex locations inside their neighbourhoods. The relaxation is aimed to minimize the stretch metric of the parameterization and is guaranteed to maintain a valid embedding.

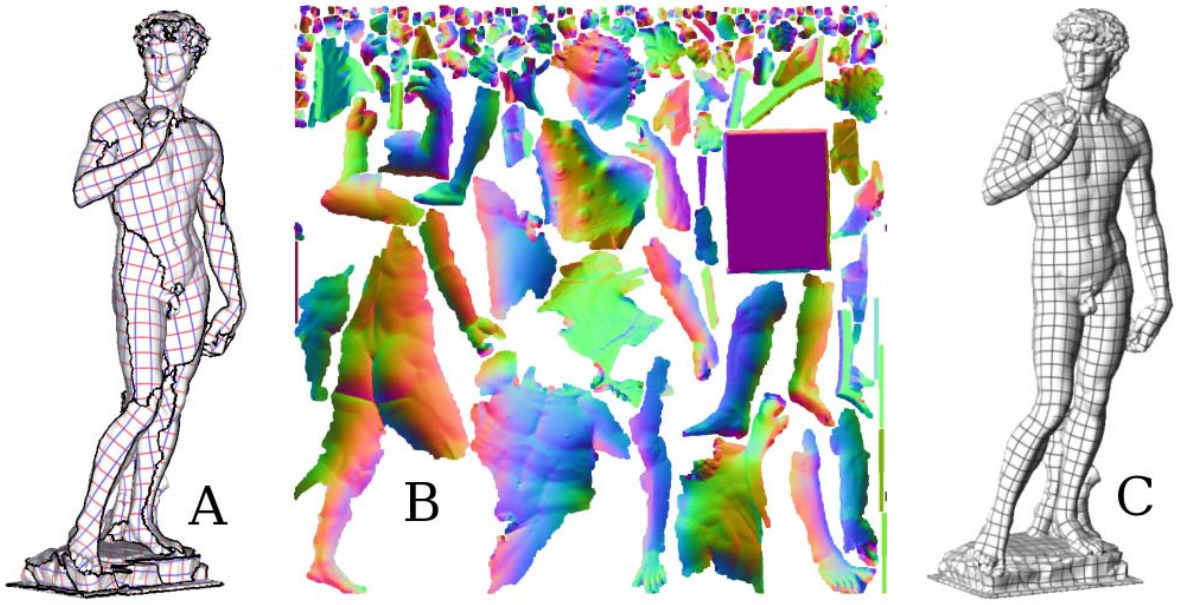


Figure 8.2: A,B: Parameterization methods for disk-topology combined with segmentation algorithms can create a texture atlas from a shape of arbitrary topology. However, the large number of discontinuities can be problematic for the applications. C: Global parameterization algorithms do not suffer from this problem. (Data courtesy of the Digital Michelangelo Project, Stanford).

8.2 Simplicial and quadrilateral complexes

As seen in Chapters 3 and 4, parameterization methods can put a 3D shape with disk topology in one-to-one correspondence with a 2D domain. For a shape with arbitrary topology, it is possible to decompose the shape into a set of charts, using a segmentation algorithm (e.g. VSA [Cohen-Steiner et al., 2004]). Each chart is then parameterized (see Figure 8.2-A,B). Even if this solution works, it is not completely satisfactory : why one should "damage" the surface just to define a coordinate system on it ? From the application point of view, chart boundaries are difficult to handle in remeshing algorithms, and introduce artefacts in texture mapping applications. For this reason, we focus in this section on *global* parameterization algorithms, that do not require segmenting the surface. (Figure 8.2-C).

To compute such a global parameterization, the geometry processing community first developed methods that operate by segmenting / prameterizing / and resampling the object. To our knowledge, this idea was first developed in the MAPS method [Lee et al., 1998] (Multiresolution Adaptive Parameterization of Surfaces). As shown in Figure 8.3, this method starts by partitioning the initial object (Figure 8.3-A) into a set of triangular charts, called the *base complex* (Figure 8.3-C). Then, a parameterization of each chart is computed, and the object is regularly resampled in parametric space (Figure 8.3-C). Further refinements of the method improved the inter-chart continuity [Kho-

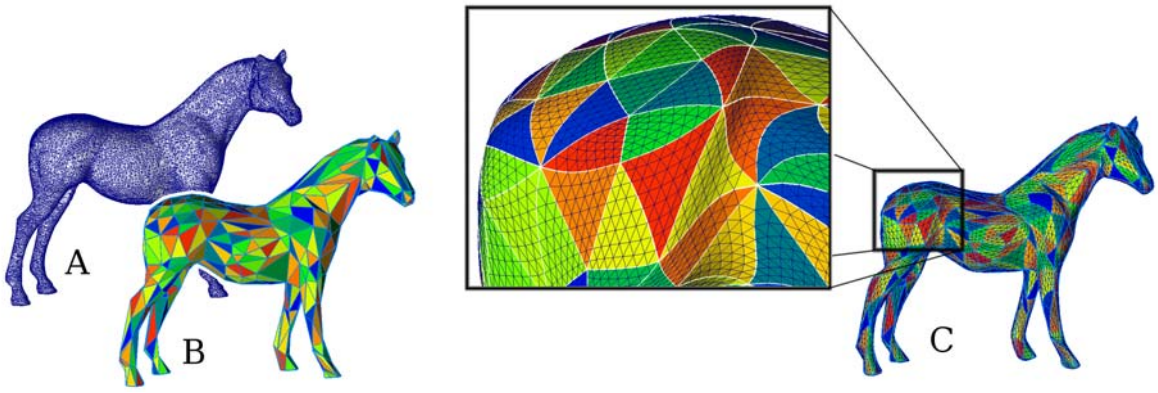


Figure 8.3: The MAPS method and its derivatives compute a global parameterization by decomposing the initial surface (A) into a set of triangular charts (B) and regularly re-samples the geometry in the parameter space of these charts (C).

dakovsky et al., 2003], formalized by the notion of *transition function*, explained further in this section. This representation facilitates defining hierarchical representations and implementing multiresolution processing tools on top of it [Guskov et al., 1999].

Historically, the most popular non-planar base domain has been a simplicial complex [Lee et al., 1998, 1999; Guskov et al., 2000, 2002; Lee et al., 2000; Praun et al., 2001; Khodakovsky et al., 2003; Purnomo et al., 2004; Schreiner et al., 2004; Kraevoy and Sheffer, 2004]. A simplicial complex can be considered as just the connectivity part of a traditional triangle mesh: the sets of vertices, edges, and faces. Most applications typically use simplicial complexes representing 2-manifolds with a boundary (an edge can only be adjacent to 1 or 2 faces) with a small number of elements. One method for obtaining such complexes is to simplify an original mesh. Once a suitable base mesh has been chosen, the original mesh is parameterized by assigning each of its vertices to a simplex of the base domain (vertex, edge, or face), along with barycentric coordinates inside it.

Early methods took a two-step approach to computing a parameterization; in the first step, elements of the fine mesh were assigned to faces of the base simplicial complex, while the second step would compute barycentric coordinates for these elements, usually using one of the fixed boundary parameterization methods discussed earlier. These steps could be repeated, but typically not mixed. More recent methods, such as [Khodakovsky et al., 2003], try to perform both steps at the same time.

8.2.1 Computing base complexes

To obtain the simplicial complex, [Eck et al., 1995] grow Voronoi regions of faces from seed points and then use the dual triangulation. The seed points are initially linked using shortest paths across mesh edges that provide the initial boundaries of the patches corresponding to base domain faces. To straighten each of these paths, the two adjacent patches are parameterized to a square. The path in question is then replaced with the

diagonal of the square mapped onto the mesh surface.

[Lee et al., 1998] simplify the original mesh, keeping track of correspondences between the original vertices and the faces of the simplified mesh. Others, like [Guskov et al., 2000] (Figure 13 (c)) and [Khodakovsky et al., 2003] use clustering techniques to generate the patch connectivity and derive the base-mesh from it.

The construction becomes more challenging when multiple models need to be parameterized on the same complex [Praun et al., 2001; Schreiner et al., 2004; Kraevoy and Sheffer, 2004]. Praun et al. [2001] partition a mesh into triangular patches, which correspond to the faces of a user given simplicial complex, by drawing a network of paths between user-supplied feature vertices that correspond to the vertices of the base mesh.

Schreiner et al. [2004] and Kraevoy and Sheffer [2004] extend the methods of Praun et al. [2001] and Kraevoy et al. [2003] to construct the simplicial complex automatically, in parallel to the patch formation. The input to both methods includes a set of correspondences between feature vertices on the two input models. The methods use those as the vertices of the base complex. They simultaneously trace paths on the input meshes between corresponding pairs of vertices, splitting existing mesh edges if necessary. Tarini et al. [2004] were the first, to our knowledge, to use a quadrilateral base domain. Such a domain is much more suitable for quadrilateral remeshing of the input surface and for spline fitting. Tarini et al. [2004] generate the base domain manually.

8.2.2 Mapping to the base mesh

Once the discrete assignment to base domain faces has been done, the barycentric coordinates can be computed using fixed-boundary planar parameterization. Earlier methods computed the barycentric coordinates once, based on the initial assignment of the vertices to the base triangles. More recent methods [Khodakovsky et al., 2003; Kraevoy and Sheffer, 2004, 2005; Tarini et al., 2004] use an iterative process where vertices can be reassigned between base faces.

Khodakovsky et al. [2003] perform the vertex-to-patch assignment and coordinate relaxation in a single procedure, by letting vertices cross patch boundaries using transition functions. A transition function expresses the barycentric coordinates of a vertex with respect to a base domain face as barycentric coordinates for a neighboring base domain face. For this procedure only the images of the base domain vertices needs to be fixed, rather than the edges as well as in the previous methods. The authors relax the base domain vertices separately, prompting a new run of the main relaxation. In practice, this cycle is repeated only very few times. The implementation sometimes needs to discard some relaxation results when mesh vertices moved around base domain vertices end up with barycentric coordinates that are invalid for all the base domain faces around that vertex.

Tarini et al. [2004] and Kraevoy and Sheffer [2004, 2005] fix the boundary of a group of base mesh faces, update the barycentric coordinates in the interior, and then possibly re-assign some vertices to different faces inside the group. The methods differ in the grouping they use and the choice of parameterization technique used for the barycentric coordinates computation.

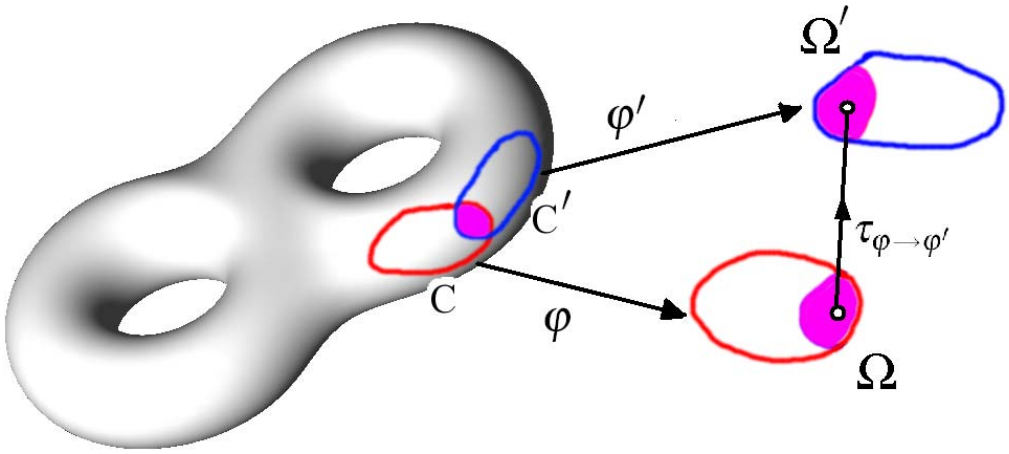


Figure 8.4: A global parameterization (also called differential manifold in the mathematics literature) is a set of parameterized charts $(\varphi, \varphi', \ldots)$ connected by differentiable transition functions $(\tau_{\varphi\to\varphi'} \ldots)$.

8.3 Global Parameterization

This family of methods use a set of triangular charts to define the base complex. For some applications, such as texture mapping, or surface approximation with tensor-product splines, it is preferred to use a base complex composed of quadrilaterals. The difference seems subtle at first sight, but automatically constructing a good quadrilateral base complex is still an open problem. A variant of the MIPS [Hormann and Greiner, 2000a] method, applied to a quadrilateral base complex, was proposed [Tarini et al., 2004]. The method lets the user interactively define the base complex. More recently, advances to automate the process were made, as shown further in this section.

The next subsection introduces the fundamental notions. Then we will review recent advances in this area.

8.3.1 Differential manifold

We start by defining the notion of *differential manifold* (that formalizes the notion of global parameterization outlined at the beginning of the section). This makes it possible to define a globally smooth parameterization on a surface of arbitrary genus, by connecting multiple local parameterizations. To our knowledge, the notion of differential manifold was first introduced to the geometry processing community by Grimm and Hugues [Grimm and Hughes, 1995]. More recently, a construct was proposed to define surfaces of class C^∞ [Ying and Zorin, 2004].

Given a surface $\mathbf{S}$, we consider a set of (possibly overlapping) topological disks $\{\mathbf{C}\}$, called *charts*, and a set of functions $\{\varphi\}$ that put each chart $\mathbf{C}$ in correspondence with

a 2D domain Ω (c.f. Figure 8.4). The 2D coordinates will be denoted by θ, ϕ. The set of functions $\{\varphi\}$ defines a differential manifold if the following condition is satisfied : *Given two charts* $\mathbf{C}$ *and* $\mathbf{C}'$*, if their intersection* $\mathbf{C} \cap \mathbf{C}'$ *is homeomorphic to a disk, then the two images of their intersection* $\mathbf{C} \cap \mathbf{C}'$ *in parametric space through* φ *and* φ' *are linked by a differentiable geometric transform* $\tau_{\varphi \to \varphi'}$*:*

$$\forall \mathbf{p} \in \mathbf{C} \cap \mathbf{C}', \quad \varphi'(\mathbf{p}) = \tau_{\varphi \to \varphi'}(\varphi(\mathbf{p}))$$

The functions $\tau_{\varphi \to \varphi'}$ are called *transition functions* [Khodakovsky et al., 2003]). A manifold is said to be *affine* if all the transition functions are translations. Complex manifolds admit any one-to-one conformal function as a transition function (c.f. Section 4.3 for the notion of conformality). In particular, this includes similarities, i.e. functions composed of translations, rotations and scalings.

8.3.2 Exterior calculus

By using a representation of a manifold as a set of charts (with the associated transition functions), it is possible to manipulate functions defined over the manifold, compute their gradients and integrate them over subsets of the manifold. However, these computations often involve the Jacobian matrix of each local parameterization in a non-trivial manner, and make all the computations quite involved. Despite the possibility of using formal tools (e.g. Maple), this way of conducting computations remains frustrating from an intellectual point of view.

Exterior calculus is an elegant alternative to this problem. This is a geometric calculus, without coordinates, that considers the functions as abstract mathematical objects, combined with operators. At the end, to use the functions, one still needs to "instantiate" them in a local coordinate frame. However, by post-poning the (lower-level) representation in coordinates at the end of the mathematical reasoning, most computations are simplified by higher level considerations, guided by the algebraic structure defined by the functions and the operators.

Exterior calculus generalizes the fundamental theorem of calculus, that defines the integral of a function :

$$\int_a^b f(x)dx = F(b) - F(a)$$

where F denotes the primitive of f.

The generalized theorem writes:

$$\int_\Omega d\omega = \int_{\partial\Omega} \omega$$

where ω is a differential k-form, $\partial\Omega$ denotes the border of Ω, and d denotes the exterior derivative. For the moment, one may think about a k-form as something that wants to be integrated over a k-dimensional domain. Therefore, 0-forms correspond to classic functions, 1-form to vector fields (that one can integrate along a curve), and 2-forms to functions that one wants to integrate over a surfacic domain. About the exterior

derivative, this is a generalization of the gradient operator. The exterior derivative has also the noticeable property of vanishing when applied twice ($dd\omega = 0$), we will explain why further.

