BIJECTIVE PARAMETERIZATION WITH FREE BOUNDARIES

A Dissertation

by

JASON DEAN SMITH

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Scott Schaefer
Committee Members,	John Keyser
	Jinxiang Chai
	Ergun Akleman
Head of Department,	Dilma Da Silva

December 2015

Major Subject: Computer Science

ABSTRACT

When displaying 3D surfaces onto computer screens, additional information is often mapped onto the surface to enhance the quality of the rendering. Surface parameterization generates a correspondence, or mapping, between the 3D surface and 2D parameterization space. This mapping has many applications in computer graphics, but in most cases cannot be performed without introducing large distortions in the 2D parameterization. Along with problems of distortion, the mapping of the 2D space to 3D for many applications can be invalidated if the property of bijectivity is violated. While there is previous research guaranteeing bijectivity, these methods must constrain or modify the boundary of the 2D parameterization. This dissertation, describes a fully automatic method for generating guaranteed bijective surface parameterizations from triangulated 3D surfaces. In particular, a new isometric distortion energy metric is introduced preventing local folds of triangles in the parameterization as well as a barrier function that prevents intersection of the 2D boundaries. By using a computationally efficient isometric metric energy, the dissertation achieves fast and comparable optimization times to previous methods. The boundary of the parameterization is free to change shape during the optimization to minimize distortion. A new optimization approach is introduced called singularity aware optimization and in conjunction with an interior point approach and barrier energy functions guarantee bijectivity. This optimization framework is then modified to allow for an importance weighting allowing for customizable and more efficient texel usage.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Scott Schaefer for his mentorship. It was Scott who originally peeked my interest in Computer Graphics. He has supported and guided me throughout my studies and has always gone above and beyond to help me and his students succeed. It was an absolute honor to work with Scott, and I hope to continue working with him in the future. Second, I thank my committee, John Keyser, Jinxiang Chai, and Ergun Akleman for the assistance and lessons they have given me. Not only have their courses grown my interest in Computer Graphics, I have developed many proffesional relationships and friendships from their research groups. Next, I want to thank my parents, my mom, Tammy Smith, and dad, Ron Smith. They have always supported my pursuit of higher education, and have given me a loving home. Finally, thanks to my loving fiancée and soon to be wife, Adrienne Gilligan. Without her constant support and motivation I would not be who I am today. I look forward to our future together and returning the love she has shown me.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Surface parameterization can be viewed as producing a correspondence between a 3D surface to another domain. Building these correspondences is a key research problem in computer graphics. The parametric domain corresponding to the 3D surface is generally a form of surface itself and the idea of surface parameterization can be generalized to producing a mapping from one surface to another. Surface parameterization, traces its origins back to Cartography when travelers wanted to flatten the spherical earth into flat maps on a plane for visualization purposes. This process of flattening was one of the first forms of parameterization and many various approaches have been developed all with their advantages and disadvantages. Now, in computer graphics, there is a wide variety of applications for parameterization including, remeshing, inter-surface matching, tetrahedralization, detail-mapping and transfer, shape-analysis, and most importantly texture mapping. In this dissertation, I discuss my contributions to a new state of the art parameterization process to guarantee bijectivity during the parameterization process.

1.1 Texture Mapping

Visualizing 3D surfaces is a main concentration of computer graphics. Representing these surfaces as well as rendering them to images or screen with high quality and efficiency is an important research topic. It is common practice to discretize 3D surfaces into polygonal models allowing for efficient storage and rendering algorithms. However, geometry is only one aspect of how we perceive an object. To achieve better realism, more information is required to render surfaces such as textures and lighting. To achieve this, typically artists must annotate discretized surfaces with additional information such as color, transparency, specularity, and other attributes

Figure 1.1: 3D surface of a monster frog rendered using just lighting and geometry(left). The surface split into several charts (top middle) and its corresponding parameterization into 2D texture space (bottom middle). The surface rendered with texture mapping (top right), and the corresponding texture image (bottom right).

to make the surface appear more realistic and provide details beyond the resolution of the geometry of the shape.

These annotations are usually performed via texture mapping, which maps 2D data from images onto the surface of the 3D object. The 2D data encodes various forms of information used to improve the rendering of the 3D surface. Figure 1.1 shows an example of a monster frog surface rendered with additional detail textures. However, to perform texture mapping and add this detail, the surface must first be parameterized into 2D texture space. Given a parameterization, texture mapping is commonly used to generate more realistic renderings of objects and is not limited to purely textures. In Figure 1.2 the monster frog is rendered with a bump map

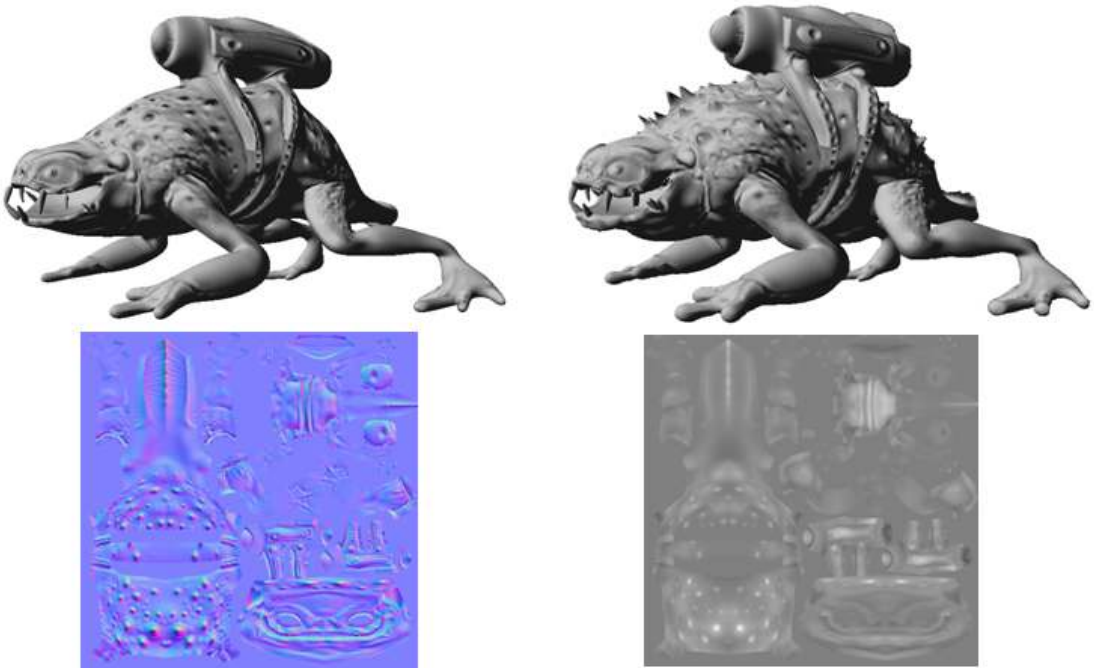


Figure 1.2: Monster frog surface rendered using bump map with bump map image (left), and normal map with bump map image (right).

(left) to simulate normal information for lighting. On the right side of the image the monster frog's geometry is enhanced with a displacement map moving vertices along the normal direction adding a significant amount of detail to the surface. In this dissertation, I will concentrate primarily on the texture mapping application where the desired goal is to produce surface parameterizations, to produce a mapping function from 2D image space onto the 3D surface.

1.2 Parameterization

In computer graphics, the idea of producing mappings between surfaces was originally introduced by Catmull [7] and is now a standard technique for providing additional information to 3D surfaces via 2D textured images by building a cor-

respondence between the 3D surface and 2D image space. The process of surface parameterization takes a 3D surface and effectively flattens it into 2D and produces such a mapping. First, given a 3D surface S, the shape is partitioned along a connected set of edges, referred to as seams, into contiguous sets of polygons called charts. Figure 1.1 shows a partitioning of the monster frog on the left into a set of charts represented as different colors in the top middle image. Parameterization flattens the charts to the two-dimensional domain and the seams of the charts in 3D become the boundaries of the flattened charts in 2D as shown in the bottom middle image. The flattening now provides a mapping from the $\mathbb{R}^2$ parameterization to the $\mathbb{R}^3$ surface.

For surfaces other than developable surfaces, this flattening introduces distortion into the shape and most parameterization methods are concerned with reducing this distortion be it in terms of deviation of angles, area, or some combination thereof. To measure the distortion of a parameterization, generally some form of error metric is measured between the 3D surface and the 2D image mapping. Conformal distortion is the measure of the deviation of angles between the 2D triangles to the 3D triangles. An equiareal distortion measures the deviation between the areas of the 2D triangles to 3D. If the triangle areas and angles match the parameterization is said to be isometric. So, the isometric error measures the deviation of both the angles and areas of the triangles.

While the quality of the parameterization and reducing distortion are certainly important, the parameterization is of limited use for texture mapping and other applications unless it forms a one-to-one mapping or a bijective map between the 3D surface and the 2D texture covered by the charts. Even if there is no distortion in a parameterization, bijectivity may not be maintained. If the parameterization is not bijective, then a single point in the texture could map to multiple, disconnected

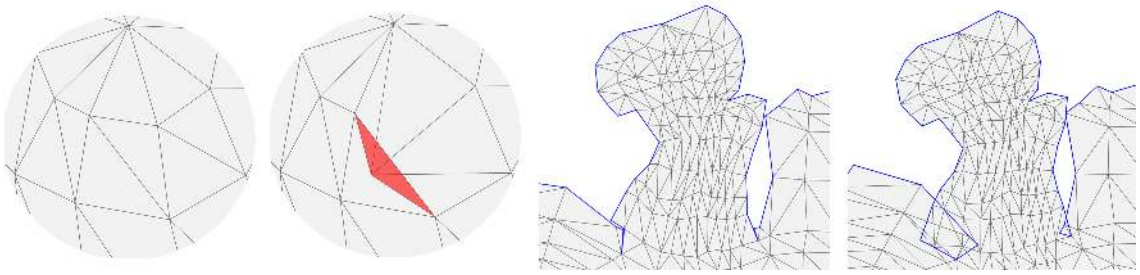


Figure 1.3: From left to right: an image of a locally injective parameterization, an example of a locally folded triangle highlighted in red, an image of a bijective parameterization, an example of a non bijective parameterization where the boundary in blue intersects itself.

regions of the surface. The result is that we cannot annotate such regions of the surface independently from one another causing the mapping of multiple positions on the surface.

There are two primary ways in which the parameterization could fail to be bijective. First there is the problem of local injectivity, where a region of the surface can "fold" in the parameterization. A folded region of the parameterization causes neighboring portions of the surface to map to the same parameterization space. Figure 1.3 shows the problem of local injectivity where a single triangle in parameterization space in the left image has flipped its orientation and locally folded onto other triangles. The folded triangle is shown in red, and has caused neighboring triangles to map to the same texture space.

Second, a parameterization could also violate the one-to-one mapping in a global fashion if separate portions of the parameterization overlap. This problem is seen if the boundaries of charts intersect themselves causing separate sections of the parameterization to overlap. The problem of bijectivity is shown in Figure 1.3 on the right side the parameterization has globally intersected itself where the boundary

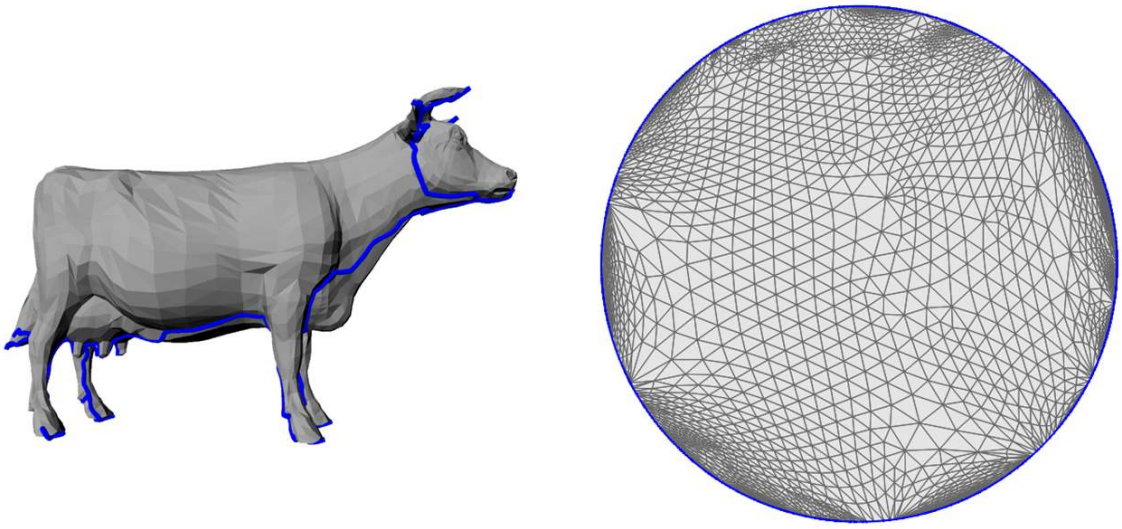


Figure 1.4: Cow mesh with seam in blue (left) with its corresponding Tutte's embedding where the boundary of the parameterization in blue is constrained to a circle(right)

of the parameterization in blue has intersections. This again, causes separate sections of the parameterization to overlap in texture space invalidating the one-to-one mapping.

While only a handful of previous methods guarantee that the parameterization will be locally injective, almost none guarantee bijectivity. If bijectivity is guaranteed in these methods the boundary or seams of the surface are either constrained in parameterization space to a non-intersecting shape, or the seams are modified or increased in some fashion to insure no global folds. However, constraining the boundary, either by user intervention or by choosing some arbitrary non-intersecting boundary curve, will produce more distortion in the parameterization than necessary since the optimization cannot modify the boundary to reduce the distortion of the parameterization. The effect of constraining the boundary is shown in Figure 1.4 where the cow mesh is parameterized using a Tutte's embedding [54] and the overall

distortion of the parameterization is poor due to the boundary constraint.

Instead of constraining the boundary of a parameterization to ensure bijectivity, the other common practice is to modify the seams of the 3D surface, either changing them or adding seams and charts. However, the seams of a surface tend to come with their own issues outside of the parameterization problem. Often, they require special consideration when attempting to use them for texture mapping. Either the seams must be hidden in low visibility areas, or special filters and texture images must be used to reduce the effect of discontinuities along the parameterization seams. The seams of a surface are generally artist generated, however there has been a large amount of research performed to automate and ensure that a better surface partitioning is performed that can be used for various applications including texture mapping. Even though the two problems of parameterization and seam generation can be performed together, this dissertation believes it is important when performing the parameterization, to preserve an artist or original input seam on the surface.

1.3 Data Representation

While there are a number of ways of representing a surface in graphics applications, in this dissertation I will restrict the discussion to discretized surface meshes consisting of triangles. Specifically, a triangle mesh defined as $M = (P, U, E)$, where $P = \{P_1, P_2, \ldots, P_n\}$ is the set of n vertex positions in 3D space, $U = \{U_1, U_2, \ldots, U_n\}$ is the set of 2D texture space coordinates. Vertex positions have corresponding 3D coordinates $P_i = \{x_i, y_i, z_i\}$ and 2D parameterized coordinates $U_i = \{u_i, v_i\}$. E is the set of edges where an edge $E_{ij} = \{P_i, P_j\}$ connects the vertices $\{P_i, P_j\}$. Figure 1.4 shows an example of a triangulated surface. The seams of a mesh shown in blue are defined as a sub set of edges $B \subset E$ which will form the boundary of the parameterization in 2D. I will show results of the parameterization

applications on general meshes found on the internet, as well as synthetic meshes to demonstrate the effectiveness of the algorithms. A simply connected domain $\Omega \subset \mathbb{R}^2$ in parametric space. Let f be a function or parameterization of surface $S \subset \mathbb{R}^3$ over Ω. If f uniquely maps points from Ω to unique points in $\mathbb{R}^3$, then f is a bijective map forming a non unique one-to-one mapping between $\mathbb{R}^2$ and $\mathbb{R}^3$.

1.4 Dissertation Overview

This dissertation introduces a new method of parameterization which guarantees bijectivity and allows for the optimization of boundary vertices to reduce distortion in the resulting parameterization. The main contributions of this thesis are

- The introduction of a generalized parameterization framework capable of minimizing the distortion of a parameterization given a distortion energy metric.
- A new barrier energy metric designed to reduce distortion and ensure local injectivity during the parameterization optimization.
- The introduction of a singularity aware interior point optimization, that computes possible singularities during a line search to ensure invalid parameterizations are not possible.
- A bijective barrier energy term, that ensures the boundary of the parameterization will not intersect itself during the parameterization optimization.
- An extension to the energy function to allow for a weighting of the parameterization energy to optimize the amount of texture space given to important regions of the 3D surface.

2. RELATED WORK

Surface parameterization is a well-studied problem with many surveys summarizing its advances. The survey by Floater et al. [13] summarizes much of the early progression before 2001. Floater and Hormann [14] summarize many of the recent advances since their earlier publication. The survey by Sheffer et al. [48] analyzes many of the current techniques at the time as well as applications, open problems, and pre-processing steps such as chart generation. Hormann et al. [20] gave a thorough course at SIGGRAPH analyzing many aspects of parameterization and many of the core concepts. A common and important theme among these surveys and course is that there are very few parameterization methods that can guarantee bijectivity. If bijectivity is guaranteed, the methods require that the boundary of the parameterization either be constrained to a specified shape, or that the initial boundary be modified in some fashion.

2.1 Seam Creating Parameterization

One class of methods interleaves the mesh segmentation process of dividing the surface into a set of charts with parameterization [26, 63]. If the parameterization is not bijective, these methods split the charts to form smaller charts. Levy et al. [26] split charts based on boundary intersections, while Zhou et al. [63] split charts based on a stretch based distortion metric. Such a splitting process continues until all charts form bijective maps. This process is guaranteed to stop since individual polygons can trivially form bijective maps, which reduces to per polygon texture mapping [4, 57]. Sander et al [38] merge charts of single faces into larger chart regions and perform parameterizations of charts to allow for surface and parameterization simplification, but require a constrained boundaries. Sorkine et al. [51] take a region growing

approach to chart creation where they detect if adding a new triangle to a chart will cause an intersection in the boundary and modify the seam. Expanding upon this idea Singh et al.[49] grow regions of triangles in a greedy fashion to allow for rapid texturing by taking advantage of salient features, however produces highly visible seams.

