Bijective Parameterization with Free Boundaries

Jason Smith[*]
Texas A&M University

Scott Schaefer[†]
Texas A&M University

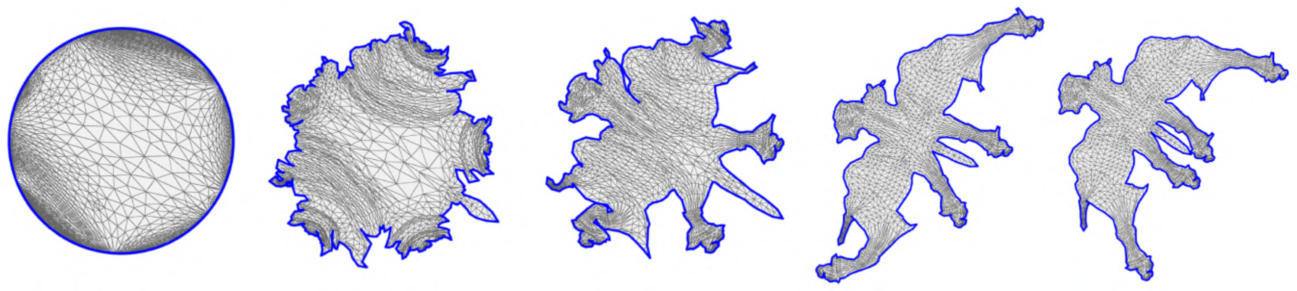

Figure 1: *Starting from Tutte's parameterization (left), our optimization generates a parameterization that minimizes distortion and guarantees a bijective map (right). We show intermediate stages of the optimization where, at every step, the parameterization is bijective. As opposed to previous techniques, we do not constrain the shape of the boundary, which is free to change shape to minimize distortion.*

Abstract

We present a fully automatic method for generating guaranteed bijective surface parameterizations from triangulated 3D surfaces partitioned into charts. We do so by using a distortion metric that prevents local folds of triangles in the parameterization and a barrier function that prevents intersection of the chart boundaries. In addition, we show how to modify the line search of an interior point method to directly compute the singularities of the distortion metric and barrier functions to maintain a bijective map. By using an isometric metric that is efficient to compute and a spatial hash to accelerate the evaluation and gradient of the barrier function for the boundary, we achieve fast optimization times. Unlike previous methods, we do not require the boundary be constrained by the user to a non-intersecting shape to guarantee a bijection, and the boundary of the parameterization is free to change shape during the optimization to minimize distortion.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

Keywords: parameterization, bijective mappings, free boundary

1 Introduction

Triangulated surfaces are ubiquitous in real-time graphics. However, geometry is only one aspect of how we perceive an object. Typically we annotate surfaces with other information such as color, lighting information, or even displacements to make the surface appear more realistic and provide details beyond the resolution of the vertices of the shape. These annotations are usually performed via texture mapping, which maps two-dimensional data onto the surface of a 3D object.

Texture mapping relies on a parameterization of a surface. Given a 3D surface, we partition the shape along a connected set of edges, which we refer to as seams, into contiguous sets of triangles called charts. Parameterization is the flattening of a chart to the two-dimensional domain and the seams of the charts in 3D become the boundaries of the flattened charts in 2D. For surfaces other than developable surfaces, this flattening introduces some 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.

While the quality of the parameterization is certainly important, the parameterization is of limited use for texture mapping unless it forms a bijective map between the 3D surface and the 2D texture covered by the charts. If the parameterization is not bijective, then a single point in the texture could map to multiple, disconnected regions of the surface. The result is that we cannot annotate such regions of the surface independently from one another rendering the parameterization useless for this application.

There are a few ways in which the parameterization could fail to be bijective. For example, triangles could "fold" or change orientation in the parameterization. A parameterization could also fail to be bijective if portions of the parameterization overlap. While only a handful of methods guarantee that the parameterization will be locally injective (no folds), almost none guarantee bijectivity without constraining the boundary to a non-intersecting curve. 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.

We propose a parameterization method that guarantees the parameterization forms a bijective map. Moreover, we do not constrain the boundary to some arbitrary shape but allow the optimization to modify the boundary to reduce distortion. We discuss the class of admissible distortion metrics that guarantee local injectivity and provide a form of isometric distortion that yields an expression

[*]email:agjdsmith09@gmail.com
[†]email:schaefer@cs.tamu.edu

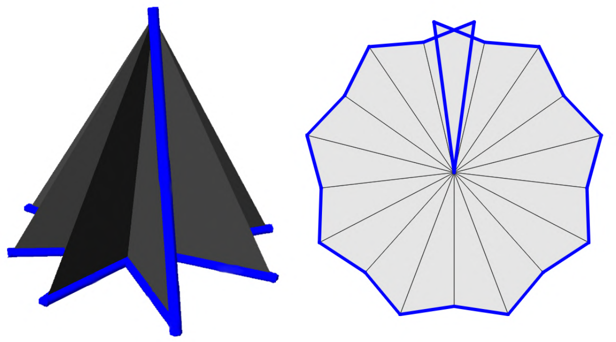

Figure 2: *A simple example of a wavy cone with the seam shown in blue (left) isometrically flattened without error to a parameterization without folded triangles (right) that is not a bijective map.*

whose value and gradient are simple to evaluate. In addition, we build a barrier term that prevents the parameterization from overlapping during optimization and discuss how to evaluate this function and its gradient efficiently. Finally, we discuss how to optimize such functions by explicitly computing the singularities in closed-form to guarantee we produce bijective maps.