The divergence theorem (Ostrogradsky-Gauss) is a particular case :

$$\int_{\mathrm{Vol}} \mathrm{div} K dv = \int_{\partial \mathrm{vol}} K \cdot N dS$$

where N denotes the normal to the border of the volume. Note that in the "exterior calculus" version, the scalar product with the normal is already taken into account by the integration over $\partial\Omega$ (this is the way exterior calculus considers the border of a subset). More generally, exterior calculus exhibits a symmetry between the integrated entities (called *differential forms*) and the integration domains (called *chains*). The integration of a form ω over a chain Ω is then a certain type of inner product (or dot product) $< \omega, \Omega >$. The border operator ∂ computes the border $\partial\Omega$ of a chain Ω, and returns another chain, of lower dimension. With this notation, Stokes theorem writes :

$$< d\omega, \Omega >=< \omega, \partial\Omega >$$

in other words, from an intuitive point of view, the exterior derivative d becomes the border operator ∂ when one "shifts" if from the integrated form ω to the integration domain Ω. This exhibits the *duality* between the exterior derivative d and the border operator ∂. For this reason, differential forms are also called *co-chains* and the exterior derivative is also called the *co-border* operator[1].

Finally, one can note that the fundamental theorem of calculus is retrieved as a special case :

$$\int_{(a,b)} f(x)dx = \int_{a,b} dF = \int_{a^- \cup b^+} F = F(b) - F(a)$$

This is the orientation of the border $\partial(a, b) = a^- \cup b^+$ that introduces the "-" sign in front of $F(a)$.

To our knowledge, notions of exterior calculus first appeared in the community in Pinkall and Polthier's paper [Pinkall and Polthier, 1993], where they used Hodge duality to compute minimal surfaces. Then, Gu *et. al* [Gu and Yau, 2003; Jin et al., 2004] used the fundamental notions involved in Poincaré's conjecture to compute global parameterizations. Anil Hirani developed in his Ph.D thesis a discrete counterpart of exterior calculus (DEC) [Hirani, 2003], especially well suited to geometry processing with meshes. More recently, in 2005, Schröeder and Desbrun gave a course at SIGGRAPH about DEC. This latter course contributed to make these complex abstract notions accessible to a wider community, and we redirect the interested reader to their course notes for more details on this fascinating topic. The next subsection quickly reviews methods based on co-homology (and introduces the related notions). In what follows, we keep the general continuous setting for the explanations.

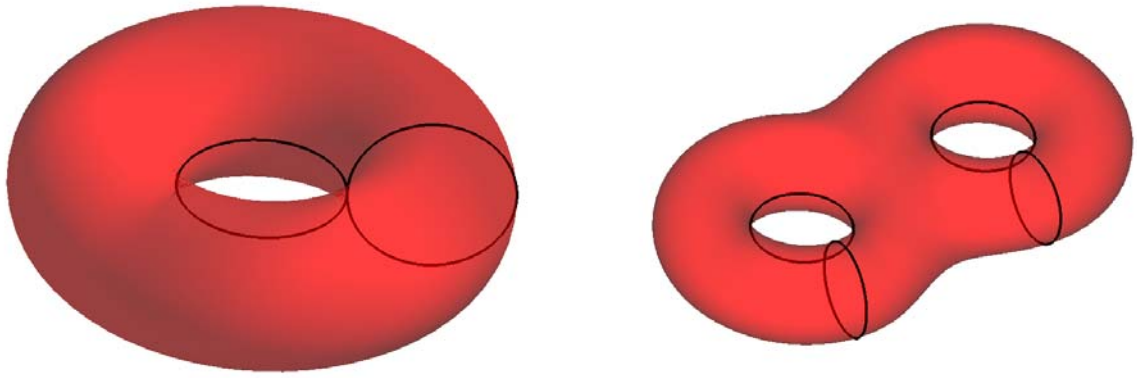


Figure 8.5: Homology bases of a torus and a double torus.

8.3.3 Homology and co-homology

As we have seen in the previous subsection, the notion of border of an integration domain plays a central role in exterior calculus, since the fundamental theorem (Stokes) connects integrals over the domain with integrals on the border. This exhibits fundamental connections between the theory of integration and the topology of the surface, characterized by its homology group. For instance, two closed curves C_1 and C_2 are homologically equivalent if one can be transformed into the other one by continuous deformation. More generally, two closed curves are homology-equivalent if their union corresponds to the border of a subset of the object[2]. This notion makes it possible to decompose any curve in the surface into a "sum" of fundamental curves, called an homology basis. Figure 8.5 shows an example of homology basis for a torus and double torus. More generally, an homology basis of an object of genus g (in other word, a coffee cup with g handles) has $2g$ elements. As can be seen in the Figure, each handle adds 2 fundamental curves (one of them winds around the handle, and the other one winds in the perpendicular direction).

We have mentioned in the previous section the duality between integration domains and differential forms. This duality also applies to the notion of homology (that relates to curves, or chains), from which co-homology (that relates to co-chains, or differential forms) can be derived. We first need to give two more definitions:

- A form ω is *closed* if $d\omega = 0$
- A form ω is *exact* if it is equal to the exterior derivative of another form ($\exists\sigma/\omega = d\sigma$);

Note that exact forms are closed (since $dd\omega = 0\forall\omega$). Based on the duality between forms (things to be integrated) and chains (domains of integration), we can now better

[1]In exterior calculus, the dual of a thing is called the co-thing.

[2]We see again that the notion of border involved in the fundamental theorem plays a central role.

understand the terminology and notations : the boundary operator ∂ is dual to the exterior derivative d, a closed form ω (i.e. such that $d\omega = 0$) is dual to a closed chain Ω (i.e. such that $\partial\Omega = 0$). Moreover, the boundary of a chain is always closed ($\partial\partial\Omega = \oslash$), this explains why the exterior derivative vanishes when applied twice ($dd\omega = 0$).

We can now elaborate on co-homology : on a surface, one can associate to each fundamental curve of the homology basis a set of closed 1-forms (e.g. vector fields) that are equivalent with respect to co-homology. More precisely, two 1-forms ω_1 and ω_2 are equivalent if their integral along all the curves of the homology basis match. Note that this is the case if their difference $\omega_2 - \omega_1$ is exact since the integration of the difference along an element Ω of the homology basis gives :

$$\exists\sigma/\omega_2 - \omega_1 = d\sigma \Rightarrow < \Omega, d\sigma >=< \partial\Omega, \sigma >=< \oslash, \sigma >= 0$$

(we remind that the elements of the homology basis are closed curves). We can now give the general definition of co-homology, that is to say the quotient space of closed forms on exact forms (i.e. two closed forms are equivalent if their difference is an exact form). This relation is dual to homology: two closed curves are equivalent if they are the border of a subset[3].

As shown in Figure 8.6, Gu and Yau used these notions to compute a global conformal parameterization on a surface of arbitrary genus [Gu and Yau, 2003; Jin et al., 2004]. To do so, they compute a holomorphic function (i.e. the generalization of conformal functions mentioned in Section 4.3), based on an important theorem that states that each co-homology class contains a unique harmonic one-form. We have already seen that the coordinates of a conformal map are two harmonic functions. Similarly, a holomorphic function is composed of two conjugate (i.e. orthogonal) harmonic one-forms. Then, their method operates as follows : they first compute a homology basis of the surface (using for instance Erickson's method [Erickson and Whittlesey, 2005] or Lazarus's one [Lazarus et al., 2001]) (A), then they deduce a co-homology basis, and find a pair of conjugate harmonic one-forms (B). Finally, they integrate the one-forms to find the parameterization (C). Note that for an object of genus g, the parameterization has necessarily $2g$ singularities. Ray et.al [Ray et al., 2006] have proposed a way of automating the placement of these singularities and generalize them to fractional indices (see Figure 8.7).

In practice, the discrete version of Gu *et. al*'s framework operates by solving linear systems with three types of constraints: harmonicity, closedness, and duality. The harmonicity equations are similar to the ones used by planar methods: the sum of vectors for edges incident to a vertex, weighed by the cotangent harmonic weights, is zero. The closedness equations state that the sum of the 3 vectors for the edges of each face is zero (a "gradient field" has no divergence). The duality conditions replace the boundary conditions in traditional parameterizations; they impose fixed values for the integral of the field on the closed loops forming the homology basis of the surface (the $2g$ curves that cut open the handles of the mesh, where g is the genus of the surface). Several

[3]Being the border of a subset is dual to exactness for forms.

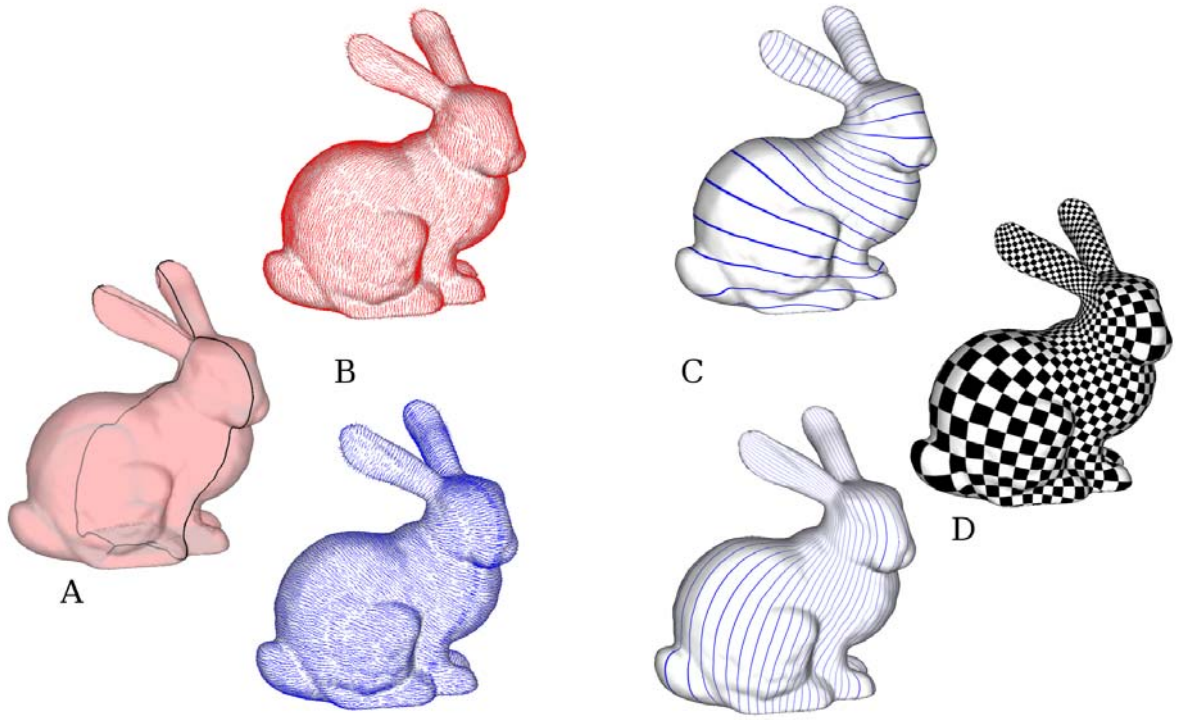


Figure 8.6: The method developed by Gu and Yau to construct a differential manifold first computes a homology basis (A), then deduces a co-homology basis and finds the (unique) harmonic one form in each co-homology class (B). Finally, the u and v potentials are obtained by integrating these harmonic one-forms (C), that together define a holomorphic function (D) (data courtesy of Stanford, parameterization courtesy of X. Gu)

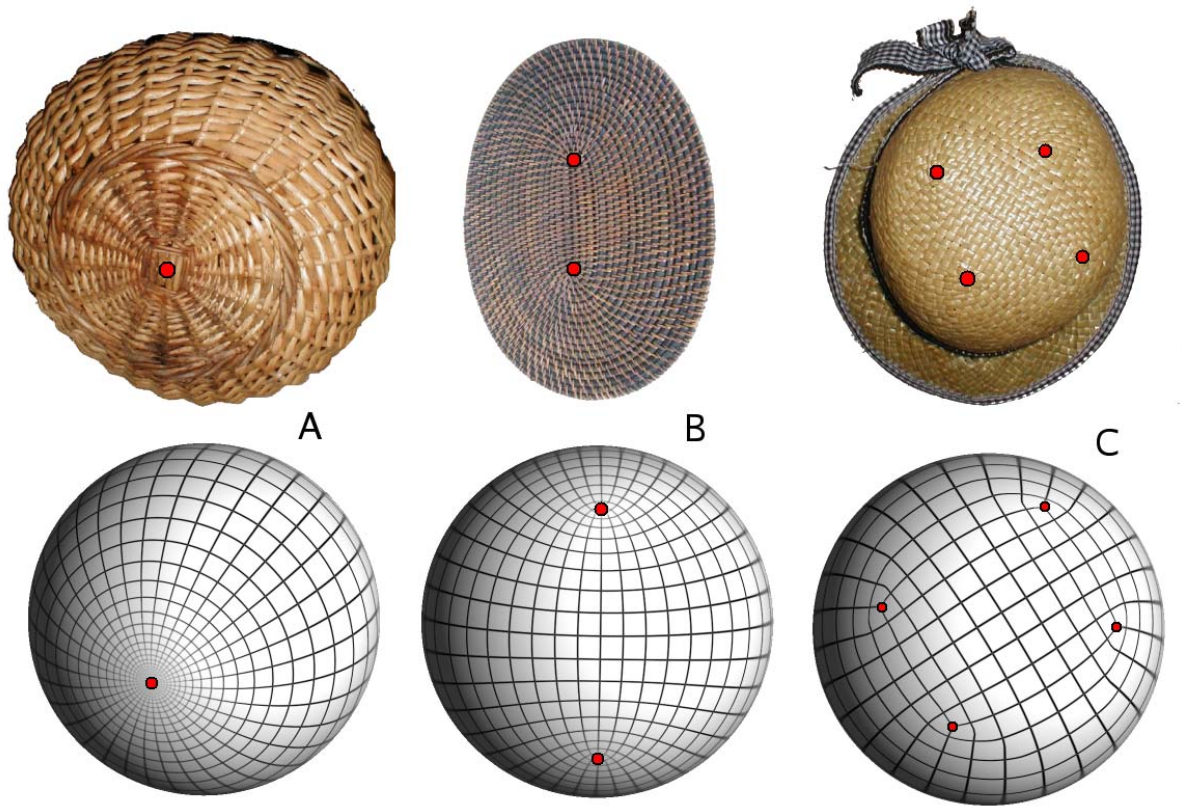


Figure 8.7: It is well known from mathematicians and rattan objects makers that the mesh of an object of genus g has $2g$ singularities (or poles). Thus, a sphere ($g = 1$) has two poles (A). It is possible to subdivide a pole into two half-poles (B) or four quarter-poles (C). The order of multiplicity of a pole, or the rotation angle that the mesh undergoes when turning around the pole is called the *index* of the pole.

systems are solved, in each system the integral across one of the curves is constrained to 1, and all the others to 0.

Other authors have proposed methods that directly compute the parameterization, based on modified Floater conditions. For instance, Steiner and Fischer have proposed to make the object equivalent to a disk using a cut graph, and insert translation vectors in Tutte's conditions related to the vertices located on the cut graph [Steiner and Fischer, 2005]. Tong *et.al* developed independently the same idea [Tong et al., 2006], using the formalism of exterior calculus mentioned above, and adding singularities of fractional index.

To summarize, these methods [Gu and Yau, 2002; Kharevych et al., 2006; Tong et al., 2006] implicitly create a parameterization by solving for differential one-forms. Instead of associating (u, v) coordinates with vertices as in typical planar parameterization, they associate planar vectors (du, dv) with the edges of the mesh, computing a gradient field, or one-form. These parameterizations can be converted to a special form of planar parameterization by fixing a vertex, then walking around to other vertices and adding

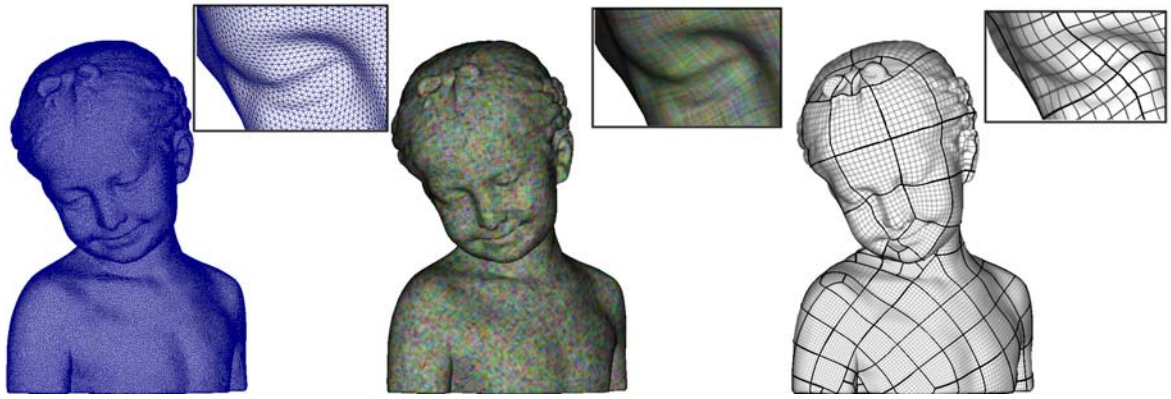


Figure 8.8: From a triangulated mesh (left), the Periodic Global Parameterization method starts by smoothing a vector field (center) and then computes a parameterization such that the gradient vectors are aligned with the vector field (right).

edge vectors. The method guarantees that inside a topological disc region, the same coordinates are obtained for each vertex regardless of the path used to walk to it from the source vertex. Genus-1 can be mapped to an infinite plane using a multi-periodic function by tiling the plane using a modular parallelogram - a fundamental domain bounded by 4 curves, parallel two by two. For higher genus, applying this method directly will cover the plane multiple times. Such parameterizations are guaranteed to be continuous everywhere, except at a small number of singular points, and are therefore sometimes referred to as globally continuous parameterizations.