Springborn et al. [52] use a discrete conformal optimization for parameterization and require no initially defined seam. Their method guarantees local injectivity by changing the connectivity of the mesh performing edge flips or subdividing edges. While this parameterization approach produces valid parameterization, this dissertation is interested in taking into account the initially prescribed seam to allow for more artist/application control of the resulting parameterization. Zhang et al. [61] build charts of a surface based off of a protrusion analysis. This paper actually guarantees bijective charts using "scaffold triangles" in a nonlinear minimization to guarantee the line search never causes the boundary to intersect. While similar in spirit to this dissertation, the scaffold criteria, while a sufficient condition to create bijective parameterizations, is not necessary and slows down the optimization restricting the possible solution space more than necessary. In contrast to these works, this dissertation will take as input initially determined seams. Allowing users to input the specific chart boundary and requiring that the boundary is constrained, gives more control over the charts' shapes.

2.2 Constrained Boundary Parameterizations

Another class of parameterization methods takes as input an initial seam and attempts to flatten the 3D surface into parameterization space. The seams or boundaries of the charts in this class are constrained in some way or user prescribed to some desired shape. While this restricts the overall solution space, often these methods

can guarantee bijectivity. Some of these parameterization methods guarantee local injectivity through the use of distortion metrics that form barriers so that triangle flips cannot occur. More detail is given about barrier energy metrics in Section 3.2. Methods such as [28, 2, 1, 33, 36] bound the distortion of triangles to guarantee locally injective parameterizations. In addition, all of these methods can guarantee a bijective map if the user constrains the boundary of the charts to form a non-intersecting curve, or performs a post processing of the parameterization to remove boundary intersections. Lipman [28] and Bommes et al. [2] convexify the solution space of their non-convex optimization problem to find injective solutions. Schuller et al. [43] and Schneider et al. [41] use a barrier energy to deform an initial bijective parameterization to match a target user prescribed boundary while maintaining injectivity. Aigerman et al. [1] produce lifted bijections between two surfaces relying on user defined correspondences, and can fail to produce a bijection if poor correspondences are provided. Poranne and Lipman [33] build injective parameterizations for deformation applications where soft and hard boundary constraints are introduced by a user, however bijective maps are not a concern of the paper. Later, I will discuss the idea of barrier distortion metrics in Section 3.2 and show that while injectivity is guaranteed, there are large difference in computational efficiency between various methods.

While such locally injective methods typically involve non-linear optimization, some methods guarantee bijective maps by solving a linear set of equations if the boundary of the charts are constrained to convex shapes such as circles [54, 12, 18]. The first discrete version of harmonic maps, based on the Dirichlet energy was proposed in Eck et al. [11] to produce a linear solution for parameterization embeddings. Gu and Yau [17] approximate the Laplace Beltrami operator using a discretized version of the Dirichlet energy to produce conformal parameterizations over a unit 2D

square using a method similar in nature to the Floater's embedding[12]. These approaches can guarantee bijections, however, as shown in Figure 1.4, the distortion for these parameterizations can be severe. In the figure, the boundary of the parameterization is constrained to a circle, and the interior vertices are embedded producing a poor triangulation of the domain. Sheffer and Sturler [44] use a different strategy to parameterization by adding triangles one at a time to the parameterization by constraining previous triangles in place and solving for the angles needed to place new triangles. The method does not produce bijective maps, but includes a post processing step of further constraining boundary edges to correct their intersections. Weber et al. [55] showed how to create a bijective map between two non-intersecting boundaries by mapping to a common, convex domain. Unfortunately, this method requires that the user specify a non-intersecting boundary curve and may still require additional post processing to refine some triangles to guarantee a bijective map.

2.3 Free Boundary Parameterization

A third class or parameterization methods also takes as input an initial seam, but instead of constraining it in parameterization space, the boundary is allowed to freely move during the parameterization process. This type of method produces lower distortion parameterization, they often have issues when producing bijective maps. The MIPS [19] and AMIPS [16] parameterization are nonlinear methods that aims to reduce angular distortion. However, the energy functions used by these methods are nonlinear and non-convex. Labsik et al. [25] accelerate the MIPS algorithm to use a multiresolution parameterization and apply MIPS to surface remeshing applications. Desbrun et al. [10] used a discretization of the Dirichlet energy derived in Pinkall and Polthier [32] to construct harmonic maps that do not require boundary constraints. This method does not guarantee local injectivity in the presence of very

obtuse angles. As-rigid-as-possible (ARAP) and as-similar-as-possible (ASAP) parameterizations [50, 29] measure the deviation of the Jacobian from being a rotation matrix. The energy minimized is non-convex and finding a global minimum does not guarantee local injectivity. Chen et al. [8] use local coordinate frames and local geometry to flatten the 3D surface into the plane, but the method can cause locally folded triangles. Kharevych et al. [24] build parameterizations based on circle patterns that produces natural looking boundaries and guarantees injectivity, but bijectivity is not guaranteed without user intervention. Zayer et al. [59] perform several linear optimizations to produce conformal free boundary parameterizations, however the method has no guarantee of bijectivity and in poor starting positions has problems preserving triangle areas.

The computationally intensive angle based flattening (ABF) [45] directly solves for angles of the parameterized triangles and then finds an embedding compatible with those angles. The authors attempt to create a bijective map by performing a local post-processing step to the angles after the initial parameterization though such a procedure can significantly increase distortion. Liesen et al. [27] and Zayer et al.[60] discuss methods for speeding up ABF, but still take a considerable amount of time for even medium size meshes with 10 thousand polygons. Later ABF++ [47] created a much faster form of angle base flattening using a hierarchical optimization procedure that guarantees local injectivity but fails to produce bijective parameterizations. Continuing to improve upon the efficiency of ABF is work by Zayer et al. [58] who reformulate the non linear constraints of ABF into linear approximations. However, this approximation opens up the possibility for the parameterization to not be injective. Another method similar to ABF by Cartade et al. [6] uses an alternating optimization to optimize the boundary and interior of the parameterization to generate "natural" boundaries, but has no specific guarantees of injectivity.

3. BIJECTIVE MAPS*

The primary goal of this dissertation is to produce a bijective map from a 3D triangulated surface to a 2D domain that minimizes a distortion metric without modifying or constraining the boundary. We begin with a 3D triangulated surface that is partitioned into a set of charts. For the moment, I restrict the discussion to a single chart consisting of a set of triangles that are topologically equivalent to a disk. A map for such a triangulated surface is bijective if two properties hold. First, the mapping must be locally injective, where no triangles reverse orientation in the parameterization. The second property is that no separate pieces of the parameterization overlap. Overlap is detected if the boundary of the parameterization intersects itself.

3.1 Distortion

The majority of parameterization methods minimize some form of distortion metric between the 3D surface and its corresponding 2D parameterization. To build a distortion metric, look at an infinitesimally small displacement $(\hat{u}, \hat{v})$ from a position (u, v) in parameter space. The new surface point can be approximately represented through a first order Taylor expansion $\tilde{f}$ as,

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

This function $\tilde{f}$ maps the surrounding parameterization around (u, v) into the tangent plane at $f(u, v)$. This same function will map infinitesimally small circles

*Part of this chapter is reprinted with permission from "Bijective Parameterization with free boundaries" by Jason Smith and Scott Schaefer, 2015. ACM Transactions on Graphics, 34, 70:1–70:9, Copyright 2015 by the Association of Computing Machinery

around (u, v) into ellipses around the corresponding position in 3D at $f(u, v)$. This deformed ellipsoid can be thought of as a representation of the distortion in parameterization space. The distortion can be represented through a rewriting of the previous equation

$$\tilde{f}(u + \hat{u}, v + \hat{v}) = f(u, v) + J(u, v) \begin{pmatrix} \hat{u} \\ \hat{v} \end{pmatrix}.$$

The Jacobian J of f is the partial derivatives of f as column vectors. Using a singular value decomposition of the Jacobian we generate J as a decomposition into three matrices with the form,

$$J = X\Sigma Y^T = X \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{pmatrix} Y^T$$

Where, matrices X, and Y^T are orthogonal matrices representing rotations. Since X and Y are simply rotations, there is no distortion present in these matrices. Also, we obtain two singular values $\{\sigma_1, \sigma_2\}$ in the middle matrix corresponding directly to the stretch of the ellipsoid in the tangent plane. Now $\{\sigma_1, \sigma_2\}$ can be used to measure the distortion of the parameterization at specific positions.

However, in the setting of a discrete triangulated surface, the distortion of individual triangles is of interest. Consider a 3D triangle shown in Figure 3.1 (left) with vertices $P_1, P_2, P_3 \in \mathbb{R}^3$. This triangle is isometrically flattened to 2D with zero distortion using a rigid transformation R_t (top). Producing the isometrically flattened triangle $F_1, F_2, F_3 \in \mathbb{R}^2$ is a simple operation where the vertex P_1 is mapped to the origin of the 2D plane. Vertex P_2 is positioned along the x-axis as to match the length of the 3D edge $\bar{P_1P_2}$. The final vertex P_3 is positioned in such a way as

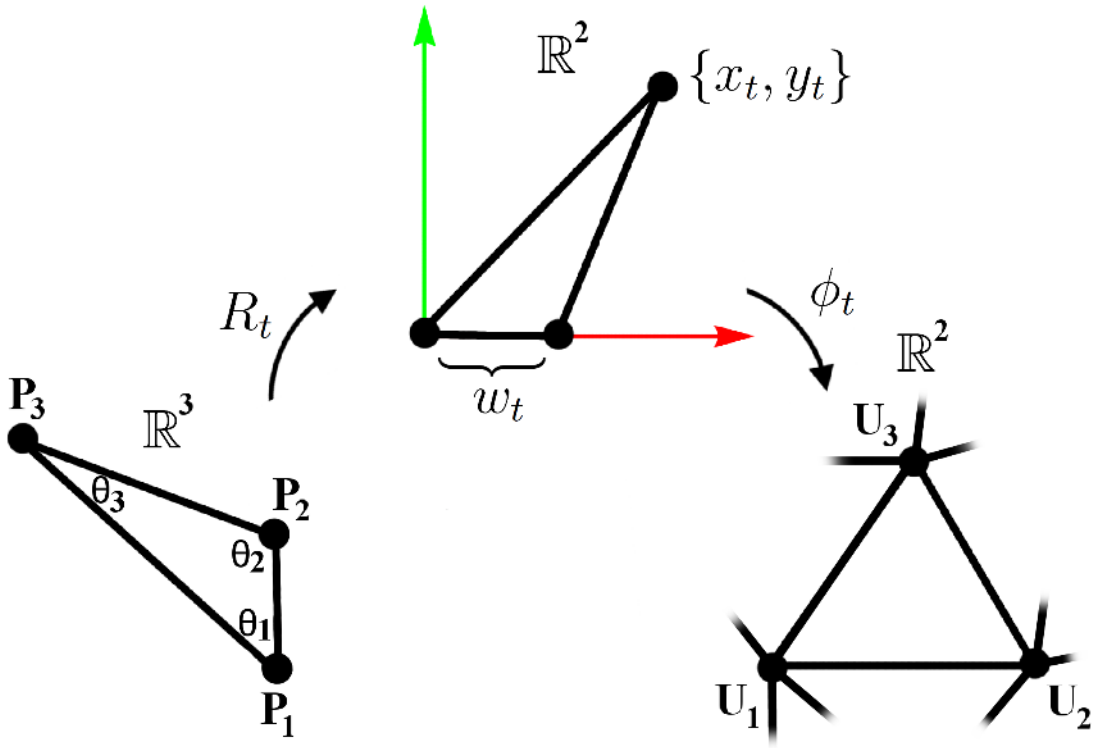


Figure 3.1: Mapping a 3D triangle to 2D using a rigid transform R, which is then affinely mapped via ϕ to the parameterization.

to maintain the angles and edge lengths of the 3D triangle. The formatting of the isometric triangle is

$$F_1 = \{0, 0\}, F_2 = \{w_t, 0\}, F_3 = \{x_t, y_t\},$$

This isometric triangle is then mapped to the parameterized shape with vertices $U_1, U_2, U_3 \in \mathbb{R}^2$ via the affine transformation ϕ_t (right). This affine transformation contains the distortion of a single triangle when transformed into 2D parameterization space. Now, similar to the surface distortion measurement we want to find a way to quantify distortion. For a single triangle, the singular values of the 2x2 linear

portion L_t of the affine transformation $\phi_t = \begin{pmatrix} L_t & m_t \\ 0 & 1 \end{pmatrix}$ define the distortion of the triangle in the parameterization. Here, m_t represents a simple translation and does not contribute to the distortion.

The common approach to representing the affine transformation ϕ_t is for a triangle t with vertices P_1, P_2, P_3,

$$\phi_t = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} \begin{pmatrix} 0 & w_t & x_t \\ 0 & 0 & y_t \\ 1 & 1 & 1 \end{pmatrix}^{-1}$$

However, we can write L_t in a simpler fashion. Let the linear portion of ϕ be given by the matrix

$$L_t = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

The singular values of this matrix are

$$\begin{aligned} \sigma_1 &= \tfrac{1}{2}\left(\sqrt{(b+c)^2+(a-d)^2} - \sqrt{(b-c)^2+(a+d)^2}\right) \\ \sigma_2 &= \tfrac{1}{2}\left(\sqrt{(b+c)^2+(a-d)^2} + \sqrt{(b-c)^2+(a+d)^2}\right) \end{aligned}$$

where $\sigma_2 > \sigma_1$. When $\sigma_1 = \sigma_2$, the scale is uniform and corresponds to a conformal flattening of the triangle. Furthermore, when $\sigma_1 = \sigma_2 = 1$, the flattening is isometric. One more important property is found from the singular values when $\sigma_1 = 0$ the triangle is degenerate. A triangle is degenerate when the vertices of the parameterized triangle are collinear.

Most common parameterization methods measure distortion using a function of these singular values. For example, a linear form of conformal energy is used

in least squares conformal maps (LSCM) [26] given by $(\sigma_1 - \sigma_2)^2$, which tends to shrink the chart since minimizing the scale also minimizes the distortion energy. As-rigid-as-possible parameterization [29] measures isometric error by minimizing $(\sigma_1 - 1)^2 + (\sigma_2 - 1)^2$. However, both of these functions have the problem that they allow triangles to reverse orientation since the energy is finite for $\sigma_1 = 0$.

3.2 Local Injectivity

To begin this section, an introduction into barrier functions is necessary. Barrier functions are commonly used to force a minimization to push away from undesirable solutions. A barrier will cause a functions value to smoothly but rapidly approach ∞. For example, in parameterization, to produce locally injective maps, triangles cannot change orientation. If a triangle does flip its orientation, this would break the local injectivity property. Therefore a type of barrier function is necessary in combination with the distortion energy. A simple way of preventing triangles from flipping orientation is to require a distortion measurement per triangle E_t that approaches ∞ as σ_1 approaches 0 given that this is when the triangle would flip orientation. Since σ_1 is the smaller of the 2 singular values, functions inversely proportional to σ_1 approach ∞ as σ_1 approaches zero.

Several previously used distortion functions already satisfy this property including conformal [9] $\left(\frac{\sigma_2}{\sigma_1}\right)$, MIPS [19] $\left(\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1}{\sigma_2}\right)$, maximal isometric distortion [51] $\left(\max(\sigma_2, \frac{1}{\sigma_1})\right)$, and a form of isometric energy [1] $\sqrt{\sigma_2^2 + \sigma_1^{-2}}$. Note, that all of these functions contain a term of $\frac{1}{\sigma_1}$. To compute the total distortion of these measurements, the functions must be integrated over the 3D surface. Since the singular values are constant per triangle, the total distortion is simply the sum of the distortion evaluated at each triangle weighted by the area of the 3D triangle.

While all of these distortion measurements can be used to ensure local injectivity,

their computational efficiency can vary significantly. For example, MIPS computes a quantity similar to conformal energy but yields a very simple expression and is much faster to optimize. In the spirit of computational efficiency, this dissertation proposes a new form of isometric distortion energy.

$$E_t = \sigma_1^2 + \sigma_2^2 + \sigma_1^{-2} + \sigma_2^{-2}, \tag{3.1}$$

which simplifies to

$$E_t = \left(1 + \sigma_1^{-2}\sigma_2^{-2}\right)\left(\sigma_1^2 + \sigma_2^2\right). \tag{3.2}$$

The distortion metric in Equation 3.2 has the property that its minimum is achieved when $\sigma_1 = \sigma_2 = 1$. Moreover, $\sigma_1^2\sigma_2^2$ has an extremely simple expression given as the ratio of the squared area of the parameterized triangle Δ_U^2 to the squared area of the 3D triangle Δ_P^2. In addition, $\sigma_1^2 + \sigma_2^2$ is the Dirichlet energy and is a quadratic function of the texture coordinates given by

$$\begin{aligned}\sigma_1^2 + \sigma_2^2 = \quad & \tfrac{|U_3-U_1|^2|P_2-P_1|^2+|U_2-U_1|^2|P_3-P_1|^2}{4\Delta_P^2} \\ & - \tfrac{((U_3-U_1)\cdot(U_2-U_1))((P_3-P_1)\cdot(P_2-P_1))}{2\Delta_P^2}.\end{aligned}$$

Integrating this energy over the 3D triangle gives the distortion of that triangle as

$$\begin{aligned}E_t = \left(1 + \tfrac{\Delta_P^2}{\Delta_U^2}\right)\Bigg(& \tfrac{|U_3-U_1|^2|P_2-P_1|^2+|U_2-U_1|^2|P_3-P_1|^2}{4\Delta_P} \\ & - \tfrac{((U_3-U_1)\cdot(U_2-U_1))((P_3-P_1)\cdot(P_2-P_1))}{2\Delta_P}\Bigg).\end{aligned} \tag{3.3}$$

To get the distortion error for the entire mesh, E_t is summed for every triangle and multiplied by the area of its 3D triangle $E_D = \sum_{t\in T} \Delta_P E_t$.