2 Related Work

Surface parameterization is a well studied problem [Floater and Hormann 2005; Sheffer et al. 2006; Hormann et al. 2007]. While there are many parameterization methods, few guarantee bijectivity or even local injectivity.

One class of methods interleaves the segmentation process of dividing the surface into a set of charts with parameterization [Lévy et al. 2002; Zhou et al. 2004]. If the parameterization is not bijective, these methods split the charts to form smaller charts. Levy et al. [2002] split charts based on boundary intersections, while Zhou et al. [2004] 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 [Burley and Lacewell 2008; Yuksel et al. 2010]. Sorkine et al. [2002] 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. In contrast, we do not modify seams but always produce a bijective map.

Springborn et al. [2008] use a discrete conformal optimization for parameterization and require no initially defined seam. Their method guarantees local injectivity by performing edge flips or subdividing edges. Zhang et al. [2005] use "scaffold triangles" in a nonlinear minimization to guarantee the line search never causes the boundary to intersect. While similar in spirit to our method, the scaffold criteria is a sufficient condition to create bijective parameterizations but not necessary. Hence, this criteria slows down the optimization. Instead we compute the maximal safe step size in the optimization.

Some parameterization methods guarantee local injectivity through the use of distortion metrics that form barriers so that triangle flips cannot occur in the parameterization [Hormann and Greiner 2000; Sander et al. 2001; Degener et al. 2003; Schüller et al. 2013; Aigerman et al. 2014; Poranne and Lipman 2014]. Others [Lipman 2012; Aigerman et al. 2014; Poranne and Lipman 2014; Sanan 2014] bound the distortion of triangles to guarantee locally injective parameterizations. In addition, all of these methods can guarantee a

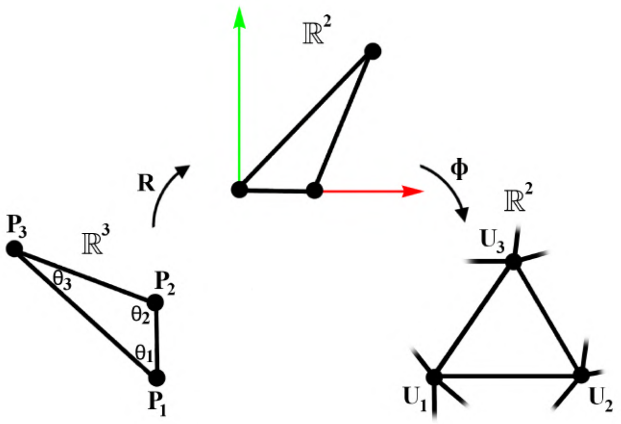

Figure 3: *Mapping a 3D triangle to 2D using a rigid transform R, which is then affinely mapped via ϕ to the parameterization.*

bijective map if the user constrains the boundary of the charts to form a non-intersecting curve.

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 [Tutte 1963; Floater 1997]. However, as shown in Figure 1, the distortion for these parameterizations can be severe. Weber et al. [2014] showed how to use such methods to create a bijective map between two non-intersecting boundaries by mapping to a common, convex domain. However, this method requires that the user specify a non-intersecting boundary curve and may still need to refine some triangles to guarantee a bijective map.

Angle based flattening [Sheffer and de Sturler 2001] 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 affect the distortion in that region. Later ABF++ [Sheffer et al. 2005] created a much faster form of angle base flattening using a hierarchical optimization procedure that guarantees local injectivity although not global bijectivity.

3 Bijective Maps

Our goal is to produce a bijective map from a 3D triangulated surface to a 2D domain that minimizes a distortion metric without constraining the boundary of the charts to fixed shapes. We begin with a 3D triangulated surface that is partitioned into a set of charts. For the moment, we will restrict ourselves to a single chart consisting of a set of triangles that are topologically equivalent to a disk. However, we will relax this assumption in Section 4.

A map for such a triangulated surface is bijective if two properties hold. The first is that the map is locally injective; that is, no triangles reverse orientation in the parameterization. This locally injective property is largely a function of the distortion energy, which we address in Section 3.1.