8.3.4 Periodic Global Parameterization

The Periodic Global Parameterization method [Ray et al., 2006], shown in Figure 8.8, aims at letting the singularities naturally emerge from the optimization of the parameterization. As a consequence, it is not possible to determine the homology basis in advance. As in Alliez *et. al*'s anisotropic remeshing method [Alliez et al., 2003], the method first computes a guidance vector field by smoothing the principal directions of curvature. Then the difficulty is to allow the coordinates to wind around the features of the object. To do so, the method uses the natural periodicity of the sine and cosine function. The complete optimization problem is restated in terms of new variables, that correspond to the sine and cosine of the actual coordinates. The main difficulty is that the method generates invalid vertices, edges and triangles around the singularities. Therefore it requires a post-processing step. At this point, we can either use methods based on co-homology (but they require manual intervention), or PGP (but it requires an inelegant post-processing to fix the singularities). However, another category of methods, based on eigenvectors computations, seem a promising research avenue to define both automatic and simple methods.

8.3.5 Spectral methods

Spectral methods study the eigenfunctions of operators (or eigenvectors of matrices in the discrete setting). Several reviews on this topic are available [Gotsman, 2003; Lévy, 2006], and we also give some web references on the following webpage `http://alice.loria.fr/publications`.

We focus on the Laplace operator, that plays a fundamental role in conformal mesh parameterization, as shown in Section 4.3.

The eigenfunctions f of the Laplace operator Δ are characterized by :

$$\Delta f = \lambda f$$

Before elaborating on the eigenfunctions, we give more detailed about the Laplacian and its generalizations. The Laplacian plays a fundamental role in physics and mathematics. In $\mathbb{R}^n$, it is defined as the divergence of the gradient:

$$\Delta = \text{div grad} = \nabla.\nabla = \sum_i \frac{\partial^2}{\partial x_i^2}$$

Intuitively, the Laplacian generalizes the second order derivative to higher dimensions, and is a characteristic of the irregularity of a function as $\Delta f(P)$ measures the difference between $f(P)$ and its average in a small neighborhood of P. Generalizing the Laplacian to curved surfaces require complex calculations, that can be greatly simplified by a mathematical tool called exterior calculus (EC). See section 8.3.2 or Peter Schroeder's Siggraph 2005 course for more details about exterior calculus. EC generalizes the gradient by d and divergence by $\delta = *d*$ (where $*$ denotes the Hodge star), which are built independently of any coordinate frame. Using EC, the definition of the Laplacian can be generalized to functions defined over a manifold $\mathcal{S}$ with metric g, and is then called the Laplace-Beltrami operator:

$$\Delta = \text{div grad} = \delta d = \sum_i \frac{1}{\sqrt{|g|}} \frac{\partial}{\partial x_i} \sqrt{|g|} \frac{\partial}{\partial x_i}$$

where $|g|$ denotes the determinant of g. The additional term $\sqrt{|g|}$ can be interpreted as a local "scale" factor since the local area element dA on $\mathcal{S}$ is given by $dA = \sqrt{|g|} dx_1 \wedge dx_2$.

The eigenfunctions and eigenvalues of the Laplacian on a (manifold) surface S, are all the pairs (H^k, λ_k) that satisfy $-\Delta H^k = \lambda_k H^k$. The "$-$" sign is here required for the eigenvalues to be positive. On a closed curve, the eigenfunctions of the Laplace operator define the function basis (sines and cosines) of Fourier analysis[4]. On a square, they correspond to the function basis of the DCT (Discrete Cosine Transform), used for instance by the JPEG image format. Finally, the eigenfunctions of the Laplace-Beltrami operator on a sphere define the Spherical Harmonics basis. Figure 8.9 shows how they look like for a more general object. Since it generalizes spherical harmonics

[4]This is easy to check by noticing that in 1D, the Laplace operator corresponds to the standard second order derivative. The eigenfunctions are simply $\sin \omega t$ (resp. cos) associated with the eigenvalues $-\omega^2$.

Figure 8.9: Some of the eigenfunctions of the Laplace operators.

to arbitrary manifolds, this function basis is naturally called the Manifold Harmonics Basis (MHB) [Vallet and Lévy, 2007]. This tech-report gives their formal definition using the finite element formalism, and explains how to efficiently compute the MHB. In the discrete setting, one can approximate the eigenfunctions by computing the eigenvectors of a discrete Laplacian. Many different applications can use these eigenfunctions (some of them replace the discrete Laplacian with a combinatorial Laplacian, that only takes the mesh connections into account) :

- matrix re-ordering [Fiedler, 1973] and mesh re-ordering [Isenburg and Lindstrom, 2005]
- mesh compression [Karni and Gotsman, 2000]
- shape classification [Reuter et al., 2006]
- graph embedding [Koren, 2003]

The graph embedding problem (i.e. how to nicely draw a graph by placing its vertices on the screen) shows strong connections with the parameterization problem. Note that Tutte's paper where his celebrated theorem is proved is entitled *how to draw a graph* [Tutte, 1963]. This general problem is also well known by the *automatic learning* research community as a *Manifold learning* problem, also called *dimension reduction*, see for instance Martin Law's web page `http://www.cse.msu.edu/~lawhiu/manifold/`. In other words, given a graph embedded in $\mathbb{R}^n$, one tries to estimate its intrinsic dimension (is this a curve, a surface, a volume or an object of even higher-dimension ?). One of these methods, called ISOMAP [Tenenbaum et al., 2000] computes the geodesic distances between each pair of vertex in the graph, and then uses MDS (*multidimensional scaling*) [Young, 1985] to compute an embedding that best approximates these distances. Multidimensional scaling simply minimizes an objective function that measures the deviation between the geodesic distances in the initial space and the Euclidean distances in the embedding space (GDD for Geodesic Distance Deviation), by computing the eigenvectors of the matrix $D = (d_{i,j})$ where $d_{i,j}$ denotes the geodesic distance between vertex i and vertex j.

Isomaps and Multidimensional scaling were used to define parameterization algorithms in [Zigelman et al., 2002], and more recently in the *ISO-charts* method [Zhou

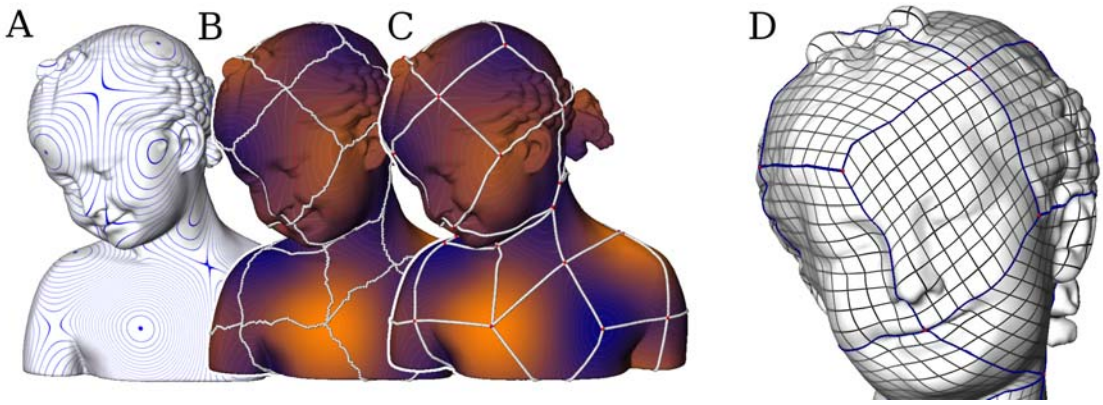


Figure 8.10: Spectral Surface Quadrangulation first computes a Laplace eigenfunction (A), then extracts its Morse complex (B), smooths it (C) and uses it to partition the mesh into quads, that can be parameterized (D).

et al., 2004], used in Microsoft's DirectX combined with the packing algorithm presented in [Lévy et al., 2002]. Interestingly, the ISO-charts method uses MDS for both segmenting the model and parameterizing the charts. This provides a nice and coherent theoretical framework, that can be relatively easily translated into efficient implementations.

However, the spectral parameterization methods listed above still need to partition the mesh into charts. More recently, Dong *et. al* used the Laplacian to decompose a mesh into quadrilaterals [Dong et al., 2005, 2006], in a way that facilitates constructing a globally smooth parameterization. As shown in Figure 8.10, their method first computes one eigenfunction of the Laplacian (the 38^{th} in this example), then extract the Morse complex of this function, filters and smooths the Morse complex and uses it to partition the mesh into quads. These quads are parameterized, and inter-chart smoothness can be further optimized using global relaxation [Khodakovsky et al., 2003; Tarini et al., 2004].

Chapter 9

Cross-Parameterization / Inter-surface Mapping

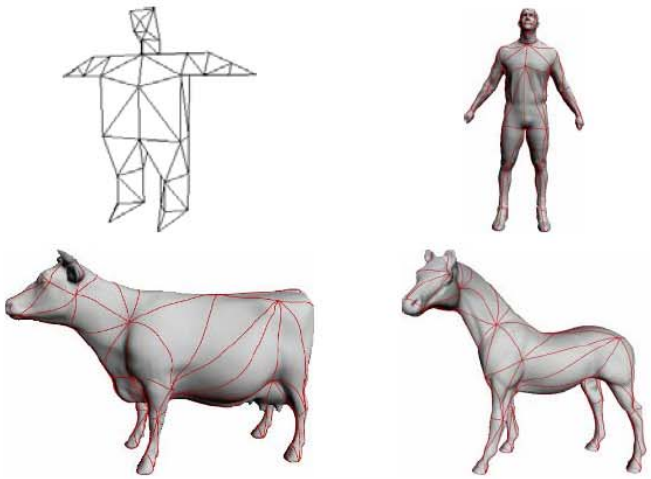
In a cross-parameterization or inter-surface-mapping setup the parameter domain for one mesh is another surface mesh. Cross-parameterization is used to morph or blend between meshes and to transfer properties between them. For morphing in addition to obtaining a mapping it is necessary to obtain a common compatible connectivity for the two meshes. The most common approach for pair-wise mapping is to parameterize both models on a common base domain. Free-boundary planar parameterization is clearly unsuitable for this purpose. Instead alternate domains such as a simplicial complex or a sphere are commonly used.

9.1 Base Complex Methods

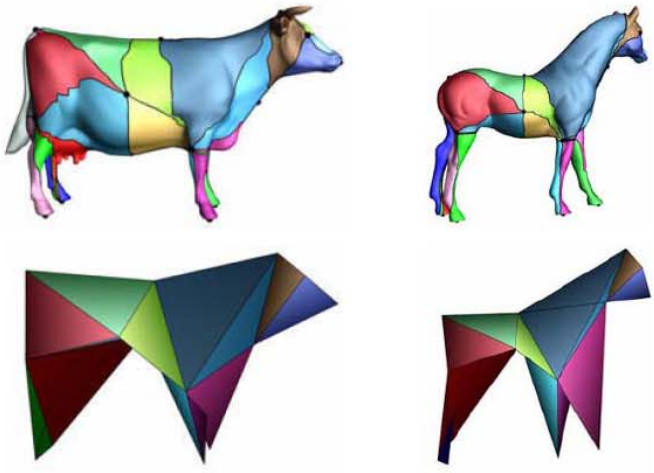
Many methods use mapping to a common base-complex to obtain a cross-parameterization. Since the base must be shared the construction is significantly more challenging than when parameterizing a single mesh. Praun et al. [2001] partition a mesh into triangular patches, which correspond to the faces of a user given simplicial complex, by drawing a network of paths between user-supplied feature vertices that correspond to the vertices of the base mesh.

Schreiner et al. [2004] and Kraevoy and Sheffer [2004] extend the methods of Praun et al. [2001] and Kraevoy et al. [2003] to construct the simplicial complex automatically, in parallel to the patch formation. The input to both methods includes a set of correspondences between feature vertices on the two input models. The methods use those as the vertices of the base complex. They simultaneously trace paths on the input meshes between corresponding pairs of vertices, splitting existing mesh edges if necessary.

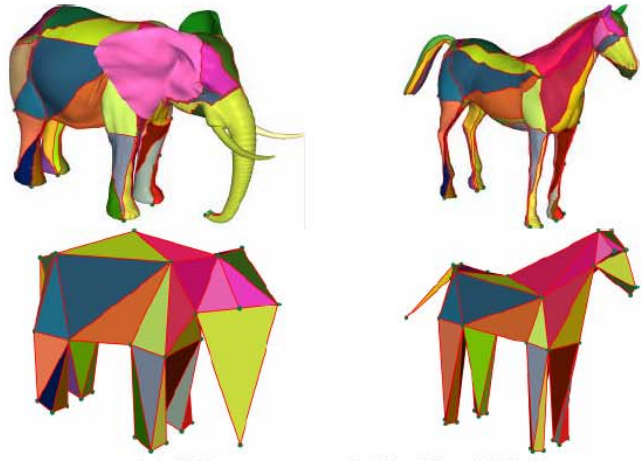
Once the base is created, the meshes can be mapped to the base using the techniques reviewed in the chapter on alternative domains. Schreiner et al. [2004] use an alternative approach. They never compute an explicit map between the full-resolution objects and the base domain. Instead, they alternate the role of base domain between the two meshes, at various complexity levels in a multi-resolution representation. They progressively refine each mesh by adding new vertices and relaxing their location using a stretch-based metric measured on a temporary planar unfolding of their neighborhoods.



(a) [Praun et al. 2001]



(b) [Schreiner et al. 2004];



(c) [Kraevoy and Sheffer 2004]

Figure 9.1: Base complex construction.

9.2 Energy driven methods

Several methods use an energy-driven approach for cross-parameterization, where one mesh is directly attracted towards the other [Allen et al., 2003; Sumner and Popović, 2004]. The attraction energy function consists of components that pull the vertices of one mesh towards nearest locations on the other while trying to preserve the shape of the mesh as much as possible. To facilitate correspondence they require the user to specify several dozen point-to-point correspondences between the input models. The methods work well when the meshes are very similar, e.g. humans in the same pose [Allen et al., 2003] but tend to converge to a poor local minimum with increase in shape difference. These methods are quite sensitive to the weights used inside the energy functional to account for the different components. One advantage of these approaches is that they can find mappings between models of different topology (genus, etc.), though these maps are no longer bijective.

9.3 Compatible Remeshing

For applications such as morphing it is not enough to obtain a cross-parameterization between the two models. For these applications, at the end of the process the two models are typically required to have the same connectivity. There are three main approaches for generating such common connectivity.

- *Base mesh subdivision:* Several methods including [Praun et al., 2001] use the base mesh connectivity and refine it using the one-to-four subdivision pattern, introducing as many levels of subdivision as necessary to capture the geometry of both models. The advantage of the method is simplicity. It's drawback is the dependence on the shape of base mesh triangles. The method also tends to require large triangle count to achieve acceptable accuracy (roughly factor 10 compared to input mesh sizes).

- *Overlay:* Another approach for generating common connectivity [Alexa, 2000; Schreiner et al., 2004] is to intersect the two input meshes in the parameter domain, combining all their vertices and generating new vertices at edge-edge intersections. The method preserves exactly the input geometries but is not-trivial to implement and like subdivision increases the triangle count by roughly a factor of 10.

- *Remeshing:* In [Kraevoy and Sheffer, 2004] the authors propose an alternative where they use the connectivity of one of the input meshes as a basis and then refine it as necessary based on an approximation error with respect to the second mesh. The resulting meshes have significantly lower triangle count than using the other two approaches. The result heavily depends on which of the inputs is selected as the source for common connectivity. Unlike in the overlay approach, some fine features of the second mesh are only approximated and it is difficult to faithfully reproduce sharp features.