Most optimization methods also require the gradient of the error function, which

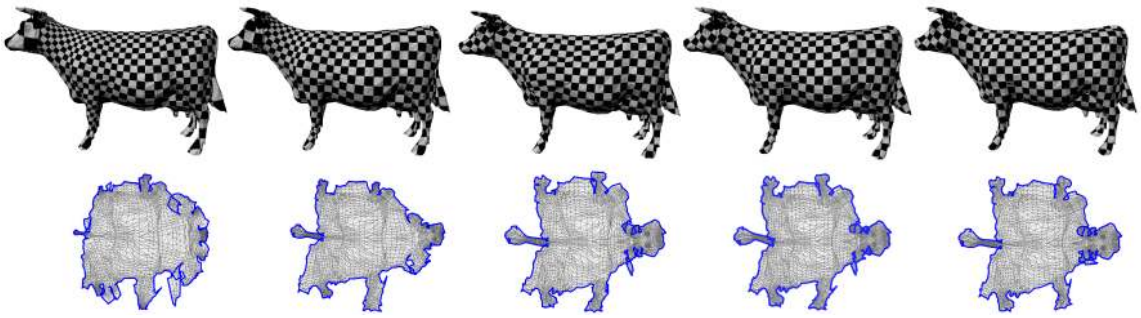


Figure 3.2: Injective optimization of E_D using different metrics to produce a locally injective parameterization. The top row shows a checkerboard mapped to the surface via the parameterization shown below. From left to right the metric used with timings in seconds: conformal 95.97, MIPS 3.42, maximal isometric 114.46, isometric 125.62, ours 1.29.

is also easy to calculate for Equation 3.3. Without loss of generality, we only consider the partial derivative of Equation 3.3 with respect to a single vertex U_1. Given that Equation 3.3 is the product of two terms, its derivative is given by the product rule and only the derivatives of each of the terms in the product need to be considered. The partial derivative with respect to U_1 of the first term of Equation 3.3 corresponding to $\left(1 + \Delta_P^2 \Delta_U^{-2}\right)$ is simple and is given by

$$-\frac{\Delta_P^2}{\Delta_U^3}(U_2 - U_3)^\perp$$

where $(U_2 - U_3)^\perp$ indicates a rotation of $(U_2 - U_3)$ by 90 degrees in the plane. Taking the partial derivative of the second term of Equation 3.3 yields the cotangent weights [32] that correspond to a discrete harmonic function where θ_i corresponds to angles of the 3D triangle shown in Figure 3.1

$$-\cot(\theta_2)U_3 - \cot(\theta_3)U_2 + (\cot(\theta_2) + \cot(\theta_3))U_1.$$

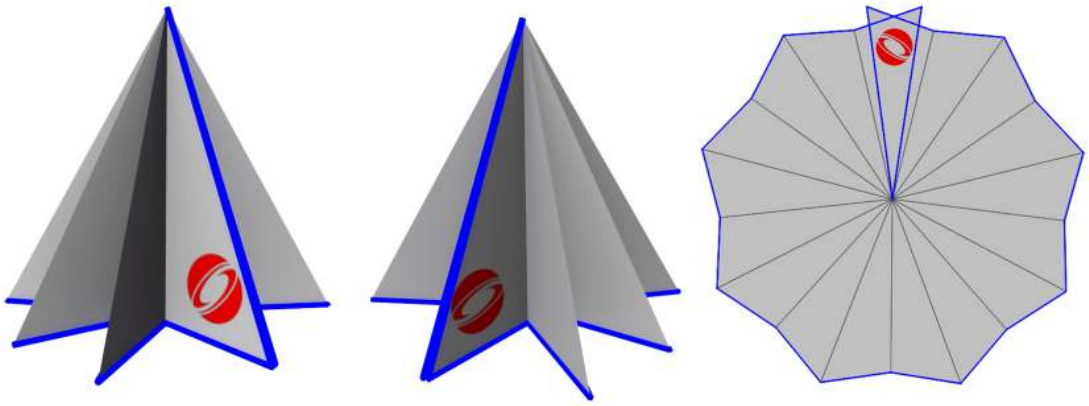


Figure 3.3: A simple example of a wavy cone with the seam shown in blue (left and middle) isometrically flattened to a parameterization without folded triangles (right) but still does not form a bijective map. A SIGGRAPH symbol has been added to the texture space of the folded region to show its effect of texture mapping.

Moreover, quantities involving the P_i are constant in both the gradient and distortion metric in Equation 3.3 and can be precomputed. Figure 3.2 shows an example of using the optimization from Section 3.4.1 on a cow model composed of a single chart with different metrics that all produce locally injective parameterizations.

Unfortunately, such local injectivity does not guarantee a bijective map. For example, Figure 3.3 shows an example of a cone that is flattened without any isometric distortion. No triangles reverse orientation in the parameterization. However, the surface folds on itself. The second property needed to construct a bijective map is that the boundary of the parameterized chart does not intersect itself. The importance of bijectivity is shown in the Figure, where the SIGGRAPH symbol is added to a single flap of the cone model. However, because of the folded texture space, the SIGGRAPH symbol appears on both sides of the flap. There is no way of modifying this region of the surface's texture without modifying the other using texture

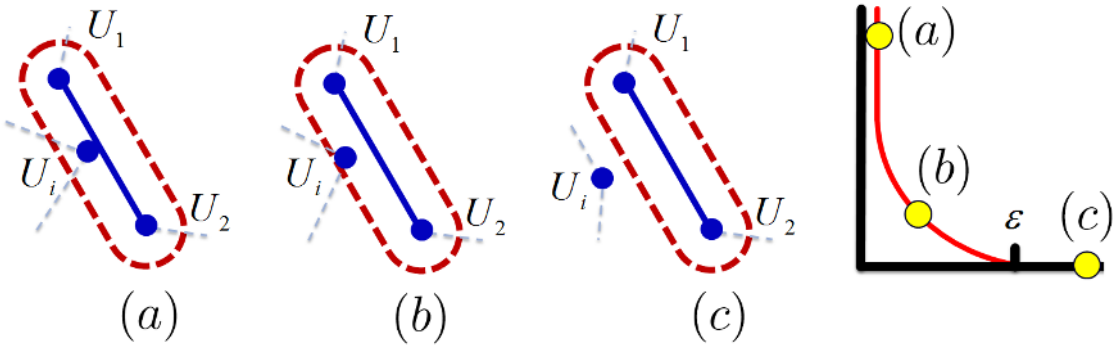


Figure 3.4: Three examples of configurations demonstrating the boundary barrier function E_D.

mapping. The bijectivity property is much more difficult to maintain as it is not a quantity that can be locally computed.

3.3 Bijectivity

While the admissible metrics in Section 3.2 will produce locally injective parameterizations, the boundaries of these parameterizations may intersect themselves meaning that they do not form a bijection. It is possible to create a bijection by constraining the boundaries of the shape [54, 12, 28, 41, 43, 33, 55]. However, doing so typically requires the user to specify the boundary of the chart independent of the distortion metric, which leads to greater distortion in the parameterization. Yet generating an intersection free boundary while minimizing E_D is a difficult computational problem because of the global nature of the boundary; that is, unlike Section 3.2, the criteria to prevent intersections is not simply a local property of a triangle.

Keeping in mind the global nature of the problem, a similar approach as the one to guarantee injectivity is taken to ensure the parameterization forms a bijective

map. Again, to make sure bijectivity is upheld, the boundaries of the parameterization must not intersect themselves. To enforce intersection free boundaries we must formulate a barrier function for the boundary that approaches ∞ as the boundary approaches an intersecting configuration. For each boundary edge with vertices U_1, U_2, we associate a barrier function

$$\max(0, \frac{\epsilon}{dist(U_1, U_2, U_i)} - 1)^2$$

where $dist(U_1, U_2, U_i)$ measures the distance from a boundary point $U_{i\neq 1,2}$ to the edge (U_1, U_2). Figure 3.4 demonstrates, this function as it shows three configurations of boundary vertices and edges and their function value. The function (a) is 0 for $dist(U_1, U_2, U_i) > \epsilon$. When $dist(U_1, U_2, U_i) = \epsilon$ the function and gradient go to 0 allowing for a smooth function when $dist(U_1, U_2, U_i) < \epsilon$ (b) transitioning from (a),and the function approaches ∞ as $dist(U_1, U_2, U_i)$ approaches 0 (c). As ϵ shrinks, so too will the gaps between the boundary curves. Because of this, ϵ can have a significant effect on the converged parameterizations and is very dependent on the scale. In practice, ϵ is chosen to be the average length of the chart's 3D seam edges divided by 4. Figure 3.5 demonstrates the effect of changing ϵ on the resulting parameterization. On the left we show the default value of ϵ used for the results in this dissertation, then moving to the right ϵ increases to show the overall effect of the boundary barrier E_B. While, smaller increases add some extra spacing between the boundary, values of ϵ larger than the size of parameterization's features have significant effect on the converged result as the boundaries begin to influence more and more pieces of the boundary. Our total boundary energy E_B is then given by summing, for each boundary edge, this function evaluated over all boundary vertices not part of this edge.

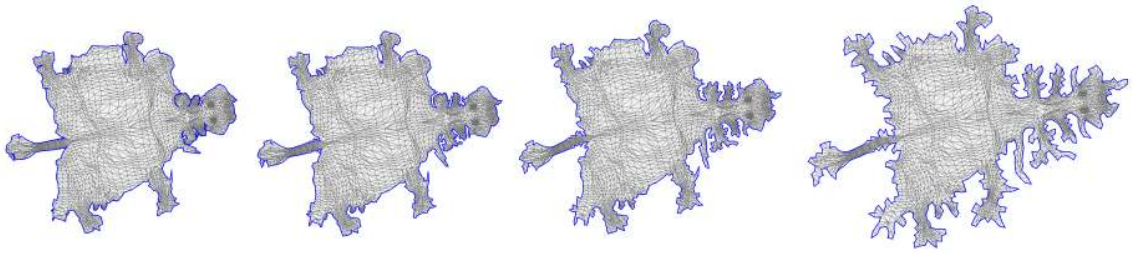


Figure 3.5: Examples of modifying ϵ. From left to right: $\epsilon = .1$, $\epsilon = 2.5$, $\epsilon = 5$, $\epsilon = 10$.

This choice of barrier function creates a number of advantages. First, it is computationally efficient given its local support as an offset of size ϵ from the current boundary of the chart. Typical barrier functions use $-\log(dist(U_1, U_2, U_i))$ or $\frac{1}{dist(U_1,U_2,U_i)}$ as barriers, but the global support of these functions means that a large number of vertices are affected by each boundary edge and the gradient becomes dense. Second, the smaller support also means that much of the optimization of E_D is unaffected by the boundary whereas edges in a globally supported function would cause some distortion in E_D for even far away vertices. Finally, though the function is locally supported, it is a smooth function to avoid discontinuous changes in the gradient during optimization.

The local support of our barrier function allows for significant acceleration of the computation of E_B. Instead of checking every boundary edge to every boundary vertex, the left side of Figure 3.6 shows a spatial hashing technique allowing for the evaluation of a small subset of boundary edges and vertices. Before evaluating E_B, a grid (black) is constructed ϵ larger than the current bounding box of the chart. For each function evaluation, first, insert boundary vertices (blue) into the grid. After all boundary vertices are placed in the grid, each boundary edge, queries all grid cells within the bounding box (red)of the edge enlarged by ϵ (dark red) disregarding

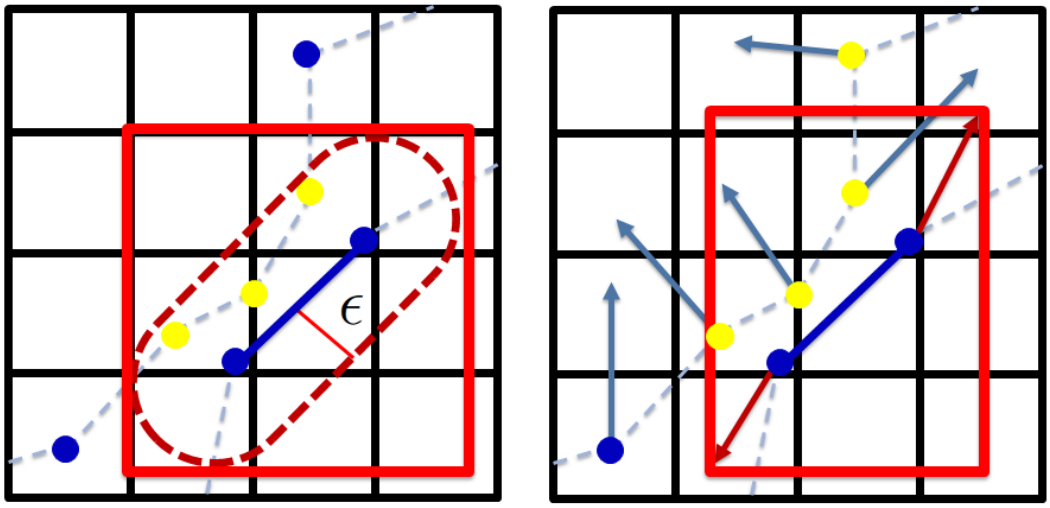


Figure 3.6: Spatial Hash example for the function and gradient evaluation (left) and the maximum step size computation (right). Blue vertices and line segments correspond to boundary edges and vertices. The red box is the size of the bounding box query with yellow points representing vertices possibly contributing to the evaluation and step size computation.

vertices that are part of this edge. Remember that, at a distance of ϵ, the function is 0 meaning that, outside of the bounding box, E_B will have no influence. The bounding box queries return a set of vertices (yellow) which are the only vertices that need to be evaluated with respect to the current edge. Such a simple change greatly enhances the speed of evaluating E_B as well as its gradient by resulting in a 85% reduction in overall timings of the total optimization time.

Figure 3.7 shows an example of optimizing the same nonlinear metrics from Figure 3.2 with the addition of E_B. The optimization times almost uniformly increase due to the extra computations involved, although it is possible to reduce optimization times if E_B causes the optimization to terminate early as was the case for the conformal metric. However, each of these parameterizations now forms a bijective map. In addition, the parameterizations appear similar to their unconstrained coun-

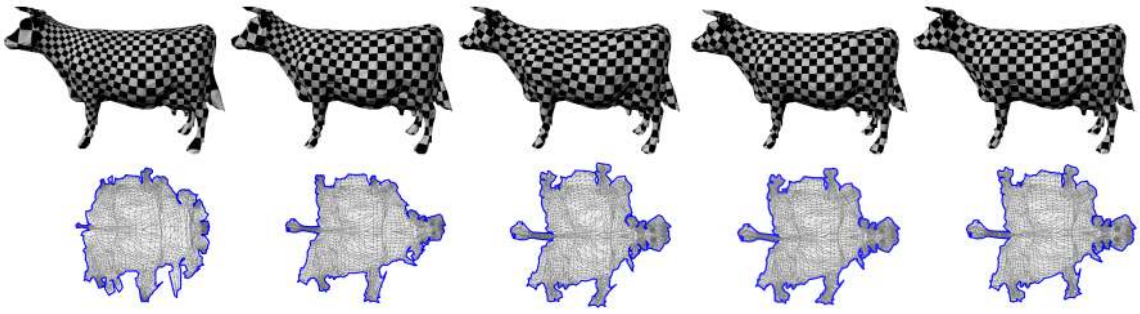


Figure 3.7: Bijective optimization of $E_D + E_B$ using different metrics to produce a bijective parameterization. The top row shows a checkerboard mapped to the surface via the parameterization shown below. From left to right the metric used with timings in seconds: conformal 91.23, MIPS 19.5, maximal isometric 230.36, isometric 136.71, ours 3.63.

terparts despite requiring that the parameterizations form a bijective map at every stage of the optimization.

3.4 Optimization

The same distortion metrics that enforce local injectivity in Section 3.2 also yield a difficult optimization problem. There are several common approaches to optimizing parameterizations. A common early tactic was to constrain the vertices of the parameterization and optimize a single vertices at a time [19]. Other methods concentrate on optimizing subsets of vertices [16] allowing for larger sets of vertices to be optimized at once. Other techniques optimize distortion error using random search directions [42]. Lipman [28] bounds distortion using inequality constraints and decompose the problem into maximal convex subsets, although such an approach requires repeated optimization for each subset. A main idea of this dissertation is to remove these restrictions and take a more natural approach to optimize all of the vertices at the same time like shown in Figure 3.8 where a parameterization of a camel's vertices are iteratively repositioned at the same time without the need for

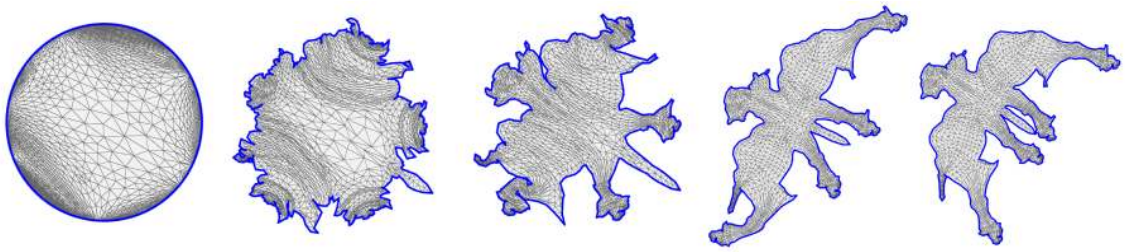


Figure 3.8: Starting from Tutte's parameterization (left), our optimization generates a parameterization that minimizes distortion and guarantees a bijective map (right). We show intermediate stages of the optimization where, at every step, the parameterization is bijective. As opposed to previous techniques, we do not constrain the shape of the boundary, which is free to change shape to minimize distortion.

any constraints.

3.4.1 Interior Point Optimization

While there are many possible approaches to minimizing $E_D + E_B$, the minimization must not break the bijectivity property. An interior point method [15] guarantees that, at every step, the map remains a bijection. Therefore, an initial parameterization is required to guarantee the optimization begins in the valid solution space of a bijective map. Fortunately, Tutte's embeddings [54] and Floater's parameterization [12] both provide valid starting points, although the distortion of the parameterization is likely to be extreme since both methods constrain the boundary to a convex shape such as a circle as shown on the left of Figure 3.8.