Unfortunately, such local injectivity does not guarantee a bijective map. For example, Figure 2 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. This

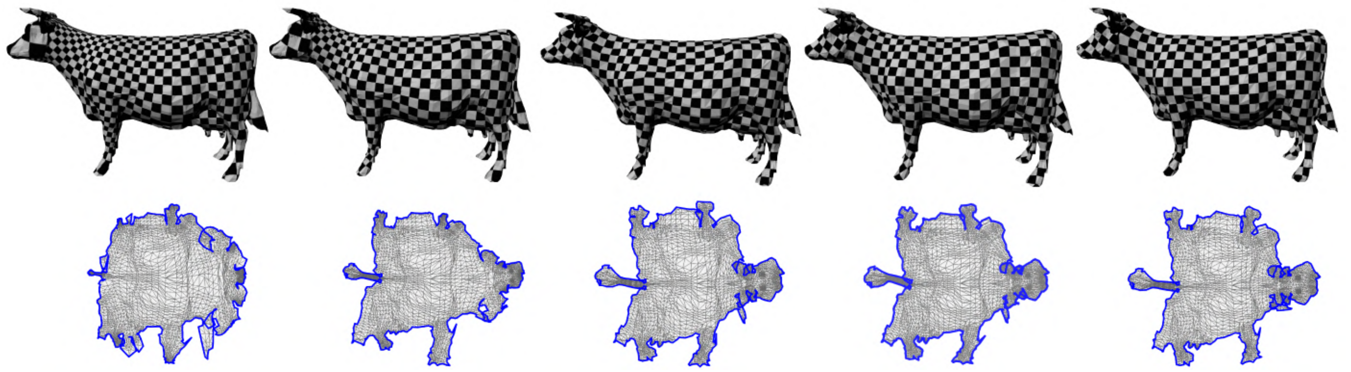

Figure 4: *Our optimization of E_D using different metrics to produce a locally injective parameterization. The top row shows a checkerboard mapped to the surface via the parameterization shown below. From left to right the metric used with timings in seconds: conformal 95.97, MIPS 3.42, maximal isometric 114.46, isometric 125.62, ours 1.29.*

property is much more difficult to maintain as it is not a quantity that can be locally computed. We discuss our strategy for enforcing this property in a computationally efficient manner in Section 3.2.

3.1 Local Injectivity

The majority of parameterization methods minimize some form of distortion E_D between the 3D surface and its 2D parameterization. Consider a 3D triangle shown in Figure 3 (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 (top), which then maps to the parameterized shape with vertices $U_1, U_2, U_3 \in \mathbb{R}^2$ via the affine transformation ϕ (right). It is the singular values of the linear portion of the matrix ϕ that defines the distortion of the triangle in the parameterization.

Let the linear portion of ϕ be given by the matrix

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

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. When $\sigma_1 = 0$ the triangle is degenerate.

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) [Lévy et al. 2002] 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 [Liu et al. 2008] measures isometric error by minimizing $(\sigma_1 - 1)^2 + (\sigma_2 - 1)^2$. However, both functions have the problem that they allow triangles to reverse orientation since the energy is finite for $\sigma_1 = 0$.

To create locally injective maps, we require that the triangles do not change orientation. A simple way of preventing these folds is to require a distortion measurement E_D that approaches ∞ as σ_1 approaches 0. Several functions already satisfy this property including conformal [Degener et al. 2003] $\left(\frac{\sigma_2}{\sigma_1}\right)$, MIPS [Hormann and Greiner 2000] $\left(\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1}{\sigma_2}\right)$, maximal isometric distortion [Sorkine et al. 2002] $\left(\max(\sigma_2, \frac{1}{\sigma_1})\right)$, and a form of isometric energy [Aigerman et al. 2014] $\sqrt{\sigma_2^2 + \sigma_1^{-2}}$. We compute the total distortion of these measurements by integrating the distortion 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 in our method, 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. We propose a computationally efficient form of isometric distortion

$$\sigma_1^2 + \sigma_2^2 + \sigma_1^{-2} + \sigma_2^{-2}, \tag{1}$$

which simplifies to

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

We should note that a similar form of symmetric stretch energy to Equation 1 was used for intersurface mapping [Schreiner et al. 2004] although the terms were weighted by different triangle areas.