Chapter 10

Numerical Optimization

In all the parameterization methods listed in the section above (Floater's parameterization method, harmonic parameterization, conformal parameterization), we need either to solve a linear system or to minimize a quadratic objective function. A specificity of the numerical problems yielded in mesh parameterization is that the involved matrices are usually sparse. As a consequence, we will first explain how to efficiently implement a data structure for sparse matrices. To make user's life easier, we propose an implementation that can dynamically grow when coefficients are added to the matrix. A C++ version of this implementation is given in the companion material, based on the `std::vector` data structure. This data structure is available in our Graphite software. We also provide a C version in OpenNL `http://alice.loria.fr/software`. Note that this data structure can be easily extended. Our C++ and C implementations also provide the following features (not detailed in the implementation given in the appendix to keep its length reasonable):

1. storage of the diagonal term
2. storage of both sparse rows and columns
3. symmetric storage (i.e., do not store the upper triangle for symmetric matrices)
4. non-square matrices
5. matrix $\times$ matrix multiply

Features (1) and (2) are interesting for implementing Jacobi preconditioner (requires 1) or SSOR preconditioner (requires 1+2). Feature (3) speeds up the conjugate gradient algorithm (but requires some modification in the matrix $\times$ vector routine). Features (4) and (5) can be used by more sophisticated non-linear solvers (such as MIPS and ABF).

Numerical methods are a particularly efficient formalism for geometric modeling problems that involve large meshes. The approach consists in formalizing the problem as a function of many variables and optimizing it. The variables correspond to values (e.g., coordinates, colors, texture coordinates, etc.) attached to the vertices of the mesh.

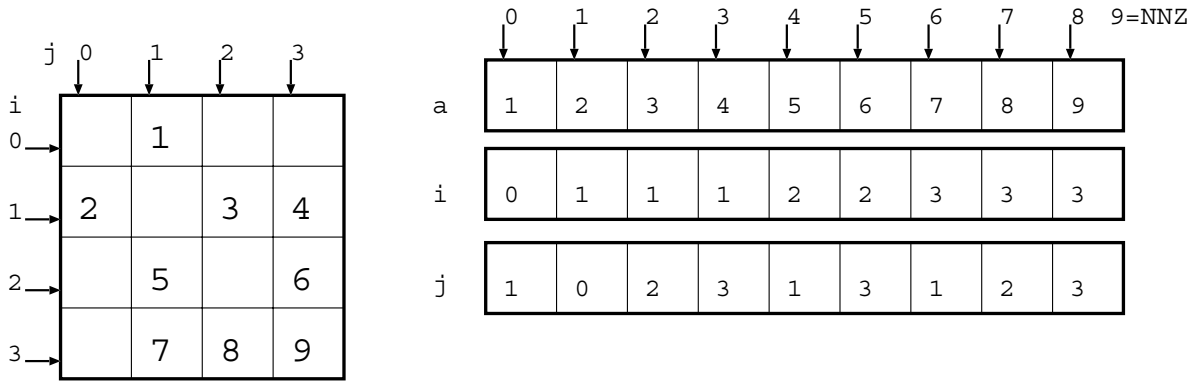


Figure 10.1: Example of sparse matrix with TRIAD format

These numerical optimization problems involve large matrices, defined from the mesh and the values attached to the vertices and/or the edges and/or the facets of the mesh. In most cases, these matrices are sparse (and have a structure identical or strongly related with the adjacency matrix of the graph formed by the mesh). We start this section by reviewing some efficient data structures to represent sparse matrices.

10.1 Data structures for sparse matrices

Algorithm 1 TRIAD data structure for sparse matrices

```
struct TRIADMatrix {
        int N ; // dimension of the matrix
        int NNZ ; // number of non-zero coefficients
        double* a ; // non-zero coefficients (array of size NNZ)
        int* I ; // row indices (array of size NNZ)
        int* J ; //  column indices (array of size NNZ)
} ;
```

Algorithm 2 Matrix $\times$ vector product with the TRIAD data structure

```
void mult(double* y, TRIADMatrix* M, double* x) {
   for(int i=0; i<M->N; i++) {
      y[i] = 0.0 ;
   }
   for(int k=0; k<M->NNZ; k++) {
      y[M->I[k]] += M->a[k] * x[M->J[k]] ;
   }
}
```

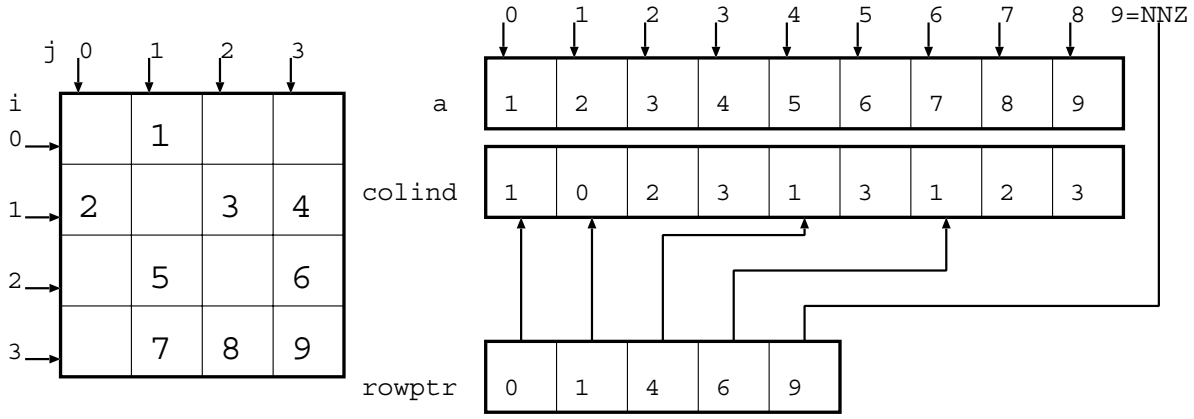


Figure 10.2: Example of sparse matrix represented in the CRS format.

10.1.1 TRIAD data structure

To optimize the storage of sparse matrices, the first idea that comes naturally to mind consists in storing only the non-zero coefficients, with the associated (i, j) indices. Therefore, instead of storing n^2 coefficients, we only need to store NNZ coefficients (NNZ: Number of Non-Zero coefficients) and the same number of (i, j) indices. This data structure is referred to as the *TRIAD format*, depicted in Figure 10.1. We also show an implementation in C (Algorithm 1). The function that computes the product between a sparse matrix in TRIAD format and a vector is given by Algorithm 2). For instance, this function is particularly useful to implement a solver based on the Conjugate Gradient method (see below).

Despite the obvious gain realized by the TRIAD format as compared to a regular 2D array, this representation still shows a high redundancy of the coefficients i, that store the row index associated to each entry. For this reason, this format is seldom used in numerical libraries (it is limited to input file formats, since its simplicity favors its understanding and use). In practice, the CRS (Compressed Row Storage) data structure is preferred, since it is more compact in memory. We detail this format in the next subsection.

Algorithm 3 Compressed Row Storage data structure for sparse matrices

```
struct CRSMatrix {
        int N ; // dimension of the matrix
        int NNZ ; // number of non-zero coefficients
        double* a ; // non-zero coefficients (array of size NNZ)
        int* colind ; // column indices (array of size N)
        int* rowptr ; // row pointers (array of size N+1)
        double* diag ; // diagonal elements (array of size N)
} ;
```

10.1.2 Compressed Row Storage

The CRS format (Compressed Row Storage) is one of the most commonly used data structure for sparse matrices, used in large numerical libraries. The transposed variant CCS (Compressed Column Storage) is also used (depending on the algorithms and implementation choices). We present here the CRS format. Algorithm 3 presents the implementation in C, and Figure 10.2 shows an example. The CRS data structure uses three indices to represent non-zero coefficients and their indices. As in the TRIAD format, the array `a` stores all the non-zero entries. The array `colind` gives for each entry the corresponding column index. The rows are encoded in a different, more compact manner, by the `rowptr` array. This array gives for each row its start and end in the arrays `a` and `colind`. To facilitate the coding of algorithm, a common practice consists in completing the array `colind` by an additional entry, that points one entry past the last entry of the matrix. This additional entry of `colind` is called a *sentry* and avoids a test in the matrix × vector product, as shown below.

Algorithm 4 Matrix × vector product with the CRS format

```
void mult(double* y, CRSMatrix* M, double* x) {
   for(int i=0; i<M->N; i++) {
      y[i] = 0.0 ;
      for(int jj=M->rowptr[i]; jj<M->rowptr[i+1]; jj++) {
         y[i] += M->a[jj] * x[M->colind[jj]] ;
      }
   }
}
```

Algorithm 4 shows how to implement the product between a sparse matrix `M` stored in the CRS format and a vector `x`. The sentry `rowptr[N] = NNZ` avoids to have a particular case for the last row of the matrix.

Different versions of the CRS data structure exist. Thus, if the matrix is symmetric, it is possible to save some memory by only storing the lower triangular part of the matrix. Other variants store the diagonal coefficients in a separate array, to facilitate the implementation of the Jacobi preconditioner (see below). Other variants store the columns instead of the rows (CCS: Compressed Column Storage). Finally, the BCRS (Block Compressed Row Storage) partitions the matrix into fixed size blocks and stores them in the CRS format. This both optimizes memory accesses, and favors using extending instruction sets (e.g. SSE on Intel Processor).

The CRS data structure and its variants are both compact and efficient. However, in our particular context of numerical problems related with geometric optimization of 3D meshes, the CRS data structure is somewhat too rigid, as explained below. In a geometric optimization problem, it is natural to assemble the matrix by traversing the graph that corresponds to the mesh. During the traversal, coefficients are accumulated in the matrix, generated by elementary neighborhoods (called *stencils* in finite elements parlance). Since the CRS data structure requires to define the number of non-zero

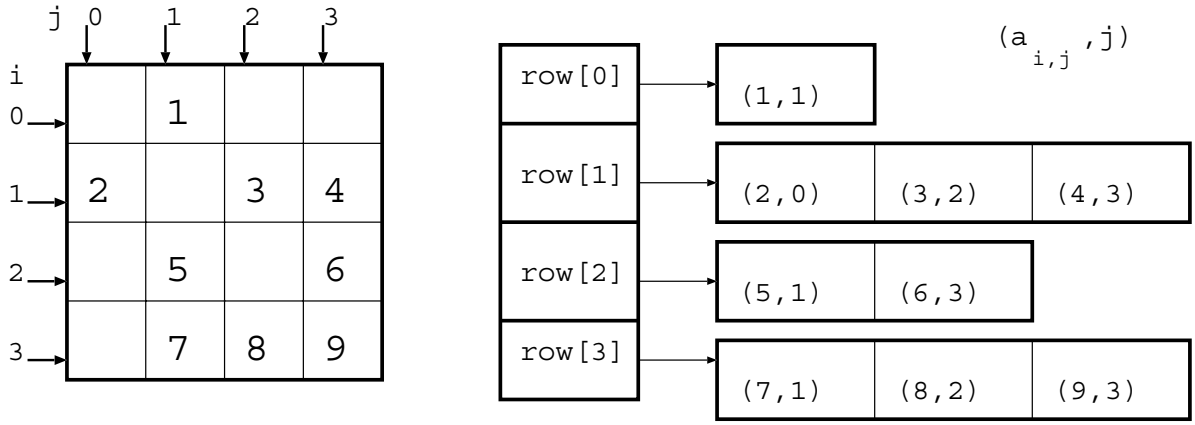


Figure 10.3: Dynamic sparse matrix data structure

coefficients in advance, constructing a CRS matrix requires two traversals of the graph. The first traversal determines the non-zero pattern, i.e. the set of (i, j) index couples such that $a_{i,j}$ is non-zero. Then, the data structure is allocated in memory. Finally, a second traversal computes the numerical values of the $a_{i,j}$ entries. In practice, this way of proceeding is uncomfortable to implement. For this reason, we propose a *dynamic* matrix data structure, for which the number of non-zero coefficients can vary. The implementation is available in our library OpenNL and in the Graphite platform (`http://alice.loria.fr/software`).

10.1.3 Dynamic matrices

Our dynamic sparse matrix is built around the class `std::vector` of the standard C++ library (STL for Standard Template Library). This container class show the interesting capability of growing in memory while minimizing the number of required copies[1]. Thus, our dynamic sparse matrix data structure, depicted in Figure 10.3, and detailed in Algorithm 5, consists of an array `row`. Each row is implemented by a `std::vector` of value-index pairs, and each pair is represented by an instance of the `Coeff` structure. The whole data structure shows a memory storage requirement comparable to the CRS format, while offering better flexibility. This flexibility is exposed to the user by the function `add(i,j,val)`, that adds the value `val` to the coefficient $a_{i,j}$.

The matrix-vector product remains simple to implement with this structure. The source code is given in Algorithm 6. However, it is important to notice that this better flexibility is obtained at the expense of lower performances of the matrix $\times$ vector product (approximately by a factor of 2 in our experimentation). This can be explained by the loss of data locality, that results in less efficient cache usage as compared to the CRS representation. For this reason, in the case where a large number of matrix

[1]This is done by doubling the size of the memory space allocated to the container each time a copy operation is necessary. This makes the number of copies decrease as a function of the log of the final size.

Algorithm 5 Dynamic sparse matrix data structure

```
class SparseMatrix {
public:
    struct Coeff {
        Coeff() { }
        Coeff(int j, double val) : index(j), a(val) { }
        int index ;
        double a ;
    } ;

    class Row : public std::vector<Coeff> {
    public:
        void add(int index, double val) {
            for(int i=0; i<size(); i++) {
                if((*this)[i].index == index) {
                    (*this)[i].a += val ;
                    return ;
                }
            }
            std::vector<Coeff>::push_back(Coeff(index, val)) ;
        }
    } ;

    SparseMatrix(int dim) : dimension(dim) {
        row = new Row[dim] ;
    }

    ~SparseMatrix() { delete[] row;  }

    // aij <- aij + val
    void add(int i, int j, double val) {
        row[i].add(j, val) ;
    }

    // A <- 0
    void clear() {
        for(int i=0; i<dimension; i++) {
            row[i].clear() ;
        }
    }

   // number of non-zero coefficients
   int nnz() const {
      int result = 0 ;
      for(int i=0; i<dimension; i++) {
          result += row[i].size() ;
      }
      return result ;
   }
    int dimension ;
    Row* row ;
} ;
```

Algorithm 6 Matrix × vector product with the dynamic data structure

```
    // y <- Mx
    void mul(double* y, const SparseMatrix& M, const double* x) {
        for(int i=0; i<M.dimension; i++) {
           y[i] = 0 ;
            const Row& R = M.row[i] ;
            for(int jj=0; jj<R.size(); jj++) {
                y[i] += R[jj].a * x[ R[jj].index ]  ;
            }
        }
    }
```

Algorithm 7 Converting a dynamic sparse matrix into the CRS format

```
void convert_to_CRS(
   const SparseMatrix& M,
   CRSMatrix& M_CRS,
   int array_base // 0 for C, 1 for Fortran
) {
   M_CRS.N = M.dimension ;
   M_CRS.NNZ = M.nnz() ;
   M_CRS.A = new double[M_CRS.NNZ] ;
   M_CRS.col_ind = new int[M_CRS.NNZ] ;
   M_CRS.row_ptr = new int[M_CRS.N+1] ;
   int count = 0 ;
   for(int i=0; i<M_CRS.N; i++) {
        const SparseMatrix::Row& R = M.row[i] ;
        M_CRS.row_ptr[i] = count + array_base ;
        for(int jj=0; jj<R.size(); jj++) {
            M_CRS.a[count] = R[jj].a ;
            M_CRS.col_ind[count] = R[jj].index + array_base ;
            count++ ;
        }
    }
    M_CRS.col_ptr[M_CRS.N] = M_CRS.NNZ + array_base ;
}
```

$\times$ vector products need to be computed (e.g., in a Conjugate-Gradient based iterative solver described later), it may be more efficient to convert the dynamic matrix into the CRS format. The conversion routine can be also used to interface the dynamic matrix data structure with sparse direct solvers (SuperLU, TAUCS, MUMPS, etc.) that use the CRS format. The source code of the conversion routine is given in Algorithm 7. This function is not difficult to implement. The only thing that one needs to take care of is the additional parameter `array_base` that deserves some explanations. The C and FORTRAN languages do not use the same convention for array indexing. The first entry of an array is indexed by 0 in C, and by 1 in FORTRAN. For this reason, if the CRS matrix is meant to be used by a FORTRAN routine, this parameter needs to be set to 1.