So, to minimize the distortion of the parameterization, the vertices need to be moved in a direction which reduces distortion. Figure 3.8 shows an iterative process of minimizing the distortion of a camel model's parameterization with Tutte's method. From left to right, vertices are moved in parameterization space to where each consecutive image reduces distortion. To move the vertices in such a manner, we need a search direction to move the vertices, which could be found from any

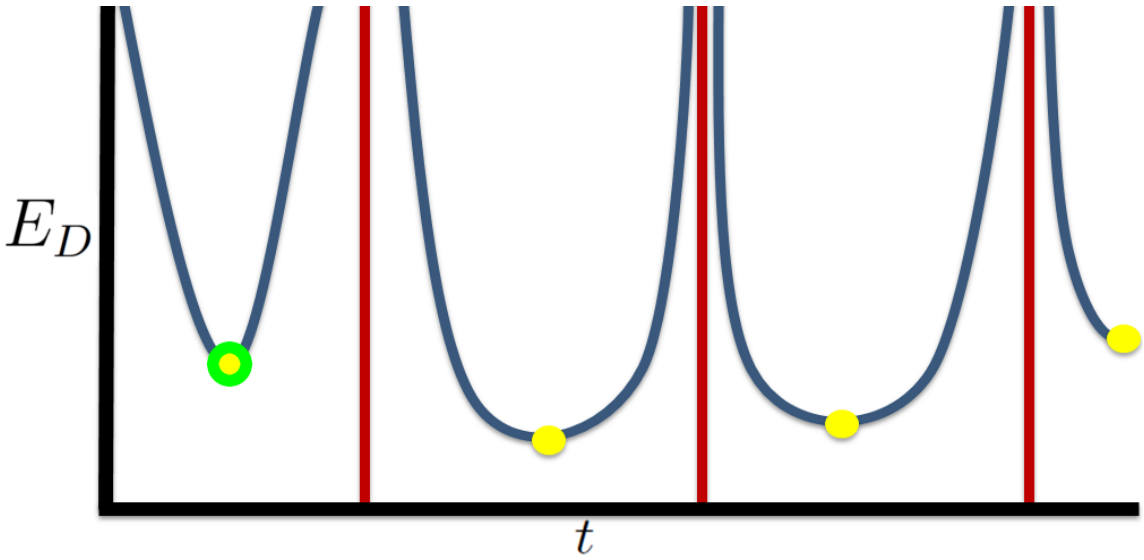


Figure 3.9: A graph of a single step of the optimization. The x-axis measures the magnitude along a search direction t, while the y-axis represents the evaluated function E_D (blue) at some the given search amount t. Singularities (red) representing a triangle flipping its orientation. Local minimum are shown with yellow dots, and the desired minimum is highlighted green.

Quasi-Newton optimization technique.

However, Quasi-Newton approaches begin with the current location of the parameterized vertices U and repeatedly finds a search direction V to search in the given energy function

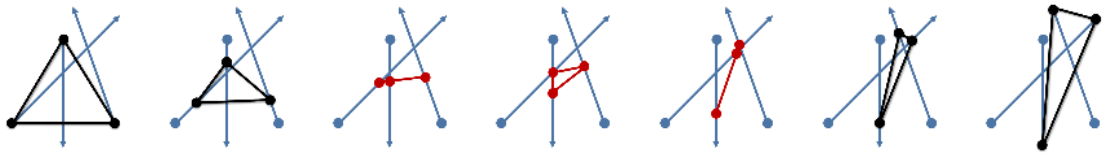
$$\min_{t,t>0} f(U + Vt)$$

Where $f = E_D + E_B$ is the total distortion function. Such methods typically rely on a backtracking line search starting from some maximal parameter t_{max} and decrease t until f is minimized or sufficiently decreases as measured by various criteria such as the Wolfe condition [56].

3.4.2 Singularity Aware Optimization

Being an interior point method, simply using common line searches will not work without modification. In the case of parameterization, interior point methods must consider the singularities of the function to ensure that the method does not leave the valid solution space. Luckily, all of the injective distortion metrics E_D considered above have the same set of singularities, namely when each individual triangle becomes degenerate. This problem can be seen in Figure 3.9, where a single line search is shown with E_D evaluated in blue along the search direction with magnitude t. The problem with taking a step during a line search is that we could accidentally pass one of the multiple singularities in E_D and find any of the local minimum shown in yellow, while the green minimum is the desired step size. This means that the method could step outside of the valid solution space. Normally, given an initial step size, an interior point method must detect that it has left the valid solution space and then undue the step and then shrink step size, repeating this process until finding a minimum. In this work, instead of the time consuming process of detecting invalid states and backtracking the line search, the interior point method will take advantage of the structure of the singularities. For all of the triangles, we explicitly compute the minimum singularity, and use it as the maximum allowed step size to guarantee that we do not cross a singularity.

However, care must be taken when computing the singularities as the number of singularities can be quite large. As shown in Figure 3.10, each triangle can contribute up to two singularities for a single search direction yielding hundreds or even thousands of singularities along a single search direction for even modest sized charts. From left to right, we move a triangle's vertices along its search direction in blue showing two possible triangle flips. The triangle in the Figure is red when



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

Figure 3.10: Visualization of a single triangle (black) during a line search with search directions in (blue). A triangle with a flipped orientation is in red.

the triangle's orientation has been flipped, and even though the triangle corrects its orientation, there is no way to guarantee the immediate neighborhood of triangles has not flipped as well. Consider a non-degenerate 2D triangle with vertices U_1, U_2, U_3 with corresponding search direction vectors V_1, V_2, V_3. The singularities in E_D are given when the triangle becomes degenerate, which is when its signed area becomes zero. To compute when the area becomes zero we can solve for the magnitude t along a search direction using a simple determinant.

$$\det \begin{pmatrix} (U_2 + V_2 t) - (U_1 + V_1 t) \\ (U_3 + V_3 t) - (U_1 + V_1 t) \end{pmatrix} = 0 \tag{3.4}$$

Fortunately, Equation 3.4 is quadratic in t and the parameters that yield a singularity in E_D are simply given by the roots of this quadratic matching the two possible singularities demonstrated by Figure 3.10. We only search along the positive direction and can ignore negative parameters and only need the minimum parameter per triangle. Computing the minimum singularity over all triangle gives the maximum value t_{max} for the line search with respect to E_D.

Now, the singularities of E_B must also be computed with respect to the line search. In this case, singularities are given by intersections of line segments of the boundary versus other boundary vertices. Let U_1, U_2 be boundary vertices that form

an edge of the boundary and U_j be any other boundary vertex with V_1, V_2, V_j be their respective search direction vectors. Luckily, the exact same test in Equation 3.4 where now the virtual triangle formed by vertices U_1, U_2, U_j gives the potential parameters associated with singularities for this combination of edge/vertex corresponding to when the edge and given vertex are all collinear. The new singularity computation is given by,

$$\det \begin{pmatrix} (U_1 + V_1 t) - (U_j + V_j t) \\ (U_2 + V_2 t) - (U_j + V_j t) \end{pmatrix} = 0. \tag{3.5}$$

The only complication is that the roots of the quadratic may not necessarily correspond to singularities. It is possible that, while the vertex and edge are collinear, the vertex lies outside the extents of the edge. However, such a test is trivial, and we discard roots of the quadratic that do not correspond to singularities. For the remaining roots, if any, we use the smallest, positive root. Computing the minimum of this quantity over all combination of boundary edges and boundary vertices along with the parameter t_{max} from E_D gives the maximal possible parameter such that f contains no singularities between $t \in [0, t_{max})$.

The only issue with the computation of the singularities of E_B is how many cases must be checked. If there are m boundary vertices, the complexity of simply choosing the maximal parameter for the line search is $O(m^2)$ as there are $O(m)$ boundary edges, each of which must be compared against $O(m)$ boundary vertices. Fortunately, we can use a similar spatial hashing tactic in Section 3.3 to accelerate this computation. Assume that we have first computed the maximal parameter t_{max} for E_D. In the extremely unlikely case that $t_{max} = \infty$, it is set to a large positive value. The right image of Figure 3.6 demonstrates the maximum step size spatial hash. For each boundary vertex U_j with search direction V_j (blue arrows), the vertex

is inserted into each grid cell intersecting the line segment between U_j and $U_j+V_jt_{max}$. Then, for each boundary edge with vertices U_1, U_2 and search direction vectors V_1, V_2 (dark red arrows), we query the grid for all boundary vertices that intersect the bounding box (red) given by U_1, U_2, $U_1 + V_1t_{max}$, $U_2 + V_2t_{max}$. The union of those boundary vertices (minus those of the edge) gives the only boundary vertices that need to be checked versus the given edge and reduces the amount of computation substantially. Moreover, as t_{max} is updated during this computation, the size of the query bounding box shrinks as well. Including this spatial hashing and using both spatial hasing techniques for the evaluation and maximum parameter reduces the timings of the overall optimization by approximately 92%.

3.5 Implementation

In this section I will give a brief overview of the implementation of our bijective parameterization algorithm.

First, the input is a triangulated surface with an initially prescribed seam. The triangles are preprocessed through an isometric flattening, and a precomputation of the areas and cotangent weights for the 3D triangles needed in the function and gradient evaluations.

Second, the initial starting parameterization is found using Tutte's method [54] for graph embedding. For the interior point method to work, any starting position can be used as long as it is guaranteed to produce a bijective mapping between the parameterization space and the 3D surface.

Next, we optimize the parameterization using a nonlinear Quasi-Newton approach optimization approach. The following steps are looped until the parameterization converges in terms of some threshold of error.

1. Evaluate the function and gradient.

2. Calculate the search direction.

3. Calculate the maximum step size t_{max}.

4. Perform the line search.

5. Update vertex positions and check convergence.

Step 1 of the optimization is to evaluate the function and gradient of $E_D + E_B$ at the current position of the parameterization. To produce the function and gradient of our distortion energy E_D, we evaluate the set of triangles in parallel. Since a triangle's distortion evaluation is independent of other triangles, the total evaluation of E_D is trivially parallelized. This parallelization using 4 processors speeds up the evaluation of E_D by an average of approximately 64% on the tested data sets. Next, the function and gradient is evaluated for the boundary barrier energy E_B. First, the vertices are binned to the grid of the spatial hash. Querying the spatial hash gives another opportunity for parallelization as each boundary edge with their bounding box expanded by ϵ can query and evaluate all vertices found within its bounding box. Even with the additional overhead of building the spatial hash, parallelizing the edge queries and evaluations using 4 processors speeds up the evaluation of E_B by an average of approximately 57% on the tested data sets.

In step 2, we now need to calculate the search direction of the vertices to best minimize the distortion. Any Quasi-Newton approach can be used to generate a search direction. We did multiple experiments using various Quasi-Newton approaches specifically, gradient descent, L-BFGS [31], and Levenberg-Marquardt[35]. Gradient descent uses the negative gradient $V = -1 * G$ as the search direction. While, gradient descent quickly computes the search direction, we found that using just the negative gradient causes a slow convergence and increases the number of itera-

tions required to converge. Using gradient descent for the camel data set causes the parameterization to converge in over 20 thousand iterations. Levenberg-Marquardt computes the search direction using the Hessian H of the function with the following formula, $V = (H + \lambda Diag(H))^{-1}G$. Where $Diag(H)$ gives the diagonal of the matrix and λ is a non-negative damping factor adjusted at each iteration depending on the speed of convergence. When λ is small the search directions is closer to the Gauss-Newton algorithm, and when λ is large, the search direction is closer to gradient descent. The Levenberg-Marquardt algorithm converges with significantly fewer iterations, approximately 400 for the camel example. However the computation of the Hessian and sparse matrix inversion is computationally expensive and causes a large increase in timings with the computation of the search direction taking two orders of magnitude more time to compute.

A good mix of convergence rate and computational complexity is found using L-BFGS. L-BFGS approximates the inverse Hessian H_k for iteration k of the function $V = H_k^{-1}G$ using information from previous m iterations of the optimization. To estimate the Hessian, store information from previous iterations of $s_k = U_{k+1} - U_k$, $y_k = G_{k+1} - G_k$, and compute $\rho = \frac{1}{y_k^T s_k}$. To produce the search direction, we perform Algorithm 1

The search direction computed from this process is significantly faster, about two orders of magnitude, than the computation of the search direction from Levenberg-Marquardt. While the search direction for L-BFGS is, about twice as slow as gradient descent, the number of iterations it takes to converge is approximately an order of magnitude lower. Through experimentation L-BFGS tended to perform with a good mix of speed and memory usage.

The number of previous iterations m used in our implementation was determined from experimental runs of all of our data sets. In Figure 3.11 the graph displayed

Algorithm 1 L-BFGS search direction

1: $q = G$
2: **for** $i = k-1, i >= k-m, i = i-1$ **do**
3: $\quad \alpha_i = \rho_i s_i^T V$
4: $\quad q = q - \alpha_i y_i$
5: **end for**
6: $H_k^0 = \frac{y_{k-1}^T s_{k-1}}{y_{k-1}^T y_{k-1}}$
7: $V = H_k^0 q$
8: **for** $i = k-m, i <= k-1, i = i+1$ **do**
9: $\quad \beta_i = \rho_i y_i^T z$
10: $\quad V = V + s_(\alpha_i - \beta_i)$
11: **end for**

shows the timings (y-axis) of running the full optimization with a varying number of past iterations stored (x-axis) to approximate the inverse Hessian. The timings shown are normalized to the number of past iterations $m = 1$, corresponding to gradient descent. The general trend in the graph is that all of the timings decrease and come to a minimum then increase from this minimum the more iterations that are added past 10. To correctly choose the number of previous iterations m to store we ran the optimization for several test meshes and found that setting m to 4-7 produces the best timings for all of the test meshes. All timings in this dissertation are set to store 5 previous iterations of the optimization.

Step 3 is to calculate the maximum step size t_{max} by explicitly computing the singularities along the search direction vector for E_B using Equation 3.4 and E_D using Equation 3.5. To compute the singularities associated with E_D the points must be binned to the spatial hash like described in Section 3.4.1. Then in parallel, sets of edges and vertices are queried from the spatial hash to compute singularities. The minimum singularity found is used as the maximum step size in the line search of the next step. The computation of the maximum step size is parallelized similarly to

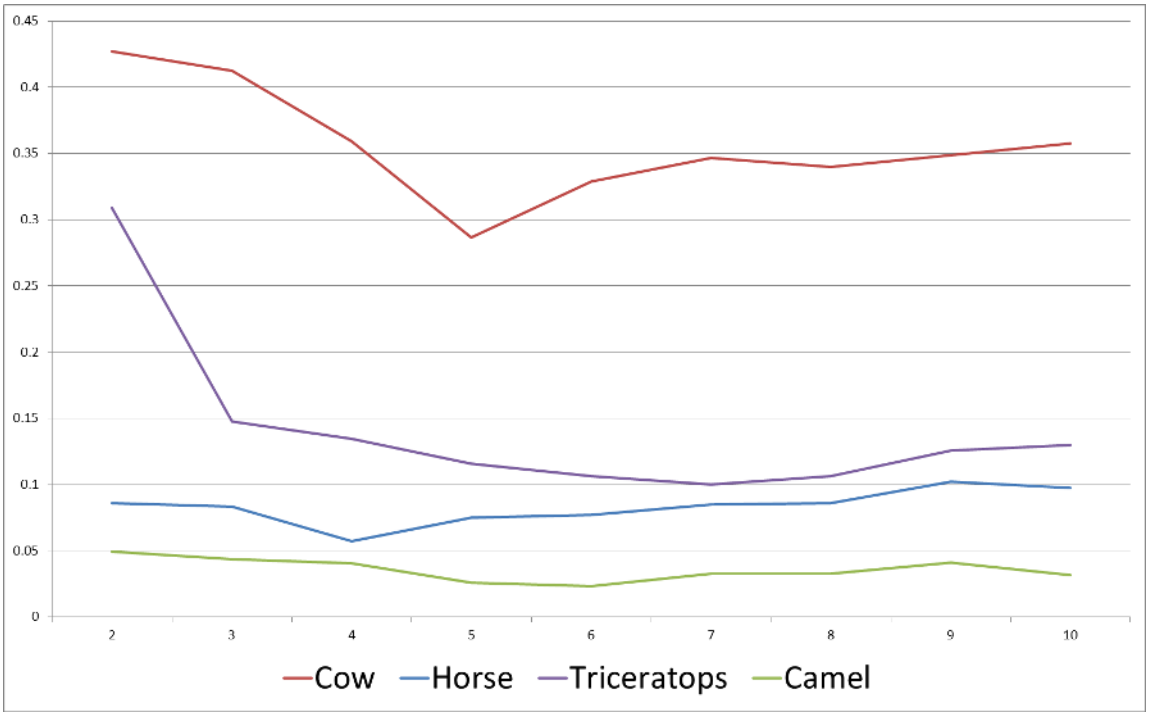


Figure 3.11: Graph of multiple data set's timings (y-axis) using L-BFGS normalized to the timing of using gradient descent. The x-axis is the value for the number of previous iterations m used inside of L-BFGS.

the evaluation of the function and gradient, and the singularities can be computed in parallel with 4 processors saving approximately 60% of the time of this computation. Figure 3.12 demonstrates the overall scaling when combining all of the parallelization together. The graph shows the timings of the overall optimization of the data sets normalized to using 1 processor.

Using the maximum step size t_{max} and the search direction, the next step is to perform a line search to minimize the distortion. After computing a step in the direction along the search direction, we update the vertex positions by moving them along the search direction and check convergence tolerances. To check if a parameterization is converged, we simply check if the difference between consecutive

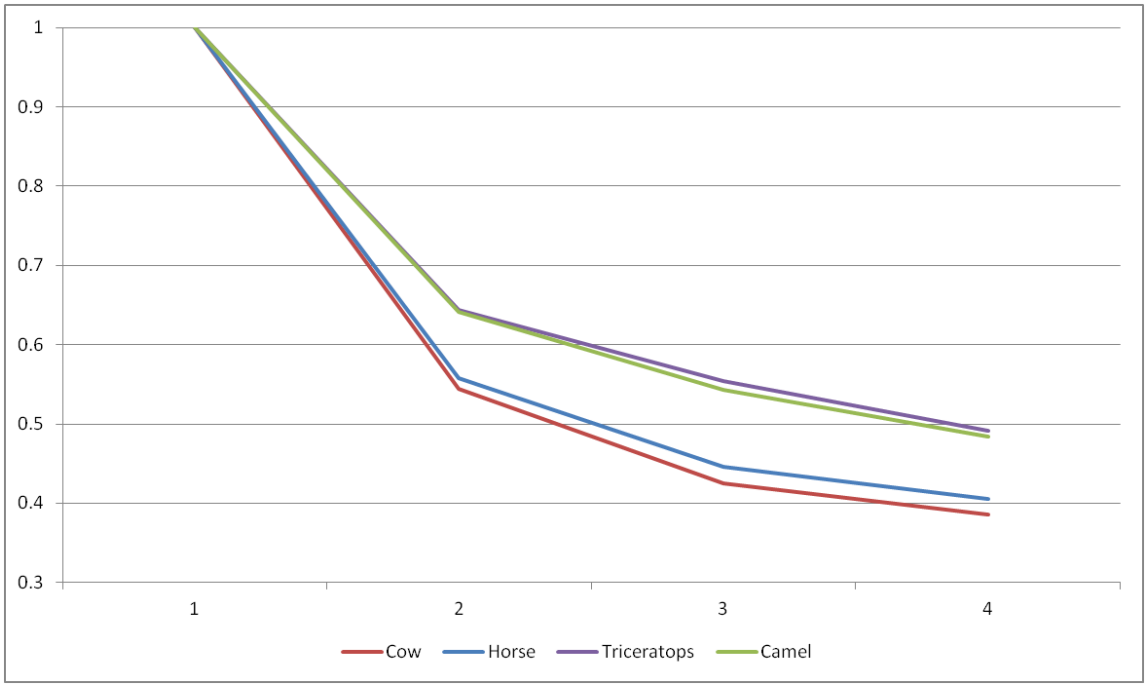


Figure 3.12: Graph of multiple data set's timings (y-axis) normalized to the timing of the parallelized with 1 processor. The x-axis is the number of processors used.

iterations' error metric is below a specified threshold

3.6 Results

Figure 3.8 shows the optimization in progress starting from Tutte's embedding of the camel model on the left. Each image from left to right shows different steps of the minimization process showing the convergence of a highly distorted parameterized camel and ending in the low distortion parameterization on the right. The optimization minimizes the isometric distortion E_D guaranteeing no triangles reverse orientations, as well as the bijective term E_B and at every step of the optimization, the parameterization remains a bijective map despite the highly intricate boundary interactions that occur during the optimization.