The distortion metric in Equation 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

$$\sigma_1^2 + \sigma_2^2 = \begin{array}{l} \frac{|U_3 - U1|^2 |P_2 - P_1|^2 + |U_2 - U1|^2 |P_3 - P_1|^2}{4\Delta_P^2} \\ - \frac{((U_3 - U_1)\cdot(U_2 - U_1))((P_3 - P_1)\cdot(P_2 - P_1))}{2\Delta_P^2}. \end{array}$$

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

$$\left(1 + \frac{\Delta_P^2}{\Delta_U^2}\right)\left(\frac{|U_3 - U1|^2 |P_2 - P_1|^2 + |U_2 - U1|^2 |P_3 - P_1|^2}{4\Delta_P} - \frac{((U_3 - U_1)\cdot(U_2 - U_1))((P_3 - P_1)\cdot(P_2 - P_1))}{2\Delta_P}\right). \tag{3}$$

Most optimization methods also require the gradient of the error function, which is also easy to calculate for Equation 3. Without loss of generality, we only consider the partial derivative of Equation 3 with respect to a single vertex U_1. Given that Equation 3 is

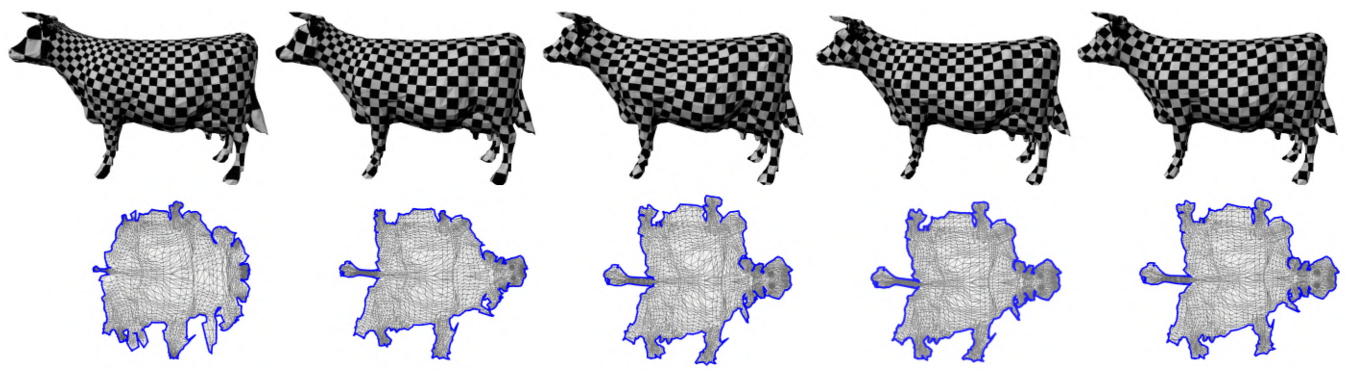

Figure 5: *Our optimization of $E_D + E_B$ using different metrics to produce a bijective parameterization. The top row shows a checkerboard mapped to the surface via the parameterization shown below. From left to right the metric used with timings in seconds: conformal 91.23, MIPS 19.5, maximal isometric 230.36, isometric 136.71, ours 3.63.*

the product of two terms, its derivative is given by the product rule and we need only consider the derivatives of each of the terms in the product. The partial derivative with respect to U_1 of the first term of Equation 3 corresponding to $\left(1 + \Delta_P^2 \Delta_U^{-2}\right)$ is

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

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 yields the cotan weights [Pinkall and Polthier 1993] that correspond to a discrete harmonic function where θ_i corresponds to angles of the 3D triangle shown in Figure 3

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

Moreover, quantities involving the P_i are constant in both the gradient and distortion metric in Equation 3 and can be precomputed. Figure 4 shows an example of using the optimization from Section 3.3 on a cow model composed of a single chart with different metrics that all produce locally injective parameterizations.

3.2 Bijective Maps with Free Boundaries

While the admissible metrics in Section 3.1 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 [Tutte 1963; Floater 1997; Lipman 2012; Schneider et al. 2013; Schüller et al. 2013; Poranne and Lipman 2014; Weber and Zorin 2014]. 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 3.1, the criteria to prevent intersections is not simply a local property of a triangle.

We enforce intersection free boundaries using 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). This function is 0

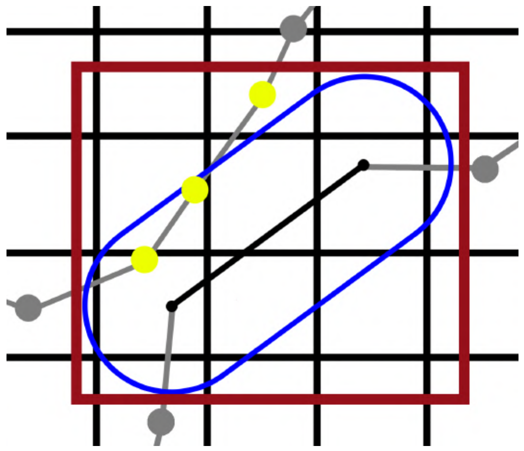

Figure 6: *Using a spatial hash to find potential boundary vertices (yellow) that intersect the support (blue) of our barrier function for an edge. The brown bounding box shows the query region for the highlighted edge.*

for $dist(U_1, U_2, U_i) > \epsilon$, smooth, and approaches ∞ as $dist(U_1, U_2, U_i)$ approaches 0. As ϵ shrinks, so too will the gaps between the boundary curves. In practice we choose ϵ to be the average length of the chart's 3D seam edges divided by 4. 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.