Now that we have seen the elementary data structures for sparse matrices, the algorithm to manipulate them and their implementation in C and C++, we will see some popular algorithms to solve linear systems.

10.2 Solving Linear Systems

A wide class of geometric modeling problems require to solve a large sparse linear system, i.e. an equation of the form $Ax = b$, with A that denotes a $n \times n$ non-singular square matrix, b a vector of dimension n, and x the unknown vector (of dimension n). In this section, we describe several methods to solve these linear systems.

10.2.1 Relaxation

The relaxation method is the most simple, both from the conceptual and the implementation points of view. This method was widely used in the 90's to implement geometry processing algorithm. It was then later replaced by more efficient algorithms, listed below.

The relaxation method can be easily understood by considering the problem $Ax = b$ as a linear system:

$$\left\{\begin{array}{lllll} a_{1,1}x_1 & +a_{1,2}x_2 & +\ldots & +a_{1,n}x_n & = b_1 \\ & & \vdots & & \\ a_{i,1}x_1 & +a_{i,2}x_2 & +\ldots & +a_{i,n}x_n & = b_i \\ & & \vdots & & \\ a_{n,1}x_1 & +a_{n,2}x_2 & +\ldots & +a_{n,n}x_n & = b_n \end{array}\right.$$

The method consists in traversing the equations one by one, and compute for each equation i the value of x_i obtained by pretending that all the other variables x_j for $j \neq i$ are known. This gives the following update scheme for the variable x_i:

$$x_i \leftarrow \frac{1}{a_{i,i}} \left(b_i - \sum_{j \neq i} a_{i,j} x_j \right)$$

The complete algorithm for the relaxation method then writes:

$$
\begin{array}{l}
\textbf{while } \|Ax - b\| < \epsilon \\
\quad \textbf{for } i \textbf{ from } 1 \textbf{ to } n \\
\qquad x_i \leftarrow \frac{1}{a_{i,i}} \left(b_i - \sum_{j \neq i} a_{i,j} x_j \right) \\
\quad \textbf{end}//\textit{for} \\
\textbf{end}//\textit{while}
\end{array}
$$

with ϵ the precision specified by the user. It is also possible to specify a maximum number of iterations, so that the algorithm stops even if it does not converge.

As can be seen, this algorithm cannot be applied to matrices with zero values on the diagonal. More generally, it is possible to prove a sufficient condition for convergence. If the matrix is diagonal dominant, i.e.

$$\forall i, \forall j \neq i, |a_{i,i}| > |a_{i,j}|$$

then the algorithm converges.

It is possible to speed-up this algorithm, by using the fact that the update applied to each variable x_i "points toward the right direction". Intuitively, going "a bit further" in that direction, by multiplying the displacement by a factor ω, is likely to accelerate the speed of convergence. The so-modified update scheme writes:

$$
\begin{array}{lcl}
x_{prev} & \leftarrow & x_i \\
x_i & \leftarrow & \frac{1}{a_{i,i}} \left(b_i - \sum_{j \neq i} a_{i,j} x_j \right) \\
x_i & \leftarrow & x_{prev} + \omega (x_i - x_{prev})
\end{array}
$$

This update scheme first computes the update of x_i as before, and then multiplies this displacement relative to the previous value x_{prev} by a factor ω. It is possible to prove that the algorithm converges under the same conditions as relaxation (diagonal dominant matrix), and for $\omega \in [1, 2[$. The so-modified algorithm is referred to as SOR (*successive over-relaxation*). By rewriting the update scheme in a more compact form, the complete SOR algorithm is given by:

$$
\begin{array}{l}
\textbf{while } \|Ax - b\| < \epsilon \\
\quad \textbf{for } i \textbf{ from } 1 \textbf{ to } n \\
\qquad x_i \leftarrow (1 - \omega) x_i + \frac{\omega}{a_{i,i}} \left(b_i - \sum_{j \neq i} a_{i,j} x_j \right) \\
\quad \textbf{end}//\textit{for} \\
\textbf{end}//\textit{while}
\end{array}
$$

The optimal choice for the parameter ω is determined by the eigenvalues of the matrix A. Generally, computing those eigenvalues is much more expensive than solving the system. For this reason, either a theoretical study of the numerical problem can give the optimal ω, or it is determined in an empirical way.

The main advantage of the SOR method is its simplicity, from both the conceptual and implementation point of view. This simplicity favored its use in the geometry processing community [Taubin, 1995]. However, as we will see later, the community now uses more efficient methods, such as the conjugate gradient and sparse direct solvers.

10.2.2 The conjugate gradient method

The conjugate gradient method is much more efficient, and does not require much efforts to be implemented. This algorithm is based on the equivalence between solving the linear system $Ax = b$ and minimizing the quadratic form $F(x) = 1/2x^tAx - b^tx$. More precisely, the algorithm computes a basis of vectors that is orthogonal in the space of the matrix (i.e. a basis of conjugate vectors), by applying the Gram-Schmidt orthogonalization algorithm. Fortunate simplifications in the computations make it possible to obtain the vectors of the basis one by one, by only keeping one vector in memory. The next one is then obtained as a linear combination of the previous one and the gradient $\nabla F = Ax - b$ at the current point x. The complete algorithm writes:

Algorithm 8 Pre-conditioned Conjugate Gradient

$i \leftarrow 0$; $\mathbf{r} \leftarrow \mathbf{b} - \mathbf{Ax}$; $\mathbf{d} \leftarrow \mathbf{M}^{-1}\mathbf{r}$;
$\delta_{new} \leftarrow \mathbf{r}^T\mathbf{d}$; $\delta_0 \leftarrow \delta_{new}$;
while $i < i_{max}$ **et** $\delta_{new} > \epsilon^2\delta_0$
    $\mathbf{q} \leftarrow \mathbf{Ad}$; $\alpha \leftarrow \frac{\delta_{new}}{\mathbf{d}^T\mathbf{q}}$;
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{d}$; $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{q}$;
    $\mathbf{s} \leftarrow \mathbf{M}^{-1}\mathbf{r}$; $\delta_{old} \leftarrow \delta_{new}$;
    $\delta_{new} \leftarrow \mathbf{r}^T\mathbf{s}$; $\beta \leftarrow \frac{\delta_{new}}{\delta_{old}}$;
    $\mathbf{d} \leftarrow \mathbf{r} + \beta\mathbf{d}$; $i \leftarrow i + 1$;
end

In this algorithm, the matrix M, called a preconditioner, allows to speed-up the speed of convergence of the algorithm. For instance, the Jacobi preconditioner is simply equal to the diagonal terms of the matrix A. More sophisticated preconditioners exist, however, our experiments have shown that reasonably efficient results are obtained with the Jacobi preconditioner.

As can be seen in this algorithm, the most complicated operation is a matrix $\times$ vector product. All the other computations are simply dot products between vectors, and linear combinations of vectors.

The conjugate gradient method cannot be applied to non-symmetric matrices. For a non-symmetric matrix, it is possible to apply the conjugate gradient method to the normal equation $A^tAx = A^tb$. The resulting method is called *conjugate gradient squared* (CGSQ). However, since this squares the condition number, the loss of numerical stability makes this method not suitable in general.

Another idea consists in deriving from the system $Ax = b$ an equivalent symmetric system, as follows:

$$\begin{pmatrix} Id & A \\ A^t & 0 \end{pmatrix} \begin{pmatrix} 0 \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

It is then possible to applied the conjugate gradient method to this system. This defines the BiCG algorithm *Bi Conjugate Gradient*. A variant named BiCGSTAB (stabilized BiCG) accelerates the speed of convergence.

Name	Location
OpenNL	`http://alice.loria.fr/software`
SuperLU	`http://crd.lbl.gov/~xiaoye/SuperLU/`
TAUCS	`http://www.tau.ac.il/~stoledo/taucs/`
MUMPS	`http://graal.ens-lyon.fr/MUMPS/`
UMFPAK	`http://www.cise.ufl.edu/research/sparse/umfpack/`

Table 10.1: Some freely available sparse direct solvers.

The reader interested in more details about the underlying theory can read the excellent tutorial by Shewchuk [1994] that is available on the web.

10.2.3 Sparse direct solvers

Another method to solve a linear system consists in factorizing the matrix, into a product of matrices that are simple to invert. For instance, the LU factorization consists in finding the lower triangular matrix L and the upper triangular matrix U such that the product LU is equal to the matrix A of the system. If the matrix is symmetric, U corresponds to the transpose of L. Once this factorization is obtained, it is then very easy to solve linear systems that involve $A = LU$, as follows:

$$LUx = b \quad \rightarrow \begin{cases} Lx_1 &= b \\ Ux &= x_1 \end{cases}$$

Thus, this algorithm solves the linear system by solving two triangular systems (by successive substitutions). From a conceptual point of view, the Gauss pivot method taught in class corresponds to this algorithm (formalized in a slightly different way).

In the case of sparse matrices, different methods can compute the factors L and U, also represented by sparse matrices. The corresponding algorithms are quite difficult to implement. Fortunately, different libraries, available on the net, propose freely available implementations (see Table 10.1). Our OpenNL front-end provides an OpenGL-like API to construct a sparse linear system row by row, to remove degrees of freedom as explained in Section 10.4.3 and then solves the linear system either with its built-in iterative solver, or with one of the interfaced sparse direct solver.

As remarked by Botsch et al. [2005], this direct solvers are particularly efficient for geometry processing problems with surface meshes. Among these solvers, TAUCS has the noticeable functionality of operating in an Out-Of-Core manner, by storing the L and U factors on the disk. This makes it possible to solve huge linear systems.

10.2.4 Conclusions on linear solvers

We conclude this subsection with a short summary of these linear system solvers (Table 10.2). In a nutshell, SOR-like methods are both easy to understand and to implement, but do not perform well for more than 10K variables. The conjugate-gradient method realizes a good trade-off between the ease of implementation and efficiency. Direct sparse

method	pros	cons
relaxation - SOR	easy to implement low memory cost	inefficient
conjugate gradient	easy to implement low memory cost	reasonably efficient
direct methods	the most efficient public codes available	high memory cost implementations are complex

Table 10.2: Classes of linear solver, pros and cons.

solvers are the most efficient, but may sometimes consumes considerable amounts of RAM. Finally, direct solvers with Out-Of-Core storage [Meshar et al., 2006] let the user benefit from the high efficiency of sparse direct solvers while keeping the control of the used amount of RAM.

10.3 Functions of many variables

We now focus on functions $F : x \mapsto F(x)$ from $\mathbb{R}^n$ to $\mathbb{R}$. For instance, such a function can formalize the geometric criterion that should be met by a surface. In this context, the arguments of the functions is a vector x that gathers all the coordinates at all the vertices of the mesh. The function returns a real number that measures the "fairness" of the so-described geometry (in the usual definitions, lower values of this quantity correspond to a better quality). The optimization algorithm then seeks for the parameters of the function—i.e. the coordinates at the vertices—that minimize the function. In this context, such a function is also called an "energy", or an "objective function". *Variational* methods minimize an objective function by studying the variations of $F(x)$ in function of the parameters $x = [x_1, \dots x_n]$. These variations are formalized by the derivatives of $F(x)$. More specifically, we will focus on methods that operate at order 2. This requires to define the notion that corresponds to the first order derivative for multivariate functions (called the *gradient*) and the one that corresponds to the second order derivative (called the *Hessian*). We will then study some theorems that facilitate the computations in expressions that combine the derivatives of F.

10.3.1 The gradient

The differential information at order 1 is represented by the vector of all derivatives relative to all variables, called the gradient of F, and denoted by ∇F:

$$\nabla F = \begin{bmatrix} \partial F/\partial x_1 \\ \vdots \\ \partial F/\partial x_i \\ \vdots \\ \partial F/\partial x_n \end{bmatrix}$$

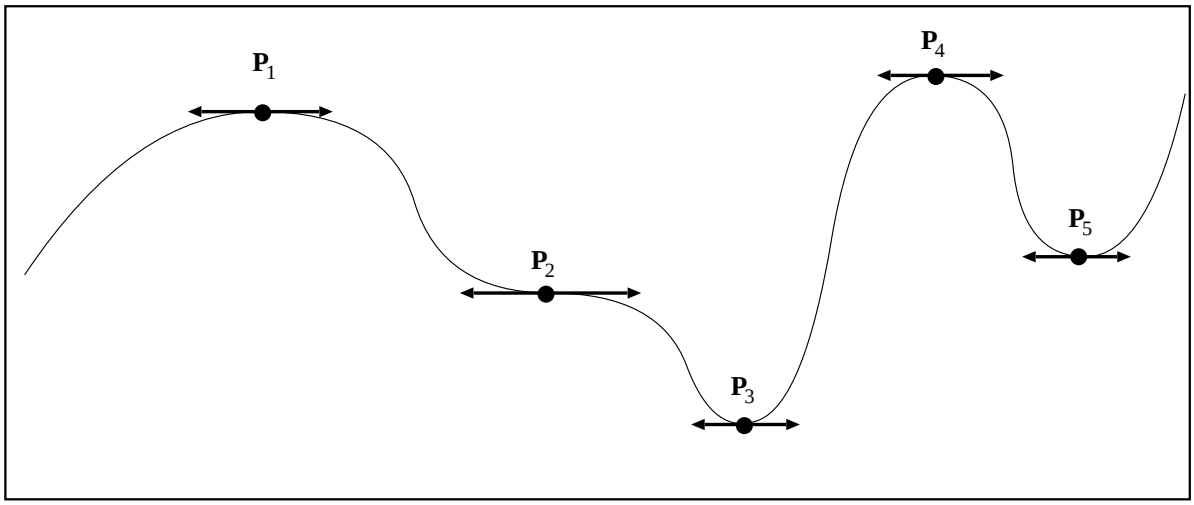


Figure 10.4: Stationary points. The points $\mathbf{P}_1$ and $\mathbf{P}_4$ are local maxima, $\mathbf{P}_2$ is an inflexion point, $\mathbf{P}_5$ is a local minimum and $\mathbf{P}_3$ is the global minimum.

The notion of gradient is very useful in minimization, since a minimum is characterized by the fact that the gradient of F is zero (if F is C^1). However, the zero-set of the gradient (also called the set of *stationary points*) also contains other points, as shown in Figure 10.4 for the univariate case. First, a minimum can be local (rather than global), but the situation can be even worse: a stationary point can also be a (local or global) maximum. It can also be an inflexion point. To further characterize a stationary point, it is also necessary to check the sign of the second order derivative.

We now give little theorems that facilitate calculation with gradients:

$$\begin{array}{lllll} (1) & \|x\|^2 & = & x^t x & \\ (2) & \nabla(b^t x) & = & \nabla(x^t b) & = & b \\ (3) & \nabla(x^t A x) & = & (A + A^t)x & = & 2Ax \text{ if } A \text{ is symmetric} \end{array}$$

(1) is just an easy way of unifying the squared norm of a vector and matrix products (we will use it later). (2) is also trivial. To prove (3), we expand $x^t Ax$ and compute its derivatives relative to the variables x_i:

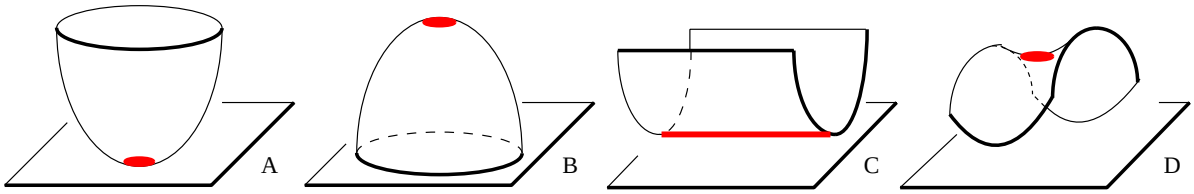


Figure 10.5: Stationary points. A: local minimum (positive definite Hessian); B: local maximum (negative definite Hessian); C: gutter (positive but non-definite Hessian); D: saddle point (Hessian with positive and negative eigenvalues)

$$
\begin{aligned}
x^t Ax &= \sum_{k_1}\sum_{k_2} a_{k_1,k_2} x_{k_1} x_{k_2} \\
\frac{\partial x^t Ax}{\partial x_i} &= \frac{\partial}{\partial x_i}\left(a_{i,i}x_i^2 + x_i \sum_{k_2 \neq i} a_{i,k_2}x_{k_2} + x_i \sum_{k_1 \neq i} a_{k_1,i}x_{k_1} \right) \\
&= 2a_{i,i}x_i + \sum_{k_2 \neq i} a_{i,k_2}x_{k_2} + \sum_{k_1 \neq i} a_{k_1,i}x_{k_1} \\
&= \sum_{k_2} a_{i,k_2}x_{k_2} + \sum_{k_1} a_{k_1,i}x_{k_1} \\
&= \sum_{j} (a_{i,j} + a_{j,i})x_j
\end{aligned}
$$

One can check that this expression corresponds to the i-th row of $(A + A^t)x$.