While our method uses a particularly simple form of isometric distortion in Sec-

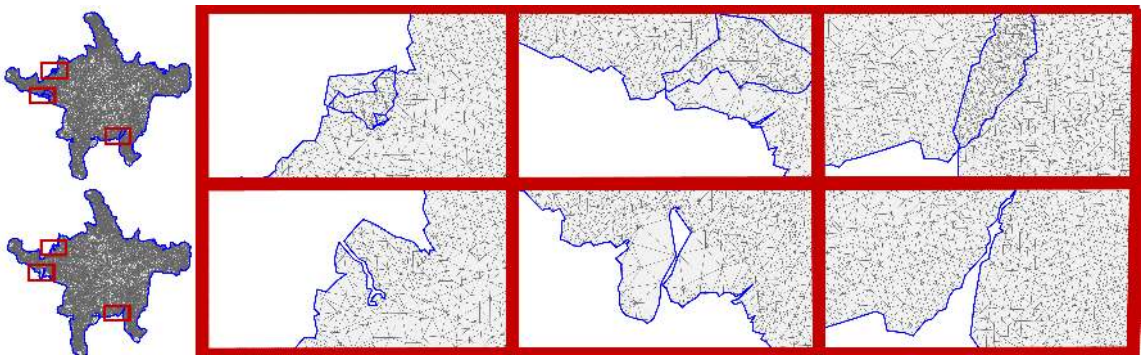


Figure 3.13: Parameterization of a horse model without our boundary term E_B (top) and with (bottom). From left to right are zoom-ins on various sections of the parameterization that demonstrate the lack of bijectivity (top) versus the results of our bijective parameterization (bottom).

tion 3.2, any injective barrier distortion function can be used in the described optimization framework. Figure 3.2 shows an example of five different metrics within the injective optimization framework without adding the boundary term E_B; conformal and MIPS measure conformal distortion followed by maximal isometric, isometric, and ours measuring a form of isometric distortion. The timings of the optimization are shown in seconds with each metric on an Intel Core i7-3770k CPU running at 3.5 GHz. The difference in timings shown here are due to the differences in implementation of the function and gradient evaluation, all other operations of the algorithm remain the same. Changes in timings maybe due to the complexity of the gradient and function evaluation, or the quality of search directions produced affecting the speed of convergence and the number of iterations. Both MIPS and our isometric metric have a simple distortion metric that is easy to compute. While the timings are implementation dependent, the simple form of these distortions yield fast optimization times.

While there are no flipped triangles in any of the examples in Figure 3.2, none of

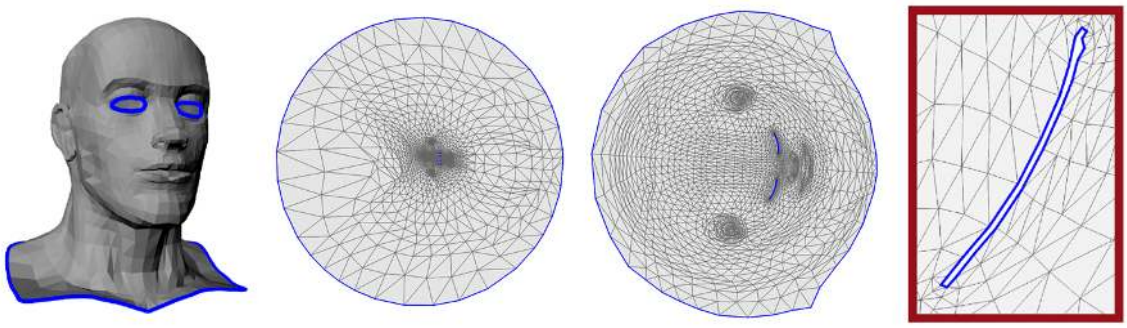


Figure 3.14: Parameterization of a chart with multiple boundaries (left) and the initial parameterization (left middle) via Tutte's parameterization by arbitrarily triangulating the holes in the eyes. The right shows the results of our parameterization with the temporary triangles in the eyes removed and a zoom-in on one of these boundary curves.

these parameterizations are bijective since the boundary intersects itself in each example. Figure 3.7 shows the same optimization and metrics except with the boundary barrier term, E_B, to each of the distortions. In this case, all of the parameterizations are bijective. However, the optimization takes longer in all cases except for the conformal metric where the optimization was terminated earlier and did not spend as much time optimizing over the folded configurations due to the bijective constraint.

An interesting aspect of these comparisons is that adding the boundary term still produces similar converged results as the non-bijective parameterizations. Figure 3.13 shows another example with the injective and bijective distortion metrics of a much more complex model of a horse with 20,636 vertices. The top of the figure demonstrates the optimization without the boundary term E_B. While there are no flipped triangles, the shape folds on itself in several places. Adding the boundary term to the optimization on the bottom row eliminates these intersections. In some cases, such as the left zoom-in, the boundaries form complex, matching curves but still remain intersection free.

Another advantage of this bijective optimization approach is that it is not limited to charts that are topologically equivalent to a disc. Most parameterization approaches require charts to be equivalent to discs. However, this framework can easily incorporate charts with holes. The only difficulty with incorporating holes is finding a valid starting point that contains no intersecting boundaries or folded triangles. The general strategy here is to simply find and fill the holes with arbitrary triangulations. By triangulating the holes, arbitrarily, Tutte's embedding or Floater's parameterization will now produce a valid starting configuration. Then the extra polygons are simply removed from the parameterization, and the original boundaries and mesh topology input into the method are now guaranteed to be in a valid bijective parameterization. From here, the optimization is run like normal. Each of the boundaries, the original and hole boundaries, are independent from one another since interior triangles would have to collapse causing a singularity for the separate boundary curves to intersect. Therefore, during the optimization, separate spatial hashes are used for each boundary curve, which lowers the boundary term's computational cost.

Figure 3.14 shows a challenging chart from a face model with three boundary curves, one for each eye and one around the neck, that creates a significant amount of distortion. To perform a bijective parameterization of this chart, first a boundary must be picked to be the primary boundary mapped to the circle for Tutte's embedding. In practice we choose the longest boundary which is the neck in this example. The other boundaries of the chart, the two eye holes, are then filled in with a simple triangulation. The holeless chart is then parameterized with Tutte's embedding. After the initial starting position is found, the temporary triangles are removed. The upper right image shows the resulting Tutte's embedding using triangulated holes for the eyes (with the artificial triangles removed). After the removal of the added

Method	Ours E_D		Ours $E_D + E_B$	
	avg	max	avg	max
Cow	5.466	14.763	5.843	14.751
Camel	9.203	28.082	9.351	27.216
Triceratops	4.327	17.465	4.455	12.669
Horse	7.280	26.499	7.300	39.890

Table 3.1: The average error and maximum error using our isometric metric E_D for all of the models in the paper with and without enforcing bijectivity. The minimum possible error is 4. Note that, though E_B is used in the bijective optimizations, only the error E_D is reported in the table above.

triangles, we can run our optimization shown on the bottom left. The holes remain intersection free and the parameterization forms a bijection which is demonstrated by the zoom in on the bottom right.

Note that such a hole-filling strategy has been used previously in parameterizations [19, 36]. However, in these cases the triangles were left in the optimization and optimized with the same distortion metric as the actual surface triangles. Such a strategy simplifies the optimization since the topology of the interactions between boundaries is known and fixed, but this choice increases the distortion of the remaining chart triangles unnecessarily. In this case, the geometry of these holes and the arbitrary triangulation chosen affect the resulting parameterization despite having no corresponding surface triangles.

Table 3.1 shows a table of both the average area, calculated as an area-weighted sum and normalized by total area, as well as the maximum error for an individual triangle using our bijective isometric metric for the different models. The minimum value of our error metric is 4, which corresponds to a perfect isometric flattening. In all cases the average error increases slightly when adding our boundary term E_B to the optimization to make the parameterization bijective. Figure 3.15 shows a color

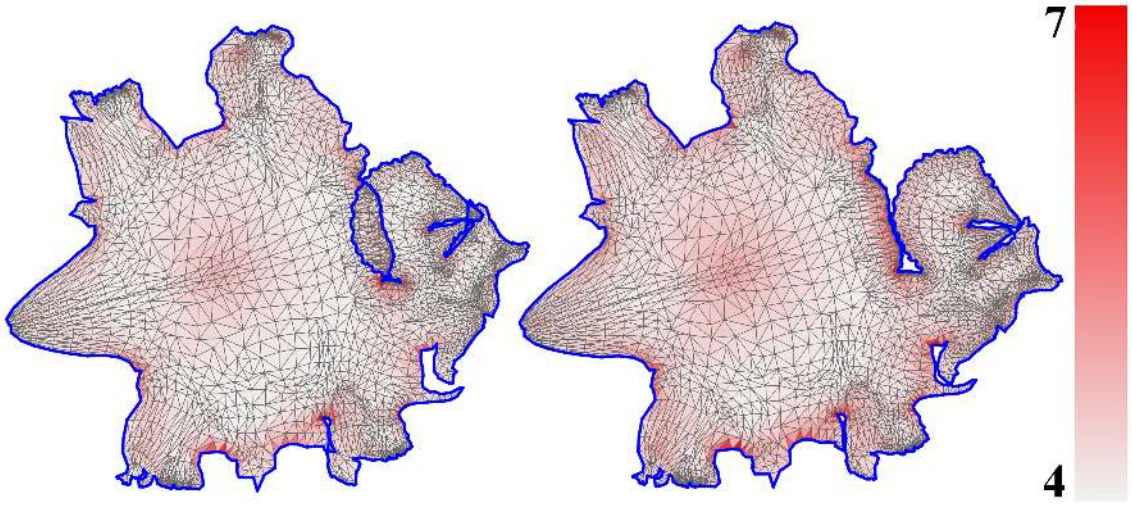


Figure 3.15: Triceratops optimized with E_D (left) and $E_D + E_B$ (right) color coded by our isometric error for each triangle.

map of the error of a parameterization without our boundary barrier term (left) and with our boundary barrier term (right). The image clearly shows that the error of the parameterization increases when forced to be bijective, particularly in the areas of large overlap from the injective optimizations.

This leads to a discussion about an initial requirement of this dissertation. We originally make it a point to ensure that the input seam is never modified or constrained in any way. However, if the increase in error is too significant, we could modify the seam unless it is a requirement of the problem. If the error is above a specified threshold, charts could be split into multiple pieces similar to [26] or [63] and re-optimized using the bijective approach. The advantage now is that we always guarantee the bijective map and don't require any seam modifications, with the tradeoff for increased distortion error.

Figure 3.16 shows a comparison of our isometric metric and boundary term versus several popular parameterization algorithms including a spectral form of LSCM [30],

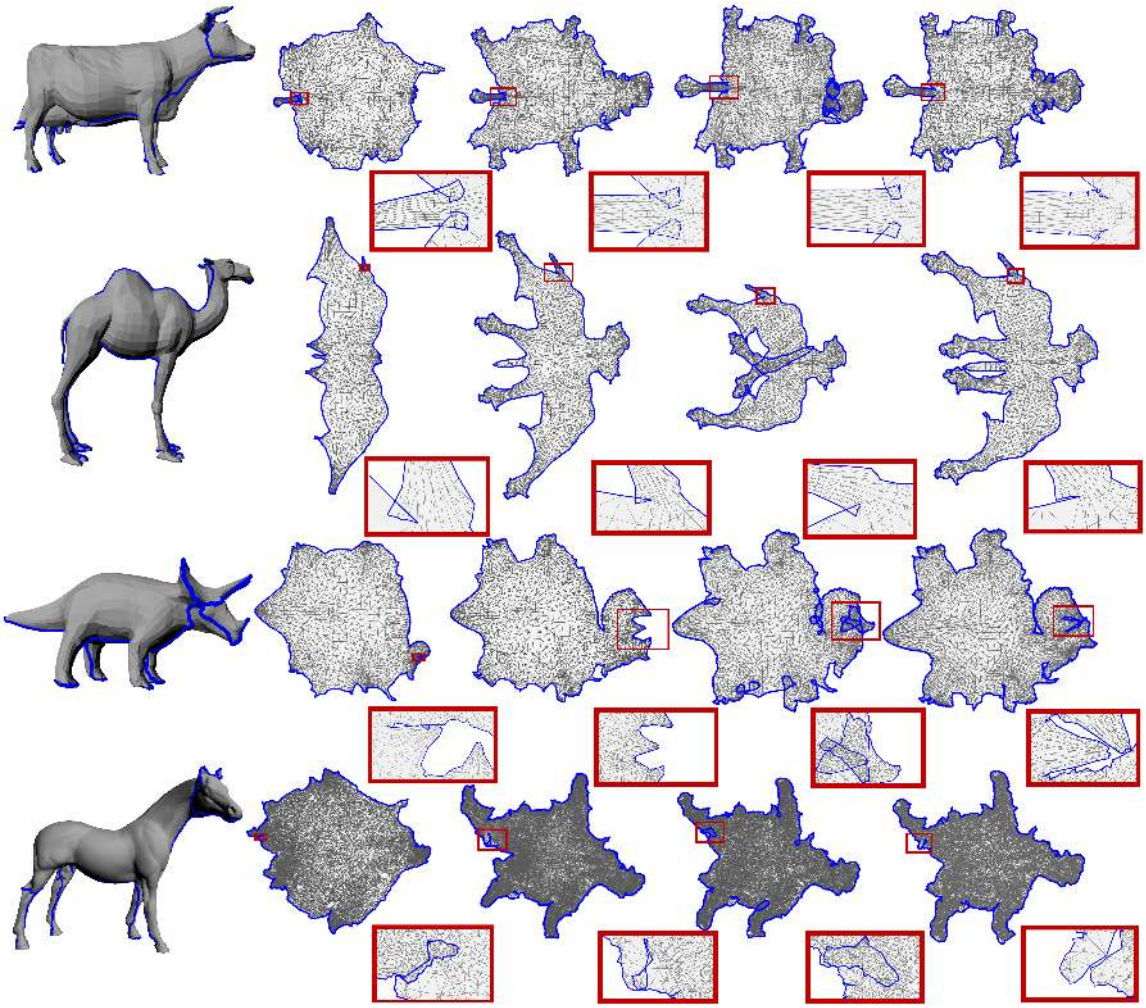


Figure 3.16: A comparison of widely used parameterization methods applied to different models. The methods from left to right are: spectral conformal parameterization, ABF++, ARAP, and ours.

Method	Faces	Vertices	Boundary	Spec	ABF++	ARAP	Ours $E_D + E_B$
Cow	3195	5804	584	.69	.004	1.99	3.63
Camel	2032	3576	486	.409	.006	2.12	6.13
Triceratops	3163	5660	664	.98	.008	3.76	4.45
Horse	20636	39698	1572	8.34	.028	118.31	35.48

Table 3.2: The time taken in seconds for all of the results in Figure 3.16. Faces give the number of triangles in the chart. Vertices gives the number of vertices in the chart. Boundary gives the number of vertices on the boundary of the chart.

ABF++ [47], and ARAP parameterization [29]. All of these parameterizations are nonlinear in nature. The first two methods (LSCM, ABF++) optimize a form of conformal distortion, while the last two (ARAP, ours) optimize a form of isometric distortion. The parameterizations are somewhat similar in nature despite optimizing different distortions except for the LSCM-based solution that causes shrinking due to the choice of distortion metric. We also show a zoom-in of the same area below each shape. All parameterization methods but ours fail to produce a bijective map. The only exception was ABF++ on the dinosaur example. Such parameterization methods can produce bijective maps (rarely for complex examples), but cannot guarantee the bijectivity. In contrast, all of our results are guaranteed to be bijections.

Table 3.2 shows the time taken in seconds for all of these methods. While our method is almost always the slowest in these comparisons, our method is still fast and computes parameterizations of charts with several thousand vertices within a few seconds. Even for large charts of tens of thousands of vertices our optimization still finishes within about half a minute.

4. VISIBILITY AWARE PARAMETERIZATION

When producing mappings for texturing we must also consider the usage of texture space. A texture image is generally a square 2D array of color values representing a pixel or "texel" of the image. The parameterization maps these color values onto the 3D surface, and the more texel's inside of the parameterized shape, the more detail achievable for this piece of the 3D surface. An isometric flattening produces mappings where the surface area of the 3D surface and the area of the 2D parameterization be equivalent. Having equivalent areas gives equal texture density to the entirety of the mesh. Another way to approach the problem of parameterization is that important regions of the model should have higher texel density than other regions, allowing an artist to create sharper, more detailed textures in regions of interest.

While many different metrics can be used to measure "importance", this dissertation proposes using visibility to measure how important a surface region is. We use visibility for importance because it is simple to precompute, doesn't require knowledge of the texture, and there is no need for user intervention. The visibility of a surface is different depending on how the surface will be used and different viewing models will change how often triangles in the mesh are seen thus modifying their texel density in parameterization space. The idea is to optimize the parameterization for the views from which a mesh is seen and in turn increase the average density of texels drawn in visible regions while using the same texture area.

4.1 Background

A parameterization determines how texels in a texture appear on the 3D surface. Parts of the texture that do not map to the 3D surface, or are rarely seen, use

memory without providing any benefit to the visualization of the surface. Another issue is that while memory is becoming less expensive, textures consume a significant amount of GPU memory, and there are several strategies to reduce memory usage, such as compression. There are several texture packing algorithms which attempt to minimize the area of unused space in the texture image by packing multiple charts together in a single texture image. The survey by Jylänki [23] gives a recent analysis of several techniques for packing textures together. The other main concept for improving memory efficiency is to compress the texture images themselves. Methods such as the DXTC format [21] compress textures for on-the-fly GPU decompression.