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, we choose a smooth function to avoid discontinuous changes in the gradient during optimization.

We can take advantage of the local support of our barrier function to accelerate the computation of E_B as well using a simple spatial hash as shown in Figure 6. Before evaluating E_B, we construct a

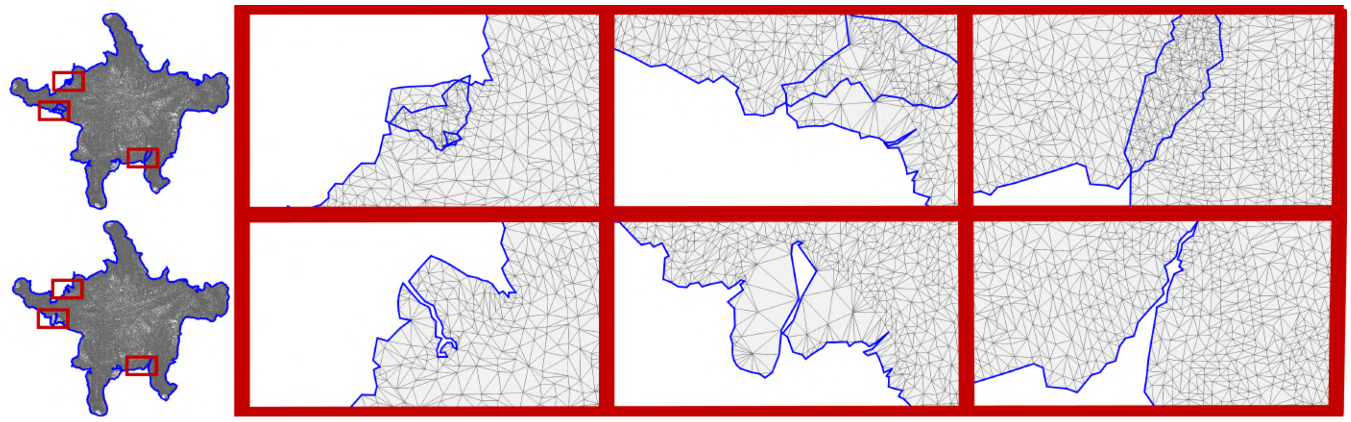

Figure 7: *Parameterization of a horse model without our boundary term E_B (top) and with (bottom). From left to right are zoom-ins on various sections of the parameterization that demonstrate the lack of bijectivity (top) versus the results of our bijective parameterization (bottom).*

grid ϵ larger than the current bounding box of the chart. For each boundary vertex, we insert that vertex into the grid. Now, given a boundary edge, we query all grid cells within the bounding box of the edge enlarged by ϵ discarding vertices that are part of this edge. These vertices 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, and similarly improves the speed of the resulting optimization.

Figure 5 shows an example of optimizing the same nonlinear metrics from Figure 4 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 counterparts despite requiring that the parameterizations form a bijective map at every stage of the optimization.

3.3 Optimization

The same distortion metrics that enforce local injectivity in Section 3.1 also yield a difficult optimization problem. There have been several approaches to optimizing these metrics from optimizing a single vertex at a time [Hormann and Greiner 2000] to using random search directions [Schreiner et al. 2004]. Lipman [2012] bounds distortion using inequality constraints and decompose the problem into maximal convex subsets, although such an approach requires repeated optimization for each subset.

Our approach is an interior point method [Forsgren et al. 2002] that guarantees that, at every step, the map remains a bijection. Therefore, we require an initial parameterization that guarantees a bijective map. Fortunately, Tutte's method [Tutte 1963] and Floater's parameterization [Floater 1997] both provide valid starting points, although the distortion 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 1.

We optimize the entire shape, as opposed to individual vertices, from the initial starting point using L-BFGS [Nocedal 1980] for its small memory usage, though we could use any Quasi-Newton method. Each of these methods begins 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 [Wolfe 1969]. However, being an interior point method, such solutions will not work without modification since interior point methods require direct computation of the singularities as part of the line search. Luckily all of the distortion metrics E_D we consider have the same set of singularities, namely when each individual triangle becomes degenerate. However, we must take care when computing the singularities as the number of singularities can be quite large. As shown in Equation 4, 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.

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.

$$\det \begin{pmatrix} (U_2 + V_2 t) - (U_1 + V_1 t) \\ (U_3 + V_3 t) - (U_1 + V_1 t) \end{pmatrix} = 0 \qquad (4)$$

Fortunately, Equation 4 is quadratic in t and the parameters that yield a singularity in E_D are simply given by the roots of this quadratic. Given that we are only concerned with searches in the positive direction, the smallest positive root gives the first singularity for this triangle. Computing the minimum parameter over all triangle gives the maximum value t_{max} for the line search with respect to E_D.