10.3.2 The Hessian

The differential information to order 2 is represented by the matrix of all second order derivatives relative to all couples of variables (x_i, x_j), called the Hessian of F, and denoted by $\nabla^2 F$:

$$
\nabla^2 F = \begin{bmatrix} \partial^2/\partial_{x_1,x_1} & \dots & \partial^2/\partial_{x_1,x_n} \\ \vdots & & \vdots \\ \partial^2/\partial_{x_n,x_1} & \dots & \partial^2/\partial_{x_n,x_n} \end{bmatrix}
$$

As for univariate functions, it is possible to compute a Taylor series expansion for a multivariate function, around a point x_0. This gives a good (second order) approximation of $F(x_0 + p)$ as a function of the displacement vector p from x_0:

$$
F(x_0 + p) \simeq F(x) + p^t \left(\nabla F(x_0)\right) + 1/2 p^t \left(\nabla^2 F(x_0)\right) p
$$

As previously mentioned, the Hessian gives more information about a stationary point. Figure 10.5 shows some configurations that can be encountered in the case of a function of two variables. According to the sign of the eigenvalues of the Hessian, the configuration is a local minimum (A), a local maximum (B), a gutter (C), or a saddle point (D). Therefore, to minimize a function F, finding the zero-set of the gradient is in general not sufficient. One also needs to check the sign of the eigenvalues of the Hessian. Even if they are positive, this may be a *local* minimum.

10.4 Quadratic Optimization

10.4.1 Quadratic Forms

A quadratic form is a polynomial function such that the degree of its terms is not larger than 2. It is always possible to write a quadratic form as follows:

$$F(x) = 1/2x^tGx + x^tc + \alpha$$

where G is a $n \times n$ symmetric matrix, c is a vector of $\mathbb{R}^n$, and α a scalar. Finding the stationary point(s) of a quadratic form is quite easy, by studying its gradient:

$$\nabla F(x) = 0 \Leftrightarrow Gx + c = 0$$

Thus, minimizing a quadratic form or solving a symmetric linear system are two problems that are equivalent.

10.4.2 Least squares

We suppose that we want to solve a linear system of equations such that the number m of equations is greater than the number n of variables:

$$\begin{cases} a_{1,1}x_1 + \ldots + a_{1,n}x_n &= b_1 \\ & \vdots \\ a_{m,1}x_1 + \ldots + a_{m,n}x_n &= b_m \end{cases}$$

or $Ax = b$. In the general case, the system has no solution. It is then possible to try to find the "least bad" solution, i.e. the vector x that minimizes the sum of the squared residuals:

$$F(x) = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{i,j}x_j - b_i \right)^2 = \|Ax - b\|^2$$

The function F is a quadratic form, that can be written as follows (Theorem (1)):

$$F(x) = \|Ax - b\|^2 = (Ax - b)^t(Ax - b) = x^tA^tAx - 2x^tA^tb + b^tb$$

(the two terms x^tA^tb and b^tAx are equal, since they are scalars, or 1×1 matrices, that are symmetric). The vector x that minimizes F is such that $\nabla F = 0$. Therefore (Theorems (2) and (3)):

$$\nabla F(x) = 2A^tAx - 2A^tb = 0$$

We retrieve the classical formula for the *least squares*, also called *linear regression*. The vector x that minimizes $\|Ax - b\|^2$ satisfies $A^tAx = A^tb$.

10.4.3 Least squares with reduced degrees of freedom

For some algorithms, it may be useful to remove some degrees of freedom of the system, by fixing some parameters of the objective function. Formally, this means that the vector x of parameters is partitioned into two sub-vectors $x = [x_f|x_l]$. The first nf components $x_f = [x_1 \dots x_f]$ correspond to the $f$ree parameters, and the remaining $n-nf$ components $x_l = [x_{nf+1} \dots x_n]$ correspond to the $l$ocked parameters. Thus, the function F solely depends on the vector x_f, and rewrites:

$$F(x_f) = \|Ax - b\|^2 = \left\| [A_f \mid A_l] \begin{bmatrix} x_f \\ \hline \\ x_l \end{bmatrix} - b \right\|^2$$

Partitioning the vector x into two subvectors $[x_f, x_l]$ naturally partitions the matrix A into two submatrices A_f, A_l (in the product Ax, the coefficients of A that weight parameters in x_f are in A_f, and those that weight parameters in x_l are in A_l). It is then possible to isolate the terms that depend of x_f:

$$F(x_f) \quad = \quad \|A_f x_f + A_l x_l - b\|^2$$

By introducing $b' = A_l x_l - b$, and by using the least-squares formula proved in the previous section, one retrieves the equation satisfied by the minimizer x_f of F:

$$A_f^t A_f x_f = A_f^t b' \quad \text{or} \quad A_f^t A_f x_f = A_f^t b - A_f^t A_l x_l$$

10.5 Non-linear optimization

We now focus on the more difficult problem of minimizing non-linear functions. In this section, we present the Newton method, that can compute a stationary point of a non-linear function. Other methods exist and the interested reader may read reference books [Nocedal and Wright, 2006] for more details.

10.5.1 Univariate functions

We first consider the case of a univariate function. The following algorithm iteratively computes a stationary point of the function f:

while $|f'(x)| > \epsilon$
  $p \leftarrow -f'(x)/f''(x)$
  $x \leftarrow x + p$
end // *while*

The algorithm works as follows: at each step, we compute the second order of the function f around the current point x. This approximation is called a *model function*, since it gives a good approximation (or a good model) of the variations of f around x in function of a displacement p:

$$f(x+p) \simeq \tilde{f}_x(p) = f(x) + pf'(x) + p^2/2f''(x)$$

Then, the algorithm computes the displacement p that minimizes the model function. This gives:

$$\begin{array}{lcl} \tilde{f}'_x(p) & = & 0 \\ f'(x) + pf''(x) & = & 0 \\ p & = & -f'(x)/f''(x) \end{array}$$

Finally, the current location is updated: $x \leftarrow x + p$, and the algorithm iterates all the previous steps.

10.5.2 Multivariate functions

The previous (univariate) algorithm can be easily adapted to the multivariate case. As we have seen in the previous section, the second order Taylor expansion of a multivariate function F writes:

$$F(x+p) \simeq \tilde{F}_x(p) = F(x) + p^t \left(\nabla F(x)\right) + 1/2p^t \left(\nabla^2 F(x)\right) p$$

It is clearly a quadratic form. We then find the displacement p p that minimizes this quadratic form:

$$\begin{array}{lcl} \nabla \tilde{F}_x(p) & = & 0 \\ \left(\nabla F(x)\right) + \left(\nabla^2 F(x)\right) p & = & 0 \\ p & = & -\left(\nabla^2 F(x)\right)^{-1} \left(\nabla F(x)\right) \end{array}$$

The Newton algorithm for multivariate functions can then be given by:

while $\|\nabla F(x)\| > \epsilon$
  solve $(\nabla^2 F(x))p = -(\nabla F(x))$
  $x \leftarrow x + p$
end // *while*

In other words, to minimize a non-linear function, the Newton algorithm minimizes a series of quadratic form (by solving a series of linear systems). To solve those linear systems, one may use one of the methods listed in Section 10.2.

10.6 Constrained Optimization: Lagrange Method

We now focus on the problem of constrained optimization. A constrained optimization problem can be formalized by:

$$\text{minimize} F(x) \qquad \text{with} \qquad \begin{cases} G_1(x) &= 0 \\ G_2(x) &= 0 \\ & \vdots \\ G_m(x) &= 0 \end{cases}$$

where F and $G_1, G_2 \dots G_m$ are functions from $\mathbb{R}^n$ to $\mathbb{R}$. The functions G_i are called *constraints*. An important theorem, known as the KKT condition[2], states that the minimizer of F that satisfies the G_i constraints is also a stationary point of the function L, defined by:

$$L(x, \lambda) = F(x) + \sum_{i=1}^{m} \lambda_i G_i(x)$$

The function L, called the Lagrangian of the constrained optimization problem, depends of m additional λ_i parameters, associated with the m constraints G_i.

The stationary points of the Lagrangian L can be found by the Newton method, presented in the previous section.

[2]Named after its authors, Karush, Kuhn and Tucker

Bibliography

J. H. Ahlberg, E. N. Nilson, and J. L. Walsh. *The theory of splines and their applications*. Academic Press, New York, 1967.

B. Aksoylu, A. Khodakovsky, and P. Schröder. Multilevel solvers for unstructured surface meshes. *SIAM Journal on Scientific Computing*, 26(4):1146–1165, 2005.

M. Alexa. Merging polyhedral shapes with scattered features. *The Visual Computer*, 16 (1):26–37, 2000.

M. Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–196, 2002.

B. Allen, B. Curless, and Z. Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics*, 22(3):587–594, 2003. Proceedings of SIGGRAPH 2003.

P. Alliez and C. Gotsman. Recent advances in compression of 3D meshes. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 3–26. Springer, Berlin, Heidelberg, 2005.

P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics*, 22(3):485–493, 2003. Proceedings of SIGGRAPH 2003.

P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. Research report, AIM@SHAPE Network of Excellence, 2005.

D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: Shape completion and animation of people. *ACM Transactions on Graphics*, 24(3): 408–416, 2005. Proceedings of SIGGRAPH 2005.

N. Arad and G. Elber. Isometric texture mapping for free-form surfaces. *Computer Graphics Forum*, 16(5):247–256, 1997.

P. N. Azariadis and N. A. Aspragathos. On using planar developments to perform texture mapping on arbitrarily curved surfaces. *Computers & Graphics*, 24(4):539–554, 2000.

L. Balmelli, G. Taubin, and F. Bernardini. Space-optimized texture maps. *Computer Graphics Forum*, 21(3):411–420, 2002. Proceedings of Eurographics 2002.

A. Belyaev. On transfinite barycentric coordinates. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing (SGP 2006)*, pages 89–99. Eurographics Association, 2006.

C. Bennis, J.-M. Vézien, and G. Iglésias. Piecewise surface flattening for non-distorted texture mapping. *ACM SIGGRAPH Computer Graphics*, 25(4):237–246, 1991. Proceedings of SIGGRAPH '91.

E. Bier and K. Sloan. Two-part texture mappings. *IEEE Computer Graphics and Applications*, 6(9):40–53, 1986.

H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM Transactions on Graphics*, 21(3):312–321, 2002. Proceedings of SIGGRAPH 2002.

H. Birkholz. Shape-preserving parametrization of genus 0 surfaces. In *Proceedings of the 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2004)*, pages 57–64, 2004.

V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of SIGGRAPH '99*, pages 187–194. ACM Press, 1999.

V. Blanz, C. Basso, T. Poggio, and T. Vetter. Reanimating faces in images and video. *Computer Graphics Forum*, 22(3):641–650, 2003. Proceedings of Eurographics 2003.

V. Blanz, K. Scherbaum, T. Vetter, and H.-P. Seidel. Exchanging faces in images. *Computer Graphics Forum*, 23(3):669–676, 2004. Proceedings of Eurographics 2004.

J. F. Blinn. Simulation of wrinkled surfaces. *ACM SIGGRAPH Computer Graphics*, 12 (3):286–292, 1978. Proceedings of SIGGRAPH '78.

I. Boier-Martin, H. Rushmeier, and J. Jin. Parameterization of triangle meshes over quadrilateral domains. In *Proceedings of the Second Eurographics Symposium on Geometry Processing (SGP 2004)*, pages 193–203. Eurographics Association, 2004.

M. Botsch, D. Bommes, and L. Kobbelt. Efficient linear system solvers for mesh processing. In *Mathematics of Surfaces XI*, volume 3604 of *Lecture Notes in Computer Science*, pages 62–83. Springer, Berlin, Heidelberg, 2005. Proceedings of the 11th IMA International Conference.

P. L. Bowers and M. K. Hurdal. Planar conformal mappings of piecewise flat surfaces. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 3–34. Springer, Heidelberg, 2003.

S. C. Brenner and L. R. Scott. *The mathematical theory of finite element methods*, volume 15 of *Texts in Applied Mathematics*. Springer, second edition, 2002.

S. Campagna and H.-P. Seidel. Parameterizing meshes with arbitrary topology. In *Proceedings of Image and Multidimensional Digital Signal Processing '98*, pages 287–290, 1998.

N. A. Carr and J. C. Hart. Meshed atlases for real-time procedural solid texturing. *ACM Transactions on Graphics*, 21(2):106–131, 2002.

N. A. Carr and J. C. Hart. Painting detail. *ACM Transactions on Graphics*, 23(3): 587–594, 2004. Proceedings of SIGGRAPH 2004.

N. A. Carr, J. Hoberock, K. Crane, and J. C. Hart. Rectangular multi-chart geometry images. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing (SGP 2006)*, pages 181–190. Eurographics Association, 2006.

G. Choquet. Sur un type de transformation analytique généralisant la représentation conforme et défiué au moyen de fonctions harmoniques. *Bulletin des Sciences Mathématiques*, 69:156–165, 1945.

F. R. K. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

B. Cipra. You can't always hear the shape of a drum. In *What's Happening in the Mathematical Sciences*, volume 1. American Mathematical Society, Providence, RI, 1993.

U. Clarenz, N. Litke, and M. Rumpf. Axioms and variational problems in surface parameterization. *Computer Aided Geometric Design*, 21(8):727–749, 2004.

D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914, 2004. Proceedings of SIGGRAPH 2004.

C. R. Collins and K. Stephenson. A circle packing algorithm. *Computational Geometry: Theory and Applications*, 25(3):233–256, 2003.

R. Courant. *Dirichlet's Principle, Conformal Mapping, and Minimal Surfaces*. Interscience, New York, 1950.

Y. C. de Verdière. On a new graph invariant and a criterion for planarity. In *Graph Structure Theory*, volume 147 of *Contemporary Mathematics*, pages 137–148. American Mathematical Society, 1993.

P. Degener, J. Meseth, and R. Klein. An adaptable surface parameterization method. In *Proceedings of the 12th International Meshing Roundtable (IMR 2003)*, pages 201–213, 2003.

M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3):209–218, 2002. Proceedings of Eurographics 2002.

M. do Carmo. *Differential geometry of curves and surfaces*. Prentice Hall, 1976.

S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Quadrangulating a mesh using Laplacian eigenvectors. Technical Report UIUCDCS-R-2005-2583, University of Illinois, June 2005.

S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Spectral surface quadrangulation. *ACM Transactions on Graphics*, 25(3):1057–1066, 2006. Proceedings of SIGGRAPH 2006.

J. Douglas. Solution of the problem of Plateau. *Transactions of the American Mathematical Society*, 33(1):263–321, 1931.

T. Duchamp, A. Certain, T. DeRose, and W. Stuetzle. Hierarchical computation of PL harmonic embeddings. Technical report, University of Washington, July 1997.

M. Eck, T. D. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH '95*, pages 173–182. ACM Press, 1995.

I. Eckstein, V. Surazhsky, and C. Gotsman. Texture mapping with hard constraints. *Computer Graphics Forum*, 20(3):95–104, 2001. Proceedings of Eurographics 2001.

J. Erickson and S. Har-Peled. Optimally cutting a surface into a disk. *Discrete & Computational Geometry*, 31(1):37–59, 2004.

J. Erickson and K. Whittlesey. Greedy optimal homotopy and homology generators. In *Proceedings of the 16th Symposium on Discrete Algorithms (SODA '05)*, pages 1038–1046. SIAM, 2005.

G. Farin. Surfaces over Dirichlet tessellations. *Computer Aided Geometric Design*, 7 (1-4):281–292, 1990.

H. Ferguson and A. Rockwood. Multiperiodic functions for surface design. *Computer Aided Geometric Design*, 10(3–4):315–328, 1993.

M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23: 298–305, 1973.

M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25:619–633, 1975.

E. Fiume, A. Fournier, and V. Canale. Conformal texture mapping. In *Proceedings of Eurographics '87*, pages 53–64, 1987.

M. S. Floater. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

M. S. Floater. Parametric tilings and scattered data approximation. *International Journal of Shape Modeling*, 4(3–4):165–182, 1998.

M. S. Floater. Meshless parameterization and B-spline surface approximation. In R. Cipolla and R. Martin, editors, *The Mathematics of Surfaces IX*, pages 1–18, London, 2000. Springer.