This chapter considers an approach to improve the efficiency of texture usage to supplement compression. Texture detail in a region of a surface is proportional to the ratio of the texture area to surface area in the region. Parts of the surface that are less visible should store less detail because they are less important to a rendered image. The general goal is to create a parameterization such that more visible regions are given more texture area. Within a chart, the texture space may be used inefficiently because some parts of the surface will distort more than others. Conformal surface methods, for example, preserve angles but do not constrain the scale of triangles. The result is that large surface areas on an object will often map to small areas in the texture. This small texture area causes low detail texels to map to the surface, even when the surface is prominently visible.

Large differences in scale are undesirable because high-resolution textures are needed in order to have sufficient detail over the entire surface. When a chart with positive Gaussian curvature is flattened, the center of the chart shrinks relative to the boundary of the chart. But, the centers of charts often contain the most important parts of the object because artists will hide the texture seams at chart the boundary from view. Thus, the most important parts of an object often have the least texture

Isometric Uniform Front half Top ring Side point

Figure 4.1: Different parameterizations shown using the same checkerboard texture to show relative texel density. Triangles are shaded by how often they are seen from the different viewing models. From left to right: an isometric parameterization and our method calculated with uniform visibility, visibility from the front hemisphere, viewed from a turntable, and viewed from one direction on the side.

detail. Cutting a surface into more charts decreases the distortion within each chart, but the number of texture seams increases. It is therefore desirable to use a parameterization method that controls usage of texture area within a chart in addition to minimizing conformal error.

Ideally we want an error metric that uses additional importance information to modify the area of texture space to match the importance value. Importance in this work will be described as visibility because meshes are often viewed under specific viewing transformations depending on the various applications the surface could be used in. This means that some regions of a mesh are more visible than others. Even when viewing a model from all directions with equal probability, not every region is viewed equally. For example, concave features of a mesh will be seen less frequently. While, visibility is the main importance metric of discussion for this dissertation, other metrics of importance could also be used, like texture complexity, geometric complexity, or even artist driven weightings.

This dissertation modifies the previously discussed isometric distortion energy E_D to use texture space based on the importance of the 3D triangles. Visibility is reduced both by a mesh occluding itself from some viewing directions and from tangentially viewing unoccluded triangles in the mesh. The best parameterization therefore depends on what angles the mesh is viewed from, which, in-turn, depends on the context in which the mesh is used. Figure 4.1 shows an example comparing our visibility-aware parameterization against the isometric parameterization under a variety of viewing scenarios. We draw portions of the surface seen more often in a light color and less frequently seen in a dark color. We use the same number of texels in all of the parameterizations, but our method allocates texture space so that texture density matches the visibility of the surface.

In terms of previous work, there are a few works on modifying parameterizations based on metric other than local geometry. For example, signal specialized parameterization [37, 53] assigns more texture area to parts of a mesh that have high levels of detail. This weighting of texture area increases the fidelity of texture samples. Signal specialization can also increase texture detail as needed while painting a textured surface [5]. The principle of changing texture area based on texture detail is similar in nature to the goal of this chapter, except that it requires knowing the texture before parameterization. Our visibility metric is independent of the image mapped to the surface. In addition, such weighting can cause significant folds in the parameterization both locally and globally. However, using the optimization from the previous chapter, bijectivity will be guaranteed and folds will not be able to happen.

Localized parameterization can be thought of as a method for non-uniformly distributing error in parametric distortions. Rather than parameterizing an entire mesh, a parameterization is only generated around a region of interest such as a

decal [40] or a drawn stroke [39]. These methods employ exponential maps, which map radial geodesic curves to polar lines in the parametric space only defined within the radius around a point. These maps have the property that parametric error is zero at the point of interest and distortion increases with distance from the point of interest. Using the importance can also be thought of as increasing the quality in regions of interest, but instead of simply reducing distortion error more texture space is allocated.

While visibility has been used for many aspects of Computer Graphics such as the construction of chart seams [46] or even surface simplification [62], we found only one instance where parameterizations are modified by visibility. Diffusion curves, a vector-based primitive used to create smooth-shaded images, require solving a linear system of equations over an image or a texture. In order to accelerate the calculation of diffusion curves in a texture, it is possible to decrease texture resolution for unseen parts of a surface [22]. Decreasing the texture space reduces the computation required to generate diffusion curves. The reparameterization algorithm used is designed to be very simple because it was intended to update the parameterization in real-time. Whenever the camera moves, the surface parameterization is updated and the diffusion curves are recalculated.

4.2 Visibility-Aware Parameterization

A mesh is typically seen from more than one direction. However, if there is no prior knowledge about where a camera is relative to the mesh, every triangle should be given equal importance as there is no prior knowledge about how an object will be viewed. If the importance of the mesh is all the same, ideally the method will compute an isometric parameterization, but if prior knowledge is known about how a surface will be viewed, we want to specialize the parameterization to better distribute

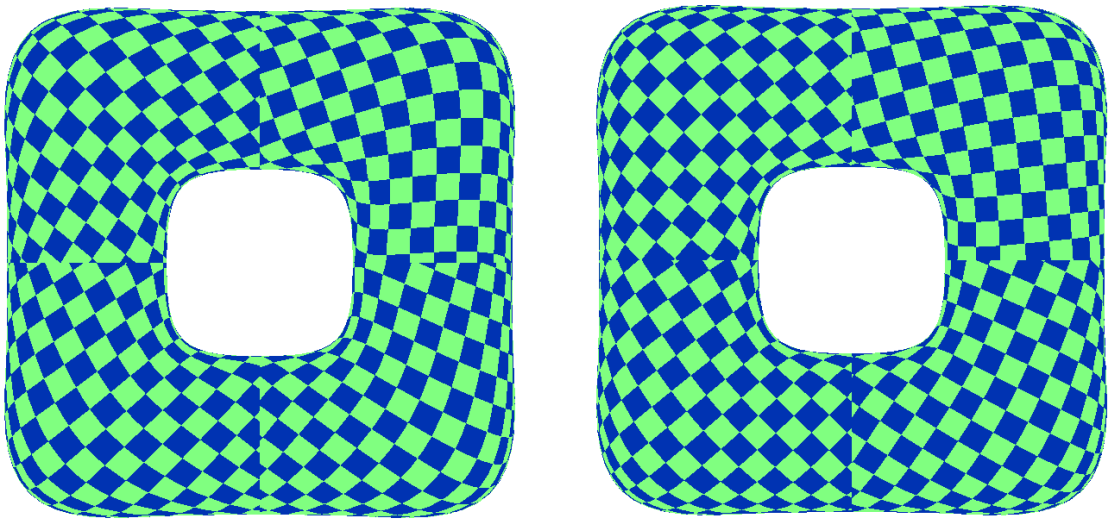
texture space. This process will give more texels to more visible regions of the surface allowing for clearer and sharper textures.

The goal is to allocate texture area in such a way that the apparent texel density is as high as possible. If we were to increase the scale of every triangle in the texture, the texel density over the surface would increase, but would also increase the overall number of texels required as the chart would take more room in texture space. To compare parameterizations fairly, they should contain equal texture area and normalize the total texture area used by the mesh. This approach allows the comparison of how well different parameterizations allocate texture space to different parts of the mesh.

Assuming the visibility for a triangle V_t is computed a priori, as discussed in Section 4.2.1, we will modify Equation 3.1 in two ways. First, we scale Equation 3.1 by V_t. This modification has the effect of reducing distortion in highly visible regions and hiding/increasing distortion in less visible regions. Second, we scale each of the singular values by $V_t^{-1/2}$. Previously, the energy metric E_D had a minimum when $\sigma_1 = \sigma_2 = 1$, but this modification changes the minimum to where $\sigma_1 = \sigma_2 = \sqrt{V_t}$ effectively making the ideal area in parameterization space for each triangle proportional to the visibility. Remember that $\sigma_1\sigma_2$ is the ratio of areas between the 3D triangle and the 2D parameterization and now with this change to the energy metric $\sigma_1\sigma_2$ is equal to V at the minimum. The modified energy is then

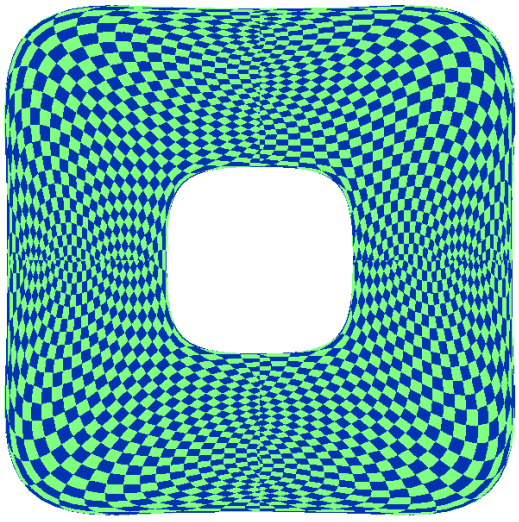
$$\hat{E}_t = V_t \left(\frac{\sigma_1^2}{V_t} + \frac{\sigma_2^2}{V_t} + \frac{V_t}{\sigma_1^2} + \frac{V_t}{\sigma_2^2} \right). \tag{4.1}$$

Note that this energy is similar in nature to [37], where the authors modified the stretch metric $(1/\sigma_1^2 + 1/\sigma_2^2)$. In the case of importance, measured per triangle, the signal-specialized parameterization simply becomes a weighted version of the

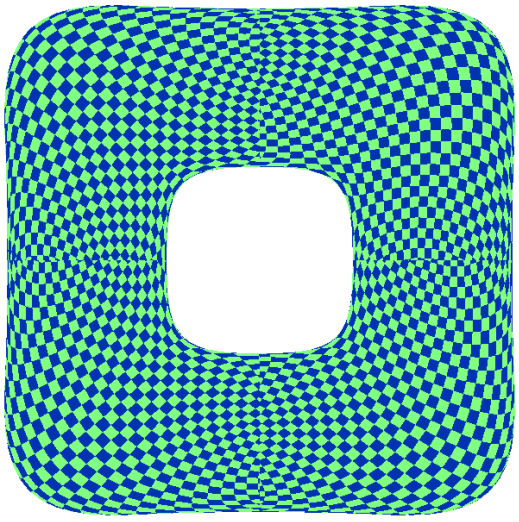


Isometric

Weight



Scale



Scale & Weight

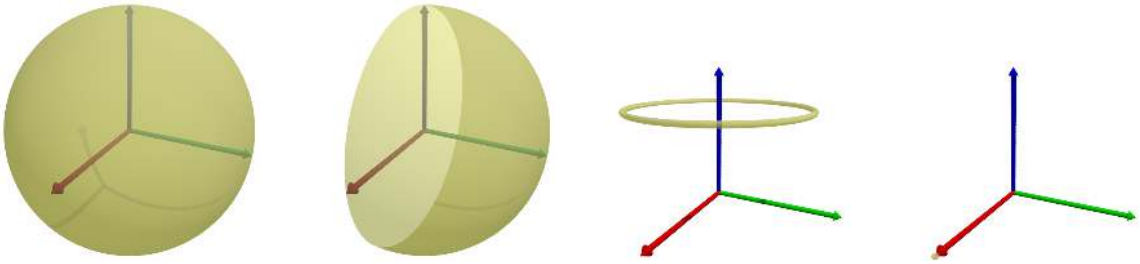
Figure 4.2: Visualization of the density and distortion of various parameterizations using a single viewpoint: the unmodified isometric parameterization (E_t, top left), weighting by V_t ($E_t V_t$, top right), scaling the singular values ($\hat{E}_t / V_t$, bottom left), and both scaling the singular values and weighting by V_t ($\hat{E}_t$, bottom right).

stretch metric. However, the authors note that the stretch metric is not scale independent and becomes zero as σ_1, σ_2 approach ∞. To combat this problem, the authors multiply the error by the chart area. In the case of a single polygon, this area is proportional to $\sigma_1\sigma_2$, which is equal to the ratio of areas between the 2D and 3D triangles. Multiplying this area by the stretch error yields $\frac{\sigma_1}{\sigma_2} + \frac{\sigma_2}{\sigma_1}$, which is a weighted approximation to the conformal MIPS energy [19].

In contrast, the error $\hat{E}_t$ is more than a weighted form of isometric error. The weight factor of V_t helps to hide distortion in less visible regions while the individual scaling of the singular values controls the local scaling of the chart in parametric space. Figure 4.2 shows the results of our modifications for the viewing model of a single viewpoint. The upper left image illustrates the unmodified, isometric parameterization. When weighting the energy function E_t by the visibility V_t, distortion is reduced in the visible regions; although the distortion increases in less visible regions. Scaling the singular values (bottom left) produces an increased pixel density over visible regions. However, the distortion of the parameterization can be extreme. Both scaling the singular values and weighting by visibility (bottom right) produce high pixel density in visible regions as well as low distortion.

4.2.1 Computing Visibility

To determine how important a point on the surface corresponds to an infinitesimally small disk, which means that the area of p projected onto the screen is $v(p, r) = n_p \cdot r$. Where n_p is the surface normal at point p and r is the direction to the camera. The projected area is maximal when the camera faces the surface straight-on, and goes to zero when the surface is viewed tangentially. Occlusions must also be considered during this computation, as other pieces of the surface may occlude p. When p is occluded it has zero importance to the view and we adapt the



Uniform Front half Top ring Side point

Figure 4.3: Different viewing models that we tested our parameterizations with. View directions are represented as yellow-colored points on a unit sphere.

occlusion testing used in ray-tracing for our purposes.

$$v(p,r) = \begin{cases} n_p \cdot r & \text{if visible} \\ 0 & \text{if occluded} \end{cases}$$

We evaluate the visibility V_t of a triangle by integrating surface visibility over a hemisphere, which is given by

$$V_t = \frac{\int_{p \in \text{tri}} \int_{r \in \text{hemisphere}} m(r) v(p,r) \; dr \; dp}{\int_{p \in \text{tri}} \int_{r \in \text{hemisphere}} m(r) \; dr \; dp}. \tag{4.2}$$

The view model $m(r)$ weights how often the camera will view the scene from the direction r. We approximate the integrals in V_t using Monte-Carlo integration by choosing random points p in the triangle, and shooting random rays r from those points to the camera.

Viewing models used for testing are shown in Figure 4.3. The cameras in these models are infinitely far from the object so that the projections onto the camera are orthographic. The individual viewing models consider different ways an object may be viewed in specific applications. The uniform viewing model considers an object is viewable from all directions corresponding to calculating visibility over an

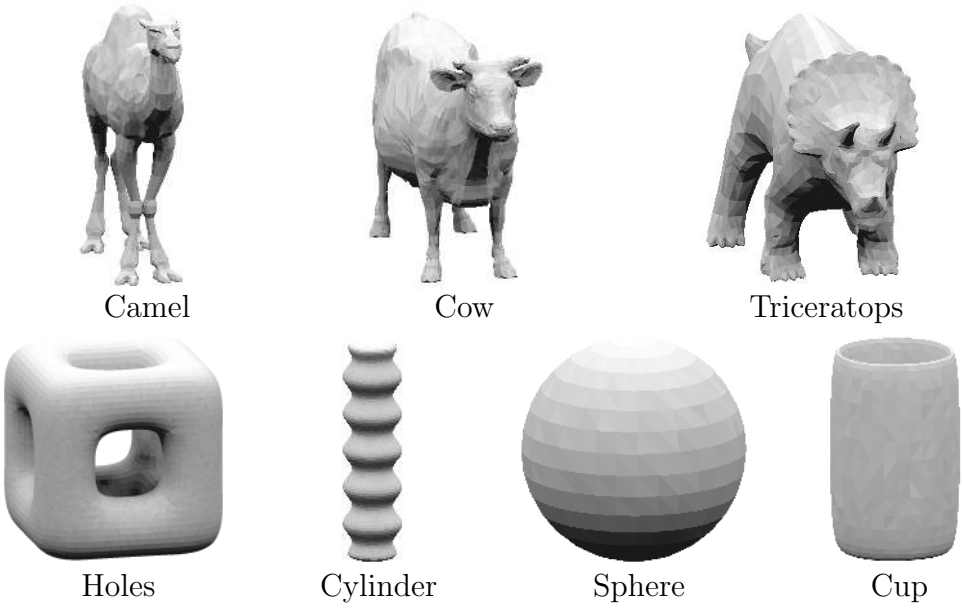


Figure 4.4: The models used to measure texel densities in Figure 4.8.

entire sphere. The front half model corresponds to a restricted view of the object represented by a region of the sphere. In some applications, the viewing model can be further restricted to a curve on the sphere like in the top ring viewing model. The most extreme of the viewing models is that of the side view corresponding to a single point on the sphere. Calculating visibility from all directions as shown in the uniform viewing model is equivalent to calculating the global ambient occlusion of points on the surface and tends to produce similar results to an isometric parameterization. However, the results can be significantly different for meshes with large cavities, like the cup shown in Figure 4.4 and its resulting parameterization shown in Figure 4.5, where the concave regions of the surface have smaller visibility values causing these regions of the surface to be given less texture space.