We also must compute the singularities of E_B with respect to our 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 4 gives the potential parameters associated with singularities for this combination of edge/vertex, which corresponds to when the edge and given vertex are all colinear. 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 colinear, the

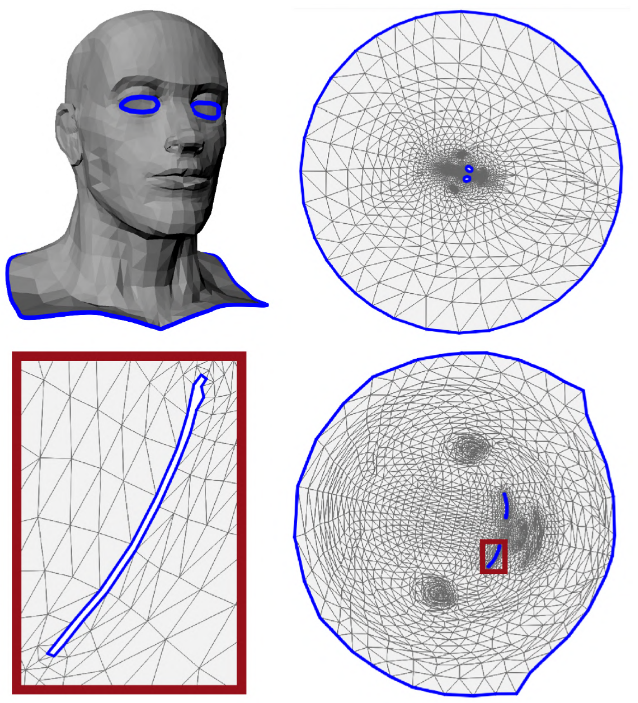

Figure 8: *Parameterization of a chart with multiple boundaries (top left) and the initial parameterization (top right) via Tutte's parameterization by arbitrarily triangulating the holes in the eyes. The bottom shows the results of our parameterization with the temporary triangles in the eyes removed and a zoom-in on one of these boundary curves.*

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 we must check. 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 the spatial hash in Section 3.2 to accelerate this computation. We assume that we have first computed the maximal parameter t_{max} for E_D. In the extremely unlikely case that $t_{max} = \infty$, we set it to a large positive value. For each boundary vertex U_j with search direction V_j, we insert the vertex into each grid cell intersecting the line segment between U_j and $U_j + V_j t_{max}$. Then, for each boundary edge with vertices U_1, U_2 and search direction vectors V_1, V_2, we query the grid for all boundary vertices that intersect the bounding box given by U_1, U_2, $U_1 + V_1 t_{max}$, $U_2 + V_2 t_{max}$. 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.

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
Head	10.081	75.904	10.097	33.357

Table 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.*

4 Results

Figure 1 shows our optimization in progress starting from Tutte's embedding on the left to a gradual unfolding of a highly distorted parameterized camel and ending in the parameterization on the right that minimizes our isometric distortion. At every step of the optimization, the parameterization remains a bijective map despite the highly intricate boundary interactions that occur during the optimization.

While we provide a particularly simple form of isometric distortion in Section 3.1, many different forms of distortion can be used with our optimization. Figure 4 shows an example where we used five different metrics within our optimization without adding the boundary term E_B; two of which measure conformal distortion followed by three measuring a form of isometric distortion. We also list the time in seconds our optimization takes with each metric on an Intel Core i7-3770k CPU running at 3.5 GHz. 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 4, none of these parameterizations are bijective since the boundary intersects itself in each example. Figure 5 shows the same optimization and metrics except we add our boundary 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.

Figure 7 shows another example with our distortion metric of a much more complex model of a horse with 20636 vertices. The top of the figure demonstrates the optimization without our boundary term. While there are no folded triangles, the shape folds backwards 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.

Our method is also not limited to charts that are topologically equivalent to a disc. We can easily incorporate charts with holes. The only complication required by our method is a valid starting point that contains no intersecting boundaries or folded triangles. By triangulating the holes, arbitrarily, Tutte's embedding or Floater's parameterization will produce such a valid starting configuration. We then delete the extra polygons and run our optimization as normal. We can even make a small performance optimization in this case as well. Each of the boundaries are independent from one another since interior triangles would have to collapse for the separate boundary curves to intersect. Therefore, during the optimization,

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 2: *The time taken in seconds for all of the results in Figure 11. Faces gives 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.*

we use a separate spatial hash for each boundary curve and compute them separately, which lowers the computational cost.

Figure 8 shows a challenging chart with three boundary curves that creates a significant amount of distortion. The upper right portion of the image shows Tutte's embedding using triangulated holes for the eyes (with the artificial triangles removed). After running our optimization (bottom), the holes remain intersection free and the parameterization forms a bijection.