M. S. Floater. Convex combination maps. In J. Levesley, I. J. Anderson, and J. C. Mason, editors, *Algorithms for Approximation IV*, pages 18–23, 2002.

M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003a.

M. S. Floater. One-to-one piecewise linear mappings over triangulations. *Mathematics of Computation*, 72(242):685–696, 2003b.

M. S. Floater and K. Hormann. Parameterization of triangulations and unorganized points. In A. Iske, E. Quak, and M. S. Floater, editors, *Tutorials on Multiresolution in Geometric Modelling*, Mathematics and Visualization, pages 287–316. Springer, Berlin, Heidelberg, 2002.

M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer, Berlin, Heidelberg, 2005.

M. S. Floater and M. Reimers. Meshless parameterization and surface reconstruction. *Computer Aided Geometric Design*, 18(2):77–92, 2001.

M. S. Floater, K. Hormann, and M. Reimers. Parameterization of manifold triangulations. In C. K. Chui, L. L. Schumaker, and J. Stöckler, editors, *Approximation Theory X: Abstract and Classical Analysis*, Innovations in Applied Mathematics, pages 197–209. Vanderbilt University Press, Nashville, TN, 2002.

M. S. Floater, G. Kós, and M. Reimers. Mean value coordinates in 3D. *Computer Aided Geometric Design*, 22(7):623–631, 2005.

M. S. Floater, K. Hormann, and G. Kós. A general construction of barycentric coordinates over convex polygons. *Advances in Computational Mathematics*, 24(1–4): 311–331, 2006.

J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Boston, second edition, 1995.

V. A. Garanzha. Maximum norm optimization of quasi-isometric mappings. *Numerical Linear Algebra with Applications*, 9(6,7):493–510, 2002.

M. Garland, A. Willmott, and P. S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics (SI3D '01)*, pages 49–58. ACM Press, 2001.

C. F. Gauß. Disquisitiones generales circa superficies curvas. *Commentationes Societatis Regiæ Scientiarum Gottingensis Recentiores*, 6:99–146, 1827.

S. J. Gortler, C. Gotsman, and D. Thurston. Discrete one-forms on meshes and applications to 3D mesh parameterization. *Computer Aided Geometric Design*, 23(2):83–112, 2006.

C. Gotsman. On graph partitioning, spectral analysis, and digital mesh processing. In *Proceedings of Shape Modeling International (SMI '03)*, pages 165–171. IEEE Computer Society, 2003.

C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3D meshes. *ACM Transactions on Graphics*, 22(3):358–363, 2003. Proceedings of SIGGRAPH 2003.

G. Greiner. Variational design and fairing of spline surfaces. *Computer Graphics Forum*, 13(3):143–154, 1994. Proceedings of Eurographics '94.

G. Greiner and K. Hormann. Interpolating and approximating scattered 3D-data with hierarchical tensor product B-splines. In A. L. Méhauté, C. Rabut, and L. L. Schumaker, editors, *Surface Fitting and Multiresolution Methods*, Innovations in Applied Mathematics, pages 163–172. Vanderbilt University Press, Nashville, 1997.

C. M. Grimm and J. F. Hughes. Parameterizing N-holed tori. In *Mathematics of Surfaces*, volume 2768 of *Lecture Notes in Computer Science*, pages 14–29. Springer, Berlin, Heidelberg, 2003. Proceedings of the 10th IMA International Conference.

C. M. Grimm and J. F. Hughes. Modeling surfaces of arbitrary topology using manifolds. In *Proceedings of SIGGRAPH '95*, pages 359–368. ACM Press, 1995.

X. Gu and S.-T. Yau. Computing conformal structures of surfaces. *Communications in Information and Systems*, 2(2):121–146, 2002.

X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proceedings of the First Eurographics Symposium on Geometry Processing (SGP 2003)*, pages 127–137. Eurographics Association, 2003.

X. Gu, S. J. Gortler, and H. Hoppe. Geometry images. *ACM Transactions on Graphics*, 21(3):355–361, 2002. Proceedings of SIGGRAPH 2002.

X. Gu, Y. Wang, and S.-T. Yau. Computing conformal invariants: Period matrices. *Communications in Information and Systems*, 3(3):153–170, 2003a.

X. Gu, Y. Wang, and S.-T. Yau. Geometric compression using Riemann surface structure. *Communications in Information and Systems*, 3(3):171–182, 2003b.

X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, and S.-T. Yau. Genus zero surface conformal mapping and its application to brain surface mapping. *IEEE Transaction on Medical Imaging*, 23(8):949–958, 2004a.

X. Gu, Y. Wang, and S.-T. Yau. Multiresolution computation of conformal structures of surfaces. *Journal of Systemics, Cybernetics and Informatics*, 1(5):45–50, 2004b.

I. Guskov. An anisotropic mesh parameterization scheme. *Engineering with Computers*, 20(2):129–135, 2004.

I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH '99*, pages 325–334. ACM Press, 1999.

I. Guskov, K. Vidimče, W. Sweldens, and P. Schröder. Normal meshes. In *Proceedings of SIGGRAPH 2000*, pages 95–102. ACM Press, 2000.

I. Guskov, A. Khodakovsky, P. Schröder, and W. Sweldens. Hybrid meshes: Multiresolution using regular and irregular refinement. In *Proceedings of the 18th Annual Symposium on Computational Geometry (SCG '02)*, pages 264–272. ACM Press, 2002.

S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):181–189, 2000.

Y. He, X. Gu, and H. Qin. Rational spherical splines for genus zero shape modeling. In *Proceedings of the 2005 International Conference on Shape Modeling and Applications (SMI '05)*, pages 82–91. IEEE Computer Society, 2005.

A. Hirani. *Discrete Exterior Calculus*. PhD thesis, California Institute of Technology, Pasadena, CA, 2003.

H. Hiyoshi and K. Sugihara. Voronoi-based interpolation with higher continuity. In *Proceedings of the 16th Annual Symposium on Computational Geometry (SCG '00)*, pages 242–250. ACM Press, 2000.

H. Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH '96*, pages 99–108. ACM Press, 1996.

H. Hoppe and E. Praun. Shape compression using spherical geometry images. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 27–46. Springer, Berlin, Heidelberg, 2005.

K. Hormann. Fitting free form surfaces. In B. Girod, G. Greiner, and H. Niemann, editors, *Principles of 3D Image Analysis and Synthesis*, volume 556 of *The International Series in Engineering and Computer Science*, chapter 4.7, pages 192–202. Kluwer Academic Publishers, Boston, MA, 2000.

K. Hormann. *Theory and Applications of Parameterizing Triangulations*. PhD thesis, Department of Computer Science, University of Erlangen, Nov. 2001.

K. Hormann and M. S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics*, 25(4):1424–1441, 2006.

K. Hormann and G. Greiner. MIPS: An efficient global parametrization method. In P.-J. Laurent, P. Sablonnière, and L. L. Schumaker, editors, *Curve and Surface Design: Saint-Malo 1999*, Innovations in Applied Mathematics, pages 153–162. Vanderbilt University Press, Nashville, TN, 2000a.

K. Hormann and G. Greiner. Quadrilateral remeshing. In B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, *Proceedings of Vision, Modeling, and Visualization 2000*, pages 153–162, Saarbrücken, Germany, 2000b. infix.

K. Hormann and M. Reimers. Triangulating point clouds with spherical topology. In T. Lyche, M.-L. Mazure, and L. L. Schumaker, editors, *Curve and Surface Design: Saint-Malo 2002*, Modern Methods in Applied Mathematics, pages 215–224. Nashboro Press, Brentwood, TN, 2003.

K. Hormann and M. Tarini. A quadrilateral rendering primitive. In *Proceedings of the 2004 Workshop on Graphics Hardware (GH 2004)*, Eurographics Symposium Proceedings, pages 7–14. Eurographics Association, 2004.

K. Hormann, G. Greiner, and S. Campagna. Hierarchical parametrization of triangulated surfaces. In B. Girod, H. Niemann, and H.-P. Seidel, editors, *Proceedings of Vision, Modeling, and Visualization 1999*, pages 219–226, Erlangen, Germany, 1999. infix.

K. Hormann, U. Labsik, and G. Greiner. Remeshing triangulated surfaces with optimal parameterizations. *Computer-Aided Design*, 33(11):779–788, 2001.

M. K. Hurdal, P. L. Bowers, K. Stephenson, D. W. L. Sumners, K. Rehm, K. Schaper, and D. A. Rottenberg. Quasi-conformally flat mapping the human cerebellum. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI '99)*, volume 1679 of *Lecture Notes in Computer Science*, pages 279–286. Springer, Berlin, Heidelberg, 1999.

T. Igarashi and D. Cosgrove. Adaptive unwrapping for interactive texture painting. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics (SI3D '01)*, pages 209–216. ACM Press, 2001.

M. Isenburg and P. Lindstrom. Streaming meshes. In *Proceedings of IEEE Visualization 2005*, pages 231–238. IEEE Computer Society, 2005.

M. Isenburg, S. Gumhold, and C. Gotsman. Connectivity shapes. In *Proceedings of IEEE Visualization 2001*, pages 135–142. IEEE Computer Society, 2001.

M. Jin, Y. Wang, S.-T. Yau, and X. Gu. Optimal global conformal surface parameterization. In *Proceedings of IEEE Visualization 2004*, pages 267–274. IEEE Computer Society, 2004.

M. Jin, F. Luo, and X. Gu. Computing surface hyperbolic structure and real projective structure. In *Proceedings of the 2006 Symposium on Solid and Physical Modeling (SPM '06)*, pages 105–116. ACM Press, 2006.

T. Ju, S. Schaefer, and J. Warren. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics*, 24(3):561–566, 2005a. Proceedings of ACM SIGGRAPH 2005.

T. Ju, S. Schaefer, J. Warren, and M. Desbrun. A geometric construction of coordinates for convex polyhedra using polar duals. In *Proceedings of the Third Eurographics Symposium on Geometry Processing (SGP 2005)*, pages 181–186. Eurographics Association, 2005b.

T. Ju, P. Liepa, and J. Warren. A general geometric construction of coordinates in a convex simplicial polytope. *Computer Aided Geometric Design*, 24(3):161–178, 2007.

D. Julius, V. Kraevoy, and A. Sheffer. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum*, 24(3):581–590, 2005. Proceedings of Eurographics 2005.

M. Kac. Can one hear the shape of a drum? *American Mathematical Monthly*, 73(4), 1966.

Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *Proceedings of SIGGRAPH 2000*, pages 279–286. ACM Press, 2000.

Z. Karni, C. Gotsman, and S. J. Gortler. Free-boundary linear parameterization of 3D meshes in the presence of constraints. In *Proceedings of the 2005 International Conference on Shape Modeling and Applications (SMI '05)*, pages 266–275. IEEE Computer Society, 2005.

L. Kharevych, B. Springborn, and P. Schröder. Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics*, 25(2):412–438, 2006.

A. Khodakovsky, N. Litke, and P. Schröder. Globally smooth parameterizations with low distortion. *ACM Transactions on Graphics*, 22(3):350–357, 2003. Proceedings of SIGGRAPH 2003.

W. Klingenberg. *A Course in Differential Geometry*. Springer, Berlin, Heidelberg, 1978.

H. Kneser. Lösung der Aufgabe 41. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 35:123–124, 1926.

L. P. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel. A shrink-wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum*, 18(3):119–130, 1999. Proceedings of Eurographics 1999.

S. Kolmanič and N. Guid. The flattening of arbitrary surfaces by approximation with developable stripes. In U. Cugini and M. J. Wozny, editors, *From geometric modeling to shape modeling*, volume 80 of *International Federation for Information Processing*, pages 35–46. Kluwer Academic Publishers, Boston, 2001.

Y. Koren. On spectral graph drawing. In *Computing and Combinatorics*, volume 2697 of *Lecture Notes in Computer Science*, pages 496–508. Springer, Berlin, Heidelberg, 2003. Proceedings of the 9th Annual International Conference (COCOON 2003).

G. Kós and T. Várady. Parameterizing complex triangular meshes. In T. Lyche, M.-L. Mazure, and L. L. Schumaker, editors, *Curve and Surface Design: Saint-Malo 2002*, Modern Methods in Mathematics, pages 265–274. Nashboro Press, Brentwood, TN, 2003.

V. Kraevoy and A. Sheffer. Cross-parameterization and compatible remeshing of 3D models. *ACM Transactions on Graphics*, 23(3):861–869, 2004. Proceedings of SIGGRAPH 2004.

V. Kraevoy and A. Sheffer. Template based mesh completion. In *Proceedings of the Third Eurographics Symposium on Geometry Processing (SGP 2005)*, pages 13–22. Eurographics Association, 2005.

V. Kraevoy, A. Sheffer, and C. Gotsman. Matchmaker: constructing constrained texture maps. *ACM Transactions on Graphics*, 22(3):326–333, 2003. Proceedings of SIGGRAPH 2003.

E. Kreyszig. *Differential Geometry*. Dover, New York, 1991.

U. Labsik, K. Hormann, and G. Greiner. Using most isometric parametrizations for remeshing polygonal surfaces. In R. Martin and W. Wang, editors, *Proceedings of Geometric Modeling and Processing 2000*, pages 220–228, Hong Kong, China, 2000. IEEE Computer Society.

J. H. Lambert. *Beyträge zum Gebrauche der Mathematik und deren Anwendung*, volume 3. Buchhandlung der Realschule, Berlin, 1772.

T. Langer, A. Belyaev, and H.-P. Seidel. Spherical barycentric coordinates. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing (SGP 2006)*, pages 81–88. Eurographics Association, 2006.

T. Langer, A. Belyaev, and H.-P. Seidel. Mean value coordinates for arbitrary spherical polygons and polyhedra in $\mathbb{R}^3$. In P. Chenin, T. Lyche, and L. L. Schumaker, editors, *Curve and Surface Design: Avignon 2006*, Modern Methods in Applied Mathematics, pages 193–202. Nashboro Press, Brentwood, TN, 2007.

F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust. Computing a canonical polygonal schema of an orientable triangulated surface. In *Proceedings of the 17th Annual Symposium on Computational Geometry (SCG '01)*, pages 80–89. ACM Press, 2001.

A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Proceedings of SIGGRAPH 2000*, pages 85–94. ACM Press, 2000.

A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *Proceedings of SIGGRAPH '98*, pages 95–104. ACM Press, 1998.

A. W. F. Lee, D. Dobkin, W. Sweldens, and P. Schröder. Multiresolution mesh morphing. In *Proceedings of SIGGRAPH '99*, pages 343–350. ACM Press, 1999.

E. T. Y. Lee. Choosing nodes in parametric curve interpolation. *Computer-Aided Design*, 21(6):363–370, 1989.

Y. Lee, H. S. Kim, and S. Lee. Mesh parameterization with a virtual boundary. *Computers & Graphics*, 26(5):677–686, 2002.

J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe. Real-time fur over arbitrary surfaces. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics (SI3D '01)*, pages 227–232. ACM Press, 2001.

B. Lévy. Constrained texture mapping for polygonal meshes. In *Proceedings of SIGGRAPH 2001*, pages 417–424. ACM Press, 2001.

B. Lévy. Dual domain extrapolation. *ACM Transactions on Graphics*, 22(3):364–369, 2003. Proceedings of SIGGRAPH 2003.

B. Lévy. Deformation analysis on a manifold. Technical report, INRIA-ALICE, 2007. `http://alice.loria.fr/publications`.

B. Lévy. Laplace-Beltrami eigenfunctions: Towards an algorithm that "understands" geometry. In *Proceedings of the 2006 International Conference on Shape Modeling and Applications (SMI '06)*, pages 65–72. IEEE Computer Society, 2006.

B. Lévy and J.-L. Mallet. Non-distorted texture mapping for sheared triangulated meshes. In *Proceedings of SIGGRAPH '98*, pages 343–352. ACM Press, 1998.

B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3):362–371, 2002. Proceedings of SIGGRAPH 2002.

W.-C. Li, N. Ray, and B. Lévy. Automatic and interactive mesh to T-spline conversion. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing (SGP 2006)*, pages 191–200. Eurographics Association, 2006.

J. Liesen, E. de Sturler, A. Sheffer, Y. Aydin, and C. Siefert. Preconditioners for indefinite linear systems arising in surface parameterization. In *Proceedings of the 9th International Meshing Roundtable*, pages 71–82, 2001.

N. Litke, M. Droske, M. Rumpf, and P. Schröder. An image processing approach to surface matching. In *Proceedings of the Third Eurographics Symposium on Geometry Processing (SGP 2005)*, pages 207–216. Eurographics Association, 2005.