Further restricted viewing models, like the front model, requires the viewer to be in front of the object. This situation might occur for a scene viewed through a

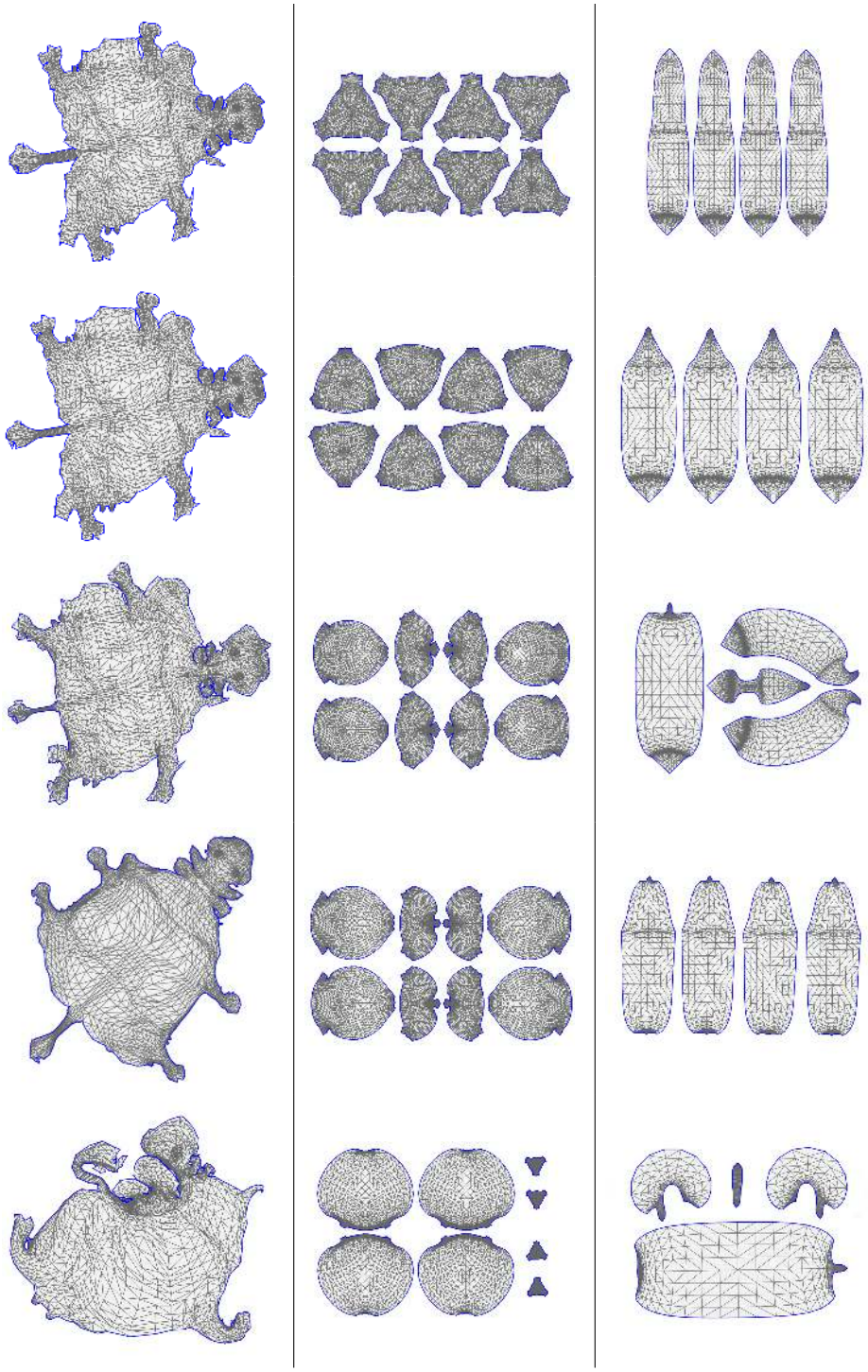


Figure 4.5: 2D texture atlases showing the parameterization of the cow, holes, and cup for different viewing models. The cow is cut down the neck and belly. The holes model is cut into eight identical corners with a seam on the inside. The cup is cut into four identical quadrants. The view models are arranged from top to bottom: isometric, uniform, front, to ring, side view.

window, like a display in a store. Valid views in this scenario can be visualized as a hemisphere in front of the object. Another possibility is that meshes are viewed from a fixed angle and rotated on a turn-table, as in an isometric drawing. This model, the top ring, can be found in some video game where the player is limited to just rotating the character. Because the view is determined from one parameter, valid views trace a curve over the surface of a sphere. The most restricted case is if a mesh is viewed from only one direction, such as being viewed from the side in a platforming game. This view is visualized as a single point on the sphere. Other viewing models are, of course, possible, but we consider these models as representative since they encompass the entire sphere, a region, a curve, and a point.

4.3 Results

We tested several example meshes, shown in Figure 4.4, that occlude the view in different ways. The top row is freely available meshes found on the web, and the bottom row are a collection of dramatically different shapes that we created. The camel, cow, and triceratops are composed of a single chart. The mesh with holes has complex topology and self-occlusion, the cylinder has ripples that cause rapid fluctuations in visibility, the sphere has a large and smooth change in visibility, and the cup is strongly self-occluding with the interior of the cup not seen by some viewing models.

Parameterizations in texture space are shown in Figure 4.5. Each row corresponds to the viewing model used to calculate the parameterization. For the cow, uniform visibility produces nearly the same result as the isometric parameterization. Viewing the cow from the front exaggerates the area of the cow, like the head and front shoulders which have been given much larger texture space, from that viewpoint while minimizing the area of regions like the tail, which are less visible. When viewed

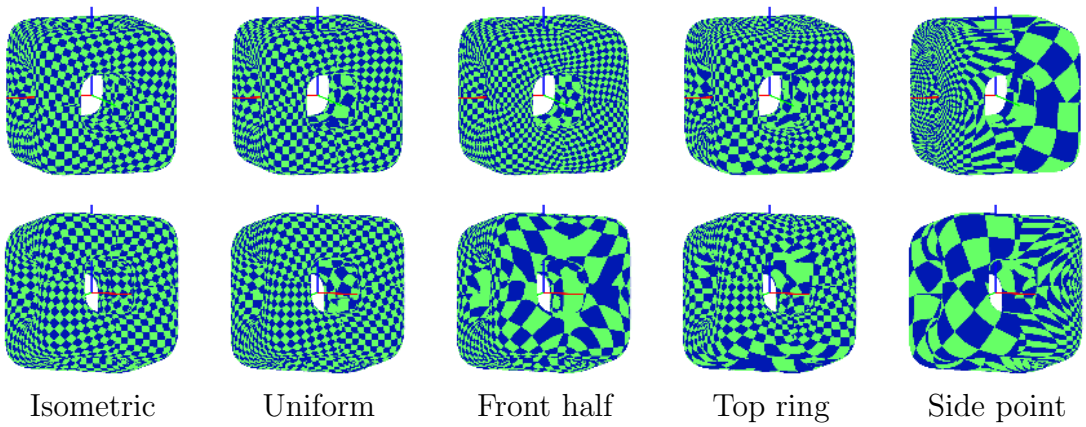
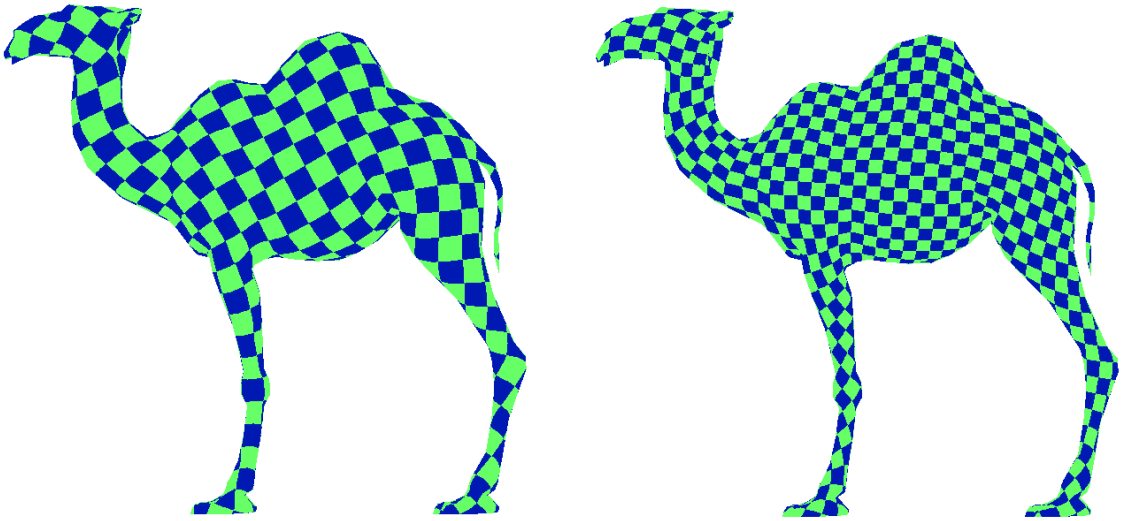


Figure 4.6: A mesh with different parameterizations from different views show from the front, looking down (top row), and the back, looking up (bottom row). Depending on the viewing model, opposite sides of a mesh may have very different texel densities.

from a ring above the model, the area of the underside of the cow, which contains the chart seams, is minimized, while the top of the cow or interior of the parameterization is allowed to grow in texture space. Viewing the model only from the side produces a highly distorted, but still bijective, parameterization that exaggerates the area of the cow along the visible region.

The cup is interesting because there is a large difference between the unmodified isometric parameterization and ours using a uniform distribution of viewing angles. Specifically, the interior of the cup is difficult to see from most angles. In Figure 4.5, the cup's parameterization is shown in the right column. The top parts of the charts are the interior of the cup, and given a large amount of texture area in the isometric parameterization. However, the inside of the cup is occluded, causing the uniform viewing model to shrink the top parts of the charts.

We demonstrate the effect of different viewing models on texel density by showing the holes mesh with a checkerboard texture from different angles in Figure 4.6. In the first column, the isometric parameterization has uniform texel density everywhere.



Isometric Side point

Figure 4.7: An example of a mesh seen from the side view. The mesh on the left uses an Isometric parameterization and the mesh on the right is parameterized for the side view.

The uniform distribution of views shown in the second column looks the same from the front and back, but the interior is partially occluded and has lower texel density. In the third column, we optimized for viewing anywhere in the front hemisphere so that the front and sides have higher texel densities than with the isometric parameterization. The back side, however, is unseen and has low density. Similarly, the top ring view in the fourth column has high densities on the top and low densities on the bottom. The view from a single point on the side is the most extreme optimization and has very high texel density on one side, but low density for all five other sides. Although the exterior in the $+x$ direction has high texel density, the interior facing in the $+x$ direction is low density, because the view of the interior surface is occluded.

It can be difficult to imagine how a parameterization like the side point distribution looks in practice, so we show the result from the viewpoint for which we

optimize in Figure 4.2 and Figure 4.7. Figure 4.2 can be compared with Figure 4.6 where the same parameterization is seen from non-optimal viewing angles. The side point distribution is the only distribution that optimizes for a single viewing angle and is therefore the only result we can show in a single image. Although some parts of the texture are very low resolution, only high-resolution texels are drawn from this viewpoint. The texel density is clearly higher for our parameterization than for the isometric parameterization.

To evaluate if our parameterization is effective at increasing texture density in the view distributions for which we optimize, we measure the average determinant of the Jacobian of the mapping from texture space to screen space. This measurement is more robust than measuring the Jacobian from screen to texture because triangles that project to zero area on the screen would have infinite texel density. The bars measuring texel densities in Figure 4.8 are arranged in a matrix, where the rows correspond to the view distributions we sample from and the columns correspond to the views for which we optimize our parameterization. Densities are reported relative to the unmodified, isometric parameterization. Hence, improved density yields a number greater than one. Because results depend on the model, we show the results for seven different models in all combinations of views and parameterizations. Higher densities are better, and a clear trend is that densities are greater than one and highest on the diagonal, which demonstrates that optimizing for a viewing model increases the texel density for that viewing model.

Although our method outperforms the unmodified isometric parameterization when we optimize for a predefined view model, isometric parameterization performs the best when the view model is not known. This effect is part of a larger trend where the benefit increases with the restrictiveness of the viewing model. From left to right in Figure 4.8, the view models are more specialized, up to the extreme case

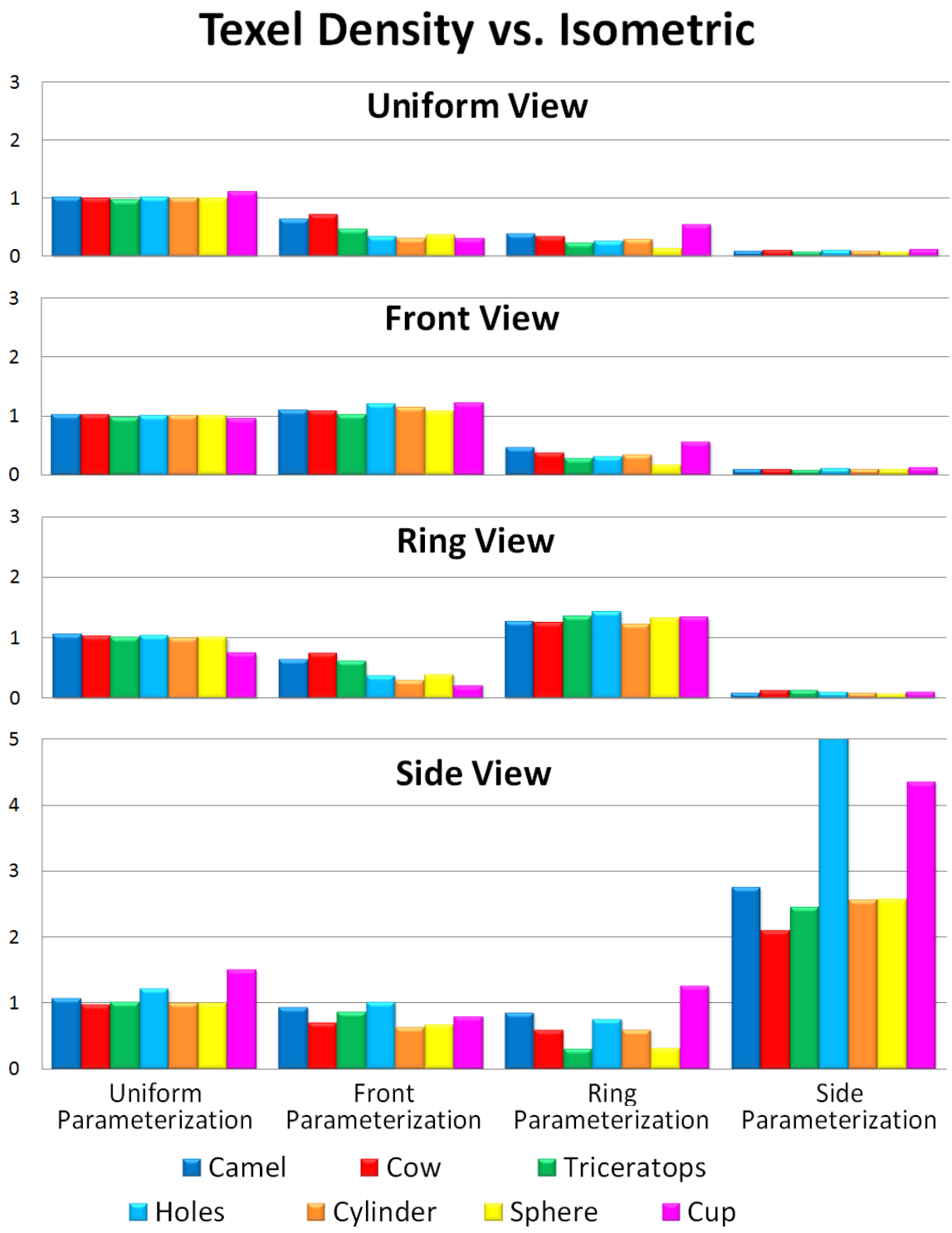


Figure 4.8: Graphs showing the visible texel density of our parameterizations relative to the texel density of Isometric parameterization. Each graph shows the average density of pixels from different distributions of views. Clusters of bars are labeled below by the view type the mesh parameterization was optimized for.

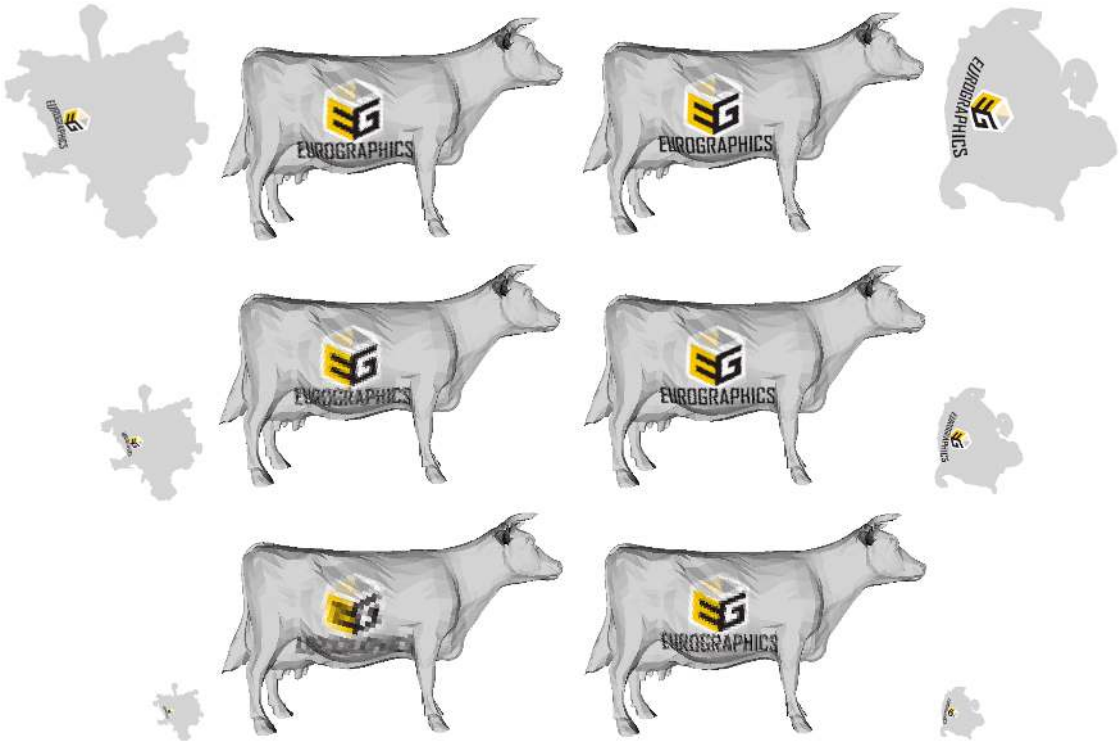


Figure 4.9: The Eurographics logo mapped onto the cow via an isometric parameterization (left), and our visibility-aware parameterization (right) from the side viewing model. From top to bottom we downsample the texture successively by a factor of two which is shown on the outside columns of the image.

of a side view from a single angle. Parameterizing for the side view performs the best of any method when viewed from the side, but the worst for any other viewing model. We can also see that the uniform viewing model tends to have similar results to isometric, because most meshes do not have a large degree of self-occlusion. The cup is the only mesh we tested that had significant benefits from parameterizing for uniformly distributed viewing angles, because the inside of the cup is occluded from many angles.

Figure 4.9 demonstrates the benefit of increased texel density using our method. In this example, the texture from an isometric parameterization is resampled to create a texture using our visibility aware parameterization. Both textures contain the same number of texels except ours has much higher resolution in the visible regions. Downsampling the images by a factor of 2 and 4 demonstrates much higher image fidelity with our parameterization. Hence, the visibility aware parameterization leads to higher quality images or, conversely, similar quality with smaller textures, which requires less storage and bandwidth.