Note that such a hole-filling strategy has been used previously in parameterizations [Hormann and Greiner 2000; Sanan 2014]. 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 triangles of the chart unnecessarily. In this case, the geometry of these holes and the arbitrary triangulation chosen affect the resulting parameterization despite having no corresponding surface triangles.

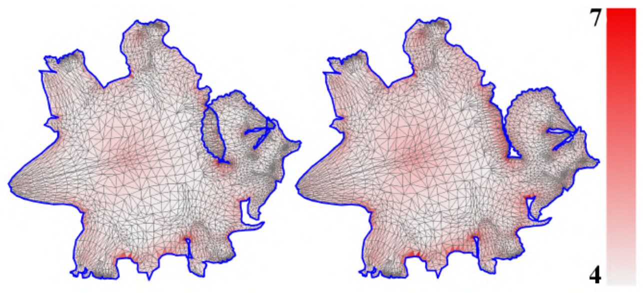

Figure 9: *Triceratops optimized with E_D on the left and $E_D + E_B$ color coded by our isometric error for each triangle.*

Table 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 isometric metric for the different models in the paper. Note that 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 9 shows a color map of the error of a parameterization without our boundary term (left) and with our boundary term (right). The image clearly shows that the error of the parameterization increases when forced to be bijective, particularly in the area of large overlap in this example. If the increase in error is too significant, it may be necessary to split the chart using an approach similar to [Lévy et al. 2002] or [Zhou et al. 2004].

Figure 11 shows a comparison of our method using our isometric metric and boundary term versus several popular parameterization algorithms including a spectral form of LSCM [Mullen et al. 2008], ABF++ [Sheffer et al. 2005], and ARAP parameterization [Liu et al. 2008]. 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 are not guaranteed to. In contrast, all of our results are guaranteed to be bijections.

Table 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.

5 Limitations and Future Work

Our method uses a non-convex energy function. Hence, we do not guarantee that we find the global minimum of the given distortion energy. With that said, we start with Tutte's parameterization, which contains significant distortion and is typically quite far from the minimum our optimization finds. Figure 10 demonstrates 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 we show several intermediate stages of the optimization before our optimization terminates in the second to the last image, which took 49.6 seconds. Obviously the parameterization is not the global minimum although the optimization made significant progress from the starting configuration. To verify we were stuck in a local minimum, 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. This example does bring up the issue of tolerances since we perform a numerical optimization. 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 with the exception that we create bijective maps.

In terms of future work, our method could greatly benefit from a better starting position. Unfortunately, few methods can currently guarantee a bijection without user intervention. In terms of speed, we would also like to explore multi-resolution methods [Hormann and Greiner 2000] for accelerating the parameterizations for very large charts consisting of hundreds of thousands of vertices. However, maintaining a bijective map as the multi-resolution structure expands, especially with respect to the boundary, may be difficult.

Finally, adding the requirement that the parameterization be seamless [Purnomo et al. 2004] would be useful. 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 [Bommes et al. 2009] and are difficult optimizations even without the bijective constraint.

Acknowledgements

We would like to thank Alla Sheffer for the models used in the paper. This work was supported by NSF Career award IIS 1148976.

References

AIGERMAN, N., PORANNE, R., AND LIPMAN, Y. 2014. Lifted bijections for low distortion surface mappings. *ACM Trans. Graph. 33*, 4, 69:1–69:12.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3, 77:1–77:10.

BURLEY, B., AND LACEWELL, D. 2008. Ptex: Per-face texture mapping for production rendering. In *Eurographics Symposium on Rendering*, 1155–1164.

DEGENER, P., MESETH, J., AND KLEIN, R. 2003. An adaptable surface parameterization method. In *Proceedings of the 12th International Meshing Roundtable*, 201–213.

FLOATER, M. S., AND HORMANN, K. 2005. Surface parameterization: a tutorial and survey. In *In Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization*, 157–186.

FLOATER, M. S. 1997. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14*, 231–250.

FORSGREN, A., GILL, P. E., AND WRIGHT, M. H. 2002. Interior methods for nonlinear optimization. *SIAM Review 44*, 4, 525–597.

HORMANN, K., AND GREINER, G. 2000. MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*. 153–162.

HORMANN, K., LÉVY, B., AND SHEFFER, A. 2007. Mesh parameterization: Theory and practice. In *ACM SIGGRAPH Courses*.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3, 362–371.

LIPMAN, Y. 2012. Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph. 31*, 4, 108:1–108:13.

LIU, L., ZHANG, L., XU, Y., GOTSMAN, C., AND GORTLER, S. J. 2008. A local/global approach to mesh parameterization. In *Symposium on Geometry Processing*, 1495–1504.

MULLEN, P., TONG, Y., ALLIEZ, P., AND DESBRUN, M. 2008. Spectral conformal parameterization. In *Symposium on Geometry Processing*, 1487–1494.