F. Losasso, H. Hoppe, S. Schaefer, and J. Warren. Smooth geometry images. In *Proceedings of the First Eurographics Symposium on Geometry Processing (SGP 2003)*, pages 138–145. Eurographics Association, 2003.

L. Lovász and A. Schrijver. On the null space of a Colin de Verdière matrix. *Annales de l'institut Fourier*, 49(3):1017–1026, 1999.

S. D. Ma and H. Lin. Optimal texture mapping. In *Proceedings of Eurographics '88*, pages 421–428, 1988.

W. Ma and J. P. Kruth. Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces. *Computer-Aided Design*, 27(9):663–675, 1995.

J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *Proceedings of SIGGRAPH '93*, pages 27–34. ACM Press, 1993.

E. A. Malsch and G. Dasgupta. Interpolations for temperature distributions: A method for all non-concave polygons. *International Journal of Solids and Structures*, 41(8): 2165–2188, 2004.

E. A. Malsch and G. Dasgupta. Algebraic construction of smooth interpolants on polygonal domains. *The Mathematica Journal*, 9(3):641–658, 2005.

E. A. Malsch, J. J. Lin, and G. Dasgupta. Smooth two dimensional interpolants: a recipe for all polygons. *Journal of Graphics Tools*, 10(2):27–39, 2005.

S. R. Marschner, B. K. Guenter, and S. Raghupathy. Modeling and rendering for realistic facial animation. In B. Péroche and H. Rushmeier, editors, *Rendering Techniques 2000*, pages 231–242. Springer, Wien, New York, 2000. Proceedings of the 11th Eurographics Workshop on Rendering.

J. McCartney, B. K. Hinds, and B. L. Seow. The flattening of triangulated surfaces incorporating darts and gussets. *Computer-Aided Design*, 31(4):249–260, 1999.

W. H. Meeks. A survey of the geometric results in the classical theory of minimal surfaces. *Bulletin of the Brazilian Mathematical Society*, 12(1):29–86, 1981.

G. Mercator. Nova et aucta orbis terrae descriptio ad usum navigantium emendate accommodata. Duisburg, 1569.

O. Meshar, D. Irony, and S. Toledo. An out-of-core sparse symmetric indefinite factorization method. *ACM Transactions on Mathematical Software*, 32(3):445–471, 2006.

M. Meyer, H. Lee, A. H. Barr, and M. Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools*, 7(1):13–22, 2002.

V. J. Milenkovic. Rotational polygon containment and minimum enclosure. In *Proceedings of the 14th Annual Symposium on Computational Geometry (SCG '98)*, pages 1–8. ACM Press, 1998.

J. Mitani and H. Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Transactions on Graphics*, 23(3):259–263, 2004. Proceedings of SIGGRAPH 2004.

F. Morgan. *Riemannian Geometry: A Beginner's Guide*. A K Peters, Wellesley, MA, second edition, 1998.

T. Needham. *Visual Complex Analysis*. Oxford University Press, Oxford, New York, 1997.

X. Ni, M. Garland, and J. C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics*, 23(3):613–622, 2004. Proceedings of SIGGRAPH 2004.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, second edition, 2006.

J. Pach and R. Wenger. Embedding planar graphs at fixed vertex locations. In *Graph Drawing*, volume 1547 of *Lecture Notes in Computer Science*, pages 263–274. Springer, Berlin, Heidelberg, 1998. Proceedings of the 6th International Symposium (GD '98).

L. Parida and S. P. Mudur. Constraint-satisfying planar development of complex surfaces. *Computer-Aided Design*, 25(4):225–232, 1993.

H. K. Pedersen. Decorating implicit surfaces. In *Proceedings of SIGGRAPH '95*, pages 291–300. ACM Press, 1995.

J. Peng, D. Kristjansson, and D. Zorin. Interactive modeling of topologically complex geometric detail. *ACM Transactions on Graphics*, 23(3):635–643, 2004. Proceedings of SIGGRAPH 2004.

U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

D. Piponi and G. Borshukov. Seamless texture mapping of subdivision surfaces by model pelting and texture blending. In *Proceedings of SIGGRAPH 2000*, pages 471–478. ACM Press, 2000.

J. A. F. Plateau. *Statistique Expérimentale et Théorique des Liquides Soumis aux Seules Forces Moléculaires*. Gauthier-Villars, Paris, 1873.

S. D. Porumbescu, B. Budge, L. Feng, and K. I. Joy. Shell maps. *ACM Transactions on Graphics*, 24(3):626–633, 2005. Proceedings of SIGGRAPH 2005.

E. Praun and H. Hoppe. Spherical parametrization and remeshing. *ACM Transactions on Graphics*, 22(3):340–349, 2003. Proceedings of SIGGRAPH 2003.

E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proceedings of SIGGRAPH 2000*, pages 465–470. ACM Press, 2000.

E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. In *Proceedings of SIGGRAPH 2001*, pages 179–184. ACM Press, 2001.

C. Ptolemy. *The Geography*. Dover, 1991. Translated by E. L. Stevenson.

B. Purnomo, J. D. Cohen, and S. Kumar. Seamless texture atlases. In *Proceedings of the Second Eurographics Symposium on Geometry Processing (SGP 2004)*, pages 67–76. Eurographics Association, 2004.

T. Radó. Aufgabe 41. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 35:49, 1926.

T. Radó. The problem of least area and the problem of Plateau. *Mathematische Zeitschrift*, 32:763–796, 1930.

N. Ray and B. Lévy. Hierarchical least squares conformal maps. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications (PG 2003)*, pages 263–270. IEEE Computer Society, 2003.

N. Ray, W. C. Li, B. Vallet, F. Boutantin, B. Lévy, and C. Borgese. Graphite, 2003. `http://alice.loria.fr/software/graphite/`.

N. Ray, W.-C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Transactions on Graphics*, 25(4):1460–1485, 2006.

M. Reuter, F.-E. Wolter, and N. Peinecke. Laplace-Beltrami spectra as 'Shape-DNA' of surfaces and solids. *Computer-Aided Design*, 38(4):342–366, 2006.

B. Riemann. *Grundlagen für eine allgemeine Theorie der Functionen einer veränderlichen complexen Größe*. PhD thesis, Universität Göttingen, 1851.

A. Rockwood and H. Park. Interactive design of smooth genus N objects using multi-periodic functions and applications. *International Journal of Shape Modeling*, 5(2): 135–157, 1999.

B. Rodin and D. Sullivan. The convergence of circle packings to the Riemann mapping. *Journal of Differential Geometry*, 26(2):349–360, 1987.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

S. Saba, I. Yavneh, C. Gotsman, and A. Sheffer. Practical spherical embedding of manifold triangle meshes. In *Proceedings of the 2005 International Conference on Shape Modeling and Applications (SMI '05)*, pages 256–265. IEEE Computer Society, 2005.

P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Proceedings of SIGGRAPH 2001*, pages 409–416. ACM Press, 2001.

P. V. Sander, S. J. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parametrization. In *Proceedings of the 13th Eurographics Workshop on Rendering (EGWR '02)*, pages 87–98. Eurographics Association, 2002.

P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proceedings of the First Eurographics Symposium on Geometry Processing (SGP 2003)*, pages 138–145. Eurographics Association, 2003.

S. Schaefer, T. Ju, and J. Warren. A unified, integral construction for coordinates over closed curves. *Computer Aided Geometric Design*, 2007. To appear.

J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe. Inter-surface mapping. *ACM Transactions on Graphics*, 23(3):870–877, 2004. Proceedings of SIGGRAPH 2004.

E. L. Schwartz, B. Merker, E. Wolfson, and A. Shaw. Applications of computer graphics and image processing to 2D and 3D modeling of the functional architecture of visual cortex. *IEEE Computer Graphics and Applications*, 8(4):13–23, 1988.

E. L. Schwartz, A. Shaw, and E. Wolfson. A numerical solution to the generalized mapmaker's problem: flattening nonconvex polyhedral surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(9):1005–1008, 1989.

A. Shamir. A formulation of boundary mesh segmentation. In *Proceedings of the Second International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2004)*, pages 82–89. IEEE Computer Society, 2004.

A. Shapiro and A. Tal. Polyhedron realization for shape transformation. *The Visual Computer*, 14(8–9):429–444, 1998.

A. Sheffer. Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In *Proceedings of Shape Modeling International (SMI '02)*, pages 61–66. IEEE Computer Society, 2002.

A. Sheffer. Non-optimal parameterization and user control. In T. Lyche, M.-L. Mazure, and L. L. Schumaker, editors, *Curve and Surface Design: Saint-Malo 2002*, Modern Methods in Applied Mathematics, pages 355–364. Nashboro Press, Brentwood, TN, 2003a.

A. Sheffer. Skinning 3D meshes. *Graphical Models*, 65(5):274–285, 2003b.

A. Sheffer and E. de Sturler. Surface parameterization for meshing by triangulation flattening. In *Proceedings of the 9th International Meshing Roundtable (IMR 2000)*, pages 161–172, 2000.

A. Sheffer and E. de Sturler. Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with Computers*, 17(3):326–337, 2001.

A. Sheffer and E. de Sturler. Smoothing an overlay grid to minimize linear distortion in texture mapping. *ACM Transactions on Graphics*, 21(4):874–890, 2002.

A. Sheffer and J. C. Hart. Seamster: inconspicuous low-distortion texture seam layout. In *Proceedings of IEEE Visualization 2002*, pages 291–298. IEEE Computer Society, 2002.

A. Sheffer, C. Gotsman, and N. Dyn. Robust spherical parametrization of triangular meshes. In D. Cohen-Or, N. Dyn, G. Elber, and A. Shamir, editors, *Proceedings of the 4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics*, pages 94–99, Tel Aviv, Israel, 2003.

A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov. ABF++: fast and robust angle based flattening. *ACM Transactions on Graphics*, 24(2):311–330, 2005.

A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision*, 2(2):105–171, 2006.

J. R. Shewchuk. An introduction to the conjugate method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University, 1994. `http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`.

T. Shimada and Y. Tada. Approximate transformation of an arbitrary curved surface into a plane using dynamic programming. *Computer-Aided Design*, 23(2):153–159, 1991.

R. Sibson. A vector identity for the Dirichlet tesselation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 87:151–155, 1980.

R. Sibson. A brief description of natural neighbour interpolation. In V. Barnett, editor, *Interpolating Multivariate Data*, pages 21–36. Wiley, New York, 1981.

C. Soler, M.-P. Cani, and A. Angelidis. Hierarchical pattern mapping. *ACM Transactions on Graphics*, 21(3):673–680, 2002. Proceedings of SIGGRAPH 2002.

O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. Bounded-distortion piecewise mesh parametrization. In *Proceedings of IEEE Visualization 2002*, pages 355–362. IEEE Computer Society, 2002.

O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the Second Eurographics Symposium on Geometry Processing (SGP 2004)*, pages 179–188. Eurographics Association, 2004.

D. Steiner and A. Fischer. Planar parameterization for closed manifold genus-g meshes using any type of positive weights. *Journal of Computing and Information Science in Engineering*, 5(2):118–125, 2005.

N. Sukumar and E. A. Malsch. Recent advances in the construction of polygonal finite element interpolants. *Archives of Computational Methods in Engineering*, 13(1):129–163, 2006.

R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 23(3):399–405, 2004. Proceedings of ACM SIGGRAPH 2004.

V. Surazhsky and C. Gotsman. Explicit surface remeshing. In *Proceedings of the First Eurographics Symposium on Geometry Processing (SGP 2003)*, pages 20–30. Eurographics Association, 2003.

M. Tarini, K. Hormann, P. Cignoni, and C. Montani. PolyCube-Maps. *ACM Transactions on Graphics*, 23(3):853–860, 2004. Proceedings of ACM SIGGRAPH 2004.

G. Taubin. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH '95*, pages 351–358. ACM Press, 1995.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

G. Tewari, J. Snyder, P. V. Sander, S. J. Gortler, and H. Hoppe. Signal-specialized parameterization for piecewise linear reconstruction. In *Proceedings of the Second Eurographics Symposium on Geometry Processing (SGP 2004)*, pages 57–66. Eurographics Association, 2004.

Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Designing quadrangulations with discrete harmonic forms. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing (SGP 2006)*, pages 201–210. Eurographics Association, 2006.

G. Turk. Texture synthesis on surfaces. In *Proceedings of SIGGRAPH 2001*, pages 347–354. ACM Press, 2001.

W. T. Tutte. Convex representations of graphs. *Proceedings of the London Mathematical Society*, 10:304–320, 1960.

W. T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13:743–767, 1963.

B. Vallet and B. Lévy. Spectral geometry processing with manifold harmonics. Technical report, INRIA-ALICE, 2007.

E. L. Wachspress. *A Rational Finite Element Basis*. Academic Press, New York, 1975.

C. C. L. Wang, S. S.-F. Smith, and M. M. F. Yuen. Surface flattening based on energy model. *Computer-Aided Design*, 34(11):823–833, 2002.

J. Warren. Barycentric coordinates for convex polytopes. *Advances in Computational Mathematics*, 6(2):97–108, 1996.

J. Warren. On the uniqueness of barycentric coordinates. In R. Goldman and R. Krasauskas, editors, *Topics in Algebraic Geometry and Geometric Modeling*, volume 334 of *Contemporary Mathematics*, pages 93–99. American Mathematical Society, 2003.

J. Warren, S. Schaefer, A. N. Hirani, and M. Desbrun. Barycentric coordinates for convex sets. *Advances in Computational Mathematics*, 2007. To appear.

L.-Y. Wei and M. Levoy. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of SIGGRAPH 2001*, pages 355–360. ACM Press, 2001.

L. Ying and D. Zorin. A simple manifold-based construction of surfaces of arbitrary smoothness. *ACM Transactions on Graphics*, 23(3):271–275, 2004. Proceedings of SIGGRAPH 2004.

L. Ying, A. Hertzmann, H. Biermann, and D. Zorin. Texture and shape synthesis on surfaces. In S. J. Gortler and K. Myszkowski, editors, *Rendering Techniques 2001*, pages 301–312. Springer, Wien, New York, 2001. Proceedings of the 12th Eurographics Workshop on Rendering.

S. Yoshizawa, A. G. Belyaev, and H.-P. Seidel. A fast and simple stretch-minimizing mesh parameterization. In *Proceedings of the 2004 International Conference on Shape Modeling and Applications (SMI '04)*, pages 200–208. IEEE Computer Society, 2004.

F. W. Young. Multidimensional scaling. In S. Kotz, N. L. Johnson, and C. B. Read, editors, *Encyclopedia of Statistical Sciences*, volume 5, pages 649–659. Wiley, New York, 1985.

R. Zayer, C. Rössl, and H.-P. Seidel. Convex boundary angle based flattening. In T. Ertl, B. Girod, G. Greiner, H. Niemann, H.-P. Seidel, E. Steinbach, and R. Westermann, editors, *Proceedings of Vision, Modeling, and Visualization 2003*, pages 281–288, München, Germany, Nov. 2003. infix.

R. Zayer, C. Rössl, and H.-P. Seidel. Discrete tensorial quasi-harmonic maps. In *Proceedings of the 2005 International Conference on Shape Modeling and Applications (SMI '05)*, pages 276–285. IEEE Computer Society, 2005a.

R. Zayer, C. Rössl, and H.-P. Seidel. Setting the boundary free: A composite approach to surface parameterization. In *Proceedings of the Third Eurographics Symposium on Geometry Processing (SGP 2005)*, pages 91–100. Eurographics Association, 2005b.

R. Zayer, C. Rössl, and H.-P. Seidel. Variations on angle based flattening. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 187–199. Springer, Berlin, Heidelberg, 2005c.

R. Zayer, C. Rössl, and H.-P. Seidel. Curvilinear spherical parameterization. In *Proceedings of the 2006 International Conference on Shape Modeling and Applications (SMI '06)*, pages 57–64. IEEE Computer Society, 2006.

E. Zhang, K. Mischaikow, and G. Turk. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics*, 24(1):1–27, 2005.

K. Zhou, J. Synder, B. Guo, and H.-Y. Shum. Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the Second Eurographics Symposium on Geometry Processing (SGP 2004)*, pages 47–56. Eurographics Association, 2004.

K. Zhou, X. Wang, Y. Tong, M. Desbrun, B. Guo, and H.-Y. Shum. TextureMontage: Seamless texturing of arbitrary surfaces from multiple images. *ACM Transactions on Graphics*, 24(3):1148–1155, 2005. Proceedings of SIGGRAPH 2005.

G. Zigelman, R. Kimmel, and N. Kiryati. Texture mapping using surface flattening via multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):198–207, 2002.