Our error metric in Equation 4.1 and optimization prevents the parameterization from folding since $\hat{E}_t \rightarrow \infty$ as $\min(\sigma_1, \sigma_2) \rightarrow 0$. Adding the boundary barrier term from the previous chapter guarantees charts will not intersect and produces a bijective map. Figure 4.10 demonstrates the bijective nature of our parameterization. In this case, two of the charts have one side that is more visible from the selected viewing model than the other. The optimal solution is to increase the arc length of the visible chart boundary and minimize the arc length of the opposite, invisible boundary, which leads to a curling effect. Our method ensures that the resulting parameterization does not fold or intersect itself while being able to naturally curl.

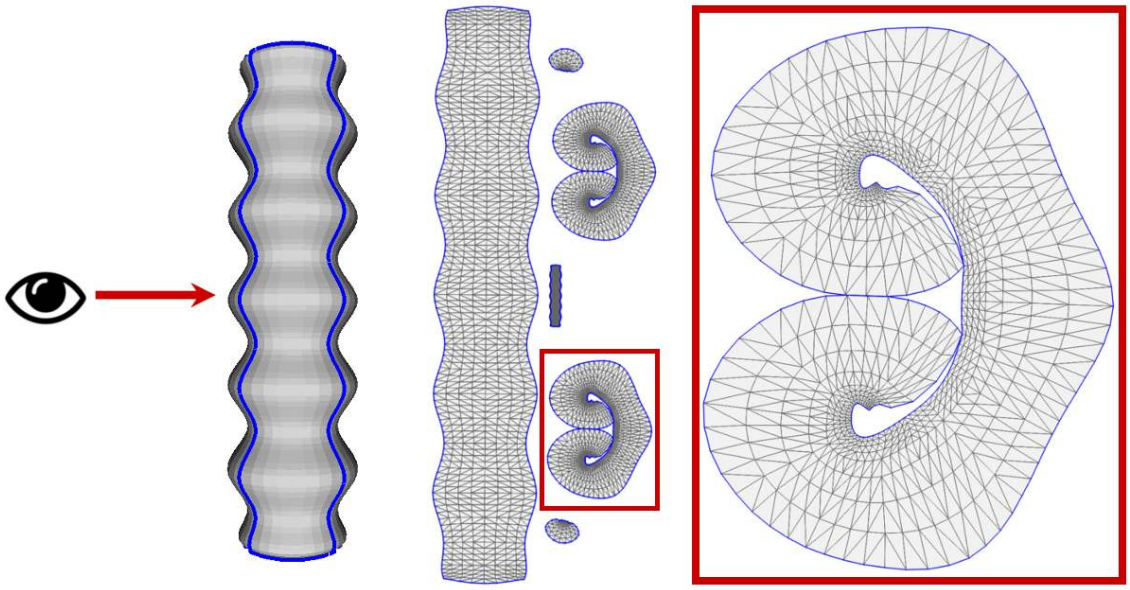


Figure 4.10: Demonstration of the bijective property for a parameterization optimized for the side-view.

5. CONCLUSIONS

This dissertation has presented a generalized optimization framework that produces globally bijective parameterizations that optimize various injective distortion metrics. To further improve the parameterization's distortion, we allow for the boundaries of the parameterization to be freed during the optimization removing the need to constrain or modify any of the boundaries. To ensure there are no global fold overs, we introduce a barrier energy function to allow for global bijective parameterizations. We introduce a singularity aware optimization framework allowing for the explicit computation of singularities of energy metrics to guarantee that our interior point approach remains in a valid solution space. Adding the additional barrier energy term to the optimization guarantees that the parameterizations are bijective and produce very similar parameterizations in terms of overall appearance and distortion.

Texture space is limited, and so, the space should be used preferentially for important parts of a mesh. We propose an automated method for determining which parts of a mesh are most important based on how often they are seen. If restrictions are known beforehand on how a mesh will be viewed, we can achieve a significant increase in perceived texture quality. The idea of weighting importance by visibility can be combined with other weighting metrics. For example, visibility could be combined with texture detail, as in signal-specialized parameterization, if the texture is known before calculating a parameterization. It may also be useful to allow artists to paint the relative importance of triangles on a mesh when importance cannot be algorithmically determined. For example, when parameterizing a mesh of a person, the face of the person should have higher weight than any other part of the mesh,

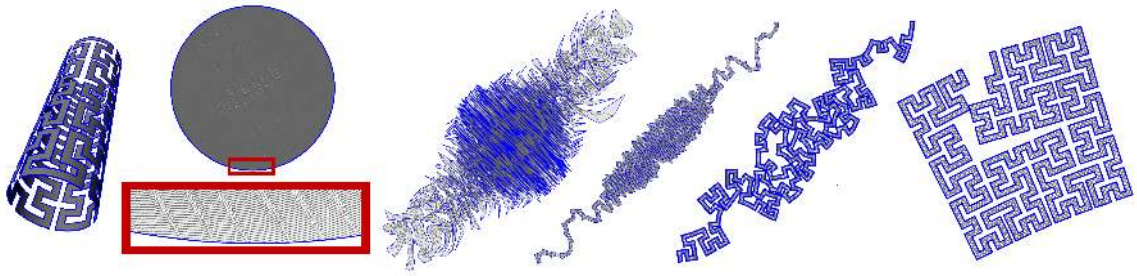


Figure 5.1: A failure case for the method. From left to right: a space filling curve on the surface of a cylinder, Tutte's embedding with a zoom-in below to show the poor triangulation, two intermediate steps during the optimization, our result with default parameters taking 49.6 seconds with an average error 13.238 and max 17.223, and the result using a lower convergence tolerance taking 8472.14 seconds with an average error of 4.210 and max 4.213.

although the visibility is similar.

5.1 Limitations

This bijective parameterization technique uses a non-convex energy function. Given this fact, there is no guarantee of finding the global minimum of the given distortion energy. However, it is important to note we start all of the optimizations with Tutte's parameterization. While this initial starting position is bijective and quick to compute, it may contain significant distortion as many of the triangles can be close to a degenerate state, and is typically quite far from the minimum the optimization finds. Even though, we start all of our examples we tested for this dissertation at a Tutte's parameterization, the optimization for all of our practical examples tends to find very reasonable solutions that appear to be close to optimal solutions.

To determine if the optimization can indeed get stuck in local minimum we construct a difficult example in Figure 5.1 as a failure case. In this example, we wrap a polygonal Hilbert space filling curve around a cylinder. Given the cylinder is a ruled

surface, the global minimum of the parameterization should be an unwrapped space filling curve with no distortion. The starting configuration of Tutte's embedding is far from the global minimum and every triangle embedded in the circle is almost degenerate producing a very high error. The figure shows several intermediate stages of the optimization before the optimization terminates in the second to the last image, taking 49.6 seconds. Obviously, at this point the parameterization is not the global minimum although the optimization made significant progress from the starting configuration. To verify the optimization is actually stuck in a local minimum using our default convergence parameters, we reduced the convergence tolerance and continued the optimization. The shape on the right is the local minimum our optimization finally reached after 8472 seconds. At this point the optimization is truly stuck although it stopped extremely close to the global minimum, but we could not lower the convergence tolerances any more. However, in the far less challenging cases in the rest of the paper, lowering our convergence tolerance did not significantly affect the parameterization. In addition, despite starting from the distorted initial parameterization provided by Tutte's embedding, all of the shapes we have tried have generated low distortion mappings comparable to other parameterizations methods with the important exception that we always create bijective maps.

Additionally, The parameterizations produced by our method may be more distorted than an unmodified isometric parameterization, especially in the case of very restrictive viewing models. However, our parameterizations purely using visibility may still be useful in compressing the space used by textures because the texture density is better distributed giving more texture space to more visible regions of the surface.

5.2 Future Work

In terms of future work, the method could greatly benefit from a better starting position other than the Tutte's embedding. Unfortunately, few methods can currently guarantee a bijection without user intervention. Questions to research include, are their better embedding techniques like [12] which may take additional computation to produce, but in the end improve the amount of iterations required for convergence? Are there better starting shapes to constrain the initial boundary to instead of just a circle? In this work, we choose a circle because it is a symmetric choice where no decisions must be made about what parts of the boundaries are located where. For example, if we constrain the boundary to a square, there is now the problem of assigning which vertices are constrained in the corners of the square.

Another possibility and common strategy to improve the speed of parameterizations is the use of multi-resolution methods [19]. While the method is relatively fast for smaller charts; for very large charts consisting of hundreds of thousands or even millions of vertices, better optimization approaches are needed. Not only are there challenges with deciding which type of hierarchical approach to take, whether we use a simplification style hierarchy, or some type of spatial decompositioning, we must adapt the multi resolution techniques to guarantee bijectivity. For example, maintaining a bijective map as a simplification structure re-expands to the full resolution mesh is quite challenging with respect to boundary intersections and injective triangle flips.

Another area of future research is to incorporate other properties into parameterizations using barrier energy functions to guarantee that they are upheld. For example, adding the requirement that the parameterization be seamless [34] would be useful for adding multiple applications to the resulting parameterization such as

requadrangulation. Such a modification would tie the optimization of all charts together since corresponding boundary edges would be required to be the same length and a multiple of a 90 degree rotation. Such methods can rely on integer constraints [3] and are difficult optimizations even without the injective and bijective constraint.

The idea of weighting importance by visibility can be combined with other weighting metrics. For example, visibility could be combined with texture detail, as in signal-specialized parameterization, if the texture is known before calculating a parameterization. It's also possible to incorporate geometry information, like geometry complexity. To allow even further control over the resulting parameterization it may be useful to allow artists to paint the relative importance of triangles on a mesh when importance cannot be algorithmically determined. For example, when parameterizing a mesh of a person, the face of the person should have higher weight than any other part of the mesh, although the visibility is similar.

Finally, this dissertation makes the assumption that the meshes are static. In reality, many surfaces are animated. It is unlikely that using the visibility from a single pose will produce optimal results for all poses. To incorporate animations into the importance weighting we would need to sample the visibility through the time of the animation. Future work would include incorporating time into the importance weighting for both self occlusions and possibly extra scene interactions with additional objects.

REFERENCES

[1] Noam Aigerman, Roi Poranne, and Yaron Lipman. Lifted bijections for low distortion surface mappings. *ACM Transactions on Graphics*, 33(4):69:1–69:12, 2014.

[2] David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. Integer-grid maps for reliable quad meshing. *ACM Transactions on Graphics*, 32(4):98:1–98:12, 2013.

[3] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. *ACM Transactions on Graphics*, 28(3):77:1–77:10, 2009.

[4] Brent Burley and Dylan Lacewell. Ptex: Per-face texture mapping for production rendering. In *Eurographics Symposium on Rendering*, pages 1155–1164, 2008.

[5] Nathan A. Carr and John C. Hart. Painting detail. In *ACM SIGGRAPH*, pages 845–852, 2004.

[6] Colin Cartade, Rémy Malgouyres, Christian Mercat, and Chafik Samir. A simple and flexible mesh parameterization method. In *Proceedings of the International Conference on Combinatorial Image Analysis*, pages 157–167, 2011.

[7] Edwin Earl Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces.* PhD thesis, 1974.

[8] Zhonggui Chen, Ligang Liu, Zhengyue Zhang, and Guojin Wang. Surface parameterization via aligning optimal local flattening. In *Symposium on Solid and Physical Modeling*, pages 291–296, New York, NY, USA, 2007. ACM.

[9] P. Degener, J. Meseth, and R. Klein. An adaptable surface parameterization method. In *Proceedings of the International Meshing Roundtable*, pages 201–213, 2003.

[10] Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic parameterizations of surface meshes. In *Computer Graphics Forum*, volume 21, pages 209–218, 2002.

[11] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, pages 173–182, 1995.

[12] Michael Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231–250, 1997.

[13] Michael Floater and Kai Hormann. Parameterization of triangulations and unorganized points. In *Principles of Multiresolution in Geometric Modelling*, pages 127–154. Springer, 2001.

[14] Michael Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization*, pages 157–186, 2005.

[15] Anders Forsgren, Philip E. Gill, and Margaret H. Wright. Interior methods for nonlinear optimization. *SIAM Review*, 44(4):525–597, 2002.

[16] Xiao-Ming Fu, Yang Liu, and Baining Guo. Computing locally injective mappings by advanced mips. *ACM Transactions on Graphics*, 34(4):71:1–71:12, July 2015.

[17] Xianfeng Gu and Shing tung Yau. Computing conformal structures of surfaces. *Communications in Information and Systems*, 2:121–146, 2002.

[18] Steven Haker, Sigurd Angenent, Allen Tannenbaum, Ron Kikinis, Guillermo Sapiro, and Michael Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):181–189, 2000.

[19] Kai Hormann and Günther Greiner. MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*, pages 153–162. 2000.

[20] Kai Hormann, Bruno Lévy, and Alla Sheffer. Mesh parameterization: Theory and practice. In *ACM SIGGRAPH Courses*, 2007.

[21] Konstantine. Iourcha, Krishna. Nayak, and Zhou Hong. System and method for fixed-rate block-based image compression with inferred pixel values. US Patent 5956431, 1999.

[22] Stefan Jeschke, David Cline, and Peter Wonka. Rendering surface details with diffusion curves. In *ACM SIGGRAPH Asia*, pages 117:1–117:8, 2009.

[23] Jukka Jylänki. A thousand ways to pack the bin-a practical approach to two-dimensional rectangle bin packing. 2010.

[24] Liliya Kharevych, Boris Springborn, and Peter Schröder. Discrete conformal mappings via circle patterns. In *ACM SIGGRAPH Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[25] Ulf Labsik, Kai Hormann, and Günther Greiner. Using most isometric parameterizations for remeshing polygonal surfaces. In *Proceedings of Geometric Modeling and Processing*, pages 220–228, 2000.

[26] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3):362–371, 2002.

[27] J. Liesen, E. De Sturler, Y. Aydin, and C. Siefert. Preconditioners for indefinite linear systems arising in surface parameterization, 2001.

[28] Yaron Lipman. Bounded distortion mapping spaces for triangular meshes. *ACM Transactions on Graphics*, 31(4):108:1–108:13, 2012.

[29] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. A local/global approach to mesh parameterization. In *Symposium on Geometry Processing*, pages 1495–1504, 2008.

[30] Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. Spectral conformal parameterization. In *Symposium on Geometry Processing*, pages 1487–1494, 2008.

[31] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[32] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[33] Roi Poranne and Yaron Lipman. Provably good planar mappings. *ACM Transactions on Graphics*, 33(4):76:1–76:11, 2014.

[34] Budirijanto Purnomo, Jonathan D. Cohen, and Subodh Kumar. Seamless texture atlases. In *Symposium on Geometry Processing*, pages 65–74, 2004.

[35] Sam Roweis. Levenberg-marquardt optimization. *Notes, University Of Toronto*, 1996.

[36] Patrick David Sanan. *Geometric elasticity for graphics, simulation, and computation*. PhD thesis, Caltech, 2014.

[37] Pedro V. Sander, Steven J. Gortler, John Snyder, and Hugues Hoppe. Signal-specialized parametrization. In *Proceedings of the Eurographics Workshop on Rendering*, pages 87–98, 2002.

[38] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *ACM SIGGRAPH*, pages 409–416, 2001.

[39] Ryan Schmidt. Stroke parameterization. *Computer Graphics Forum*, 32(2):255–263, 2013.

[40] Ryan Schmidt, Cindy Grimm, and Brian Wyvill. Interactive decal compositing with discrete exponential maps. In *ACM SIGGRAPH*, pages 605–613, 2006.

[41] T. Schneider, K. Hormann, and M. Floater. Bijective composite mean value mappings. In *Symposium on Geometry Processing*, SGP '13, pages 137–146. Eurographics Association, 2013.

[42] John Schreiner, Arul Asirvatham, Emil Praun, and Hugues Hoppe. Inter-surface mapping. In *ACM SIGGRAPH*, pages 870–877, 2004.

[43] Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. Locally injective mappings. In *Symposium on Geometry Processing*, pages 125–135, 2013.

[44] Alla Sheffer and Eric de Struler. Surface parameterization for meshing by triangulation flattening. In *Proceedings of the International Meshing Roundtable*, pages 161–172, 2000.

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

[46] Alla Sheffer and John C. Hart. Seamster: Inconspicuous low-distortion texture seam layout. In *Proceedings of the IEEE Conference on Visualization*, pages 291–298, 2002.

[47] Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. Abf++: Fast and robust angle based flattening. *ACM Transactions on Graphics*, 24(2):311–330, 2005.

[48] Alla Sheffer, Emil Praun, and Kenneth Rose. Mesh parameterization methods and their applications. *Foundation and Trends in Computer Graphics and Vision*, 2(2):105–171, 2006.

[49] R. Vikram Pratap Singh and Anoop M. Namboodiri. Efficient texture mapping by homogeneous patch discovery. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, pages 37:1–37:8, New York, NY, USA, 2012. ACM.

[50] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*, pages 109–116, 2007.

[51] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proceedings of the Conference on Visualization*, pages 355–362, 2002.

[52] Boris Springborn, Peter Schröder, and Ulrich Pinkall. Conformal equivalence of triangle meshes. *ACM Transactions on Graphics*, 27(3):77:1–77:11, 2008.

[53] Geetika Tewari, John Snyder, Pedro V. Sander, Steven J. Gortler, and Hugues Hoppe. Signal-specialized parameterization for piecewise linear reconstruction. In *Symposium on Geometry Processing*, pages 55–64, 2004.

[54] William T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13(3):743–768, 1963.

[55] Ofir Weber and Denis Zorin. Locally injective parametrization with arbitrary fixed boundaries. *ACM Transactions on Graphics*, 33(4):75:1–75:12, 2014.

[56] Philip Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

[57] Cem Yuksel, John Keyser, and Donald H. House. Mesh colors. *ACM Transactions on Graphics*, 29(2):15:1–15:11, 2010.

[58] Rhaleb Zayer, Bruno Lévy, and Hans-Peter Seidel. Linear angle based parameterization. In *Symposium on Geometry Processing*, pages 135–141, 2007.

[59] Rhaleb Zayer, Christian Rössl, and Hans-Peter Seidel. Setting the boundary free: A composite approach to surface parameterization. In *Symposium on Geometry Processing*, pages 91–100, 2005.

[60] Rhaleb Zayer, Christian Rssl, and Hans peter Seidel. Variations on angle based flattening. In *Proceedings of Multiresolution in Geometric Modelling*, pages 285–296, 2003.

[61] Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics*, 24(1):1–27, January 2005.

[62] Eugene Zhang and Greg Turk. Visibility-guided simplification. In *Proceedings of the IEEE Conference on Visualization*, pages 267–274, 2002.

[63] Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In *Symposium on Geometry Processing*, pages 45–54, 2004.