NOCEDAL, J. 1980. Updating quasi-newton matrices with limited storage. *Mathematics of Computation 35*, 151, pp. 773–782.

PINKALL, U., AND POLTHIER, K. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2*, 15–36.

PORANNE, R., AND LIPMAN, Y. 2014. Provably good planar mappings. *ACM Trans. Graph. 33*, 4, 76:1–76:11.

PURNOMO, B., COHEN, J. D., AND KUMAR, S. 2004. Seamless texture atlases. In *Symposium on Geometry Processing*, 65–74.

SANAN, P. D. 2014. *Geometric elasticity for graphics, simulation, and computation*. PhD thesis, Caltech.

SANDER, P. V., SNYDER, J., GORTLER, S. J., AND HOPPE, H. 2001. Texture mapping progressive meshes. In *ACM SIGGRAPH*, 409–416.

SCHNEIDER, T., HORMANN, K., AND FLOATER, M. S. 2013. Bijective composite mean value mappings. In *Symposium on Geometry Processing*, 137–146.

SCHREINER, J., ASIRVATHAM, A., PRAUN, E., AND HOPPE, H. 2004. Inter-surface mapping. In *ACM SIGGRAPH*, 870–877.

SCHÜLLER, C., KAVAN, L., PANOZZO, D., AND SORKINE-HORNUNG, O. 2013. Locally injective mappings. In *Symposium on Geometry Processing*, 125–135.

SHEFFER, A., AND DE STURLER, E. 2001. Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with Computers 17*, 3, 326–337.

SHEFFER, A., LÉVY, B., MOGILNITSKY, M., AND BOGOMYAKOV, A. 2005. Abf++: Fast and robust angle based flattening. *ACM Trans. Graph. 24*, 2, 311–330.

SHEFFER, A., PRAUN, E., AND ROSE, K. 2006. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis. 2*, 2, 105–171.

SORKINE, O., COHEN-OR, D., GOLDENTHAL, R., AND LISCHINSKI, D. 2002. Bounded-distortion piecewise mesh parameterization. In *Proceedings of the Conference on Visualization*, 355–362.

SPRINGBORN, B., SCHRÖDER, P., AND PINKALL, U. 2008. Conformal equivalence of triangle meshes. *ACM Trans. Graph. 27*, 3, 77:1–77:11.

TUTTE, W. T. 1963. How to draw a graph. *Proceedings of the London Mathematical Society 13*, 3, 743–768.

WEBER, O., AND ZORIN, D. 2014. Locally injective parametrization with arbitrary fixed boundaries. *ACM Trans. Graph. 33*, 4, 75:1–75:12.

WOLFE, P. 1969. Convergence conditions for ascent methods. *SIAM Review 11*, 2, 226–235.

YUKSEL, C., KEYSER, J., AND HOUSE, D. H. 2010. Mesh colors. *ACM Transactions on Graphics 29*, 2, 15:1–15:11.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2005. Feature-based surface parameterization and texture mapping. *ACM Trans. Graph. 24*, 1 (Jan.), 1–27.

ZHOU, K., SYNDER, J., GUO, B., AND SHUM, H.-Y. 2004. Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In *Symposium on Geometry Processing*, 45–54.

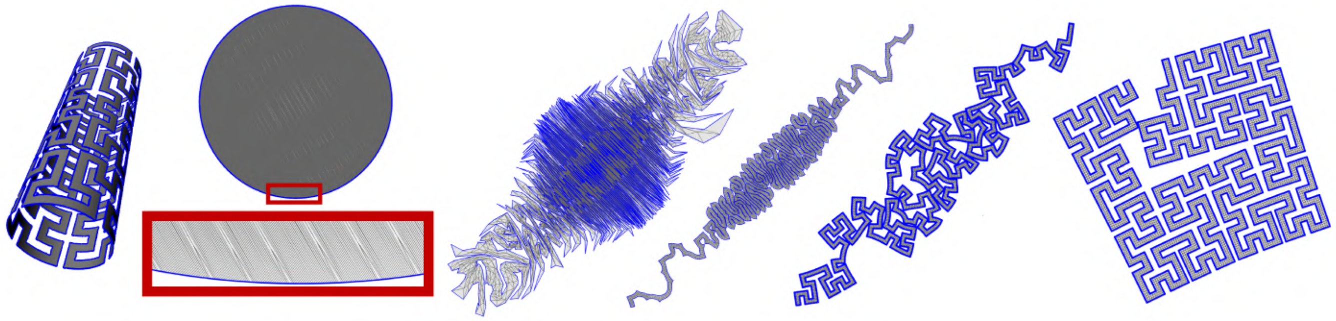

Figure 10: *A failure case for our method. From left to right: a space filling curve on the surface of a cylinder, Tutte's embedding with a zoom-in below to show the poor triangulation, two intermediate steps during the optimization, our result with default parameters taking 49.6 seconds with an average 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.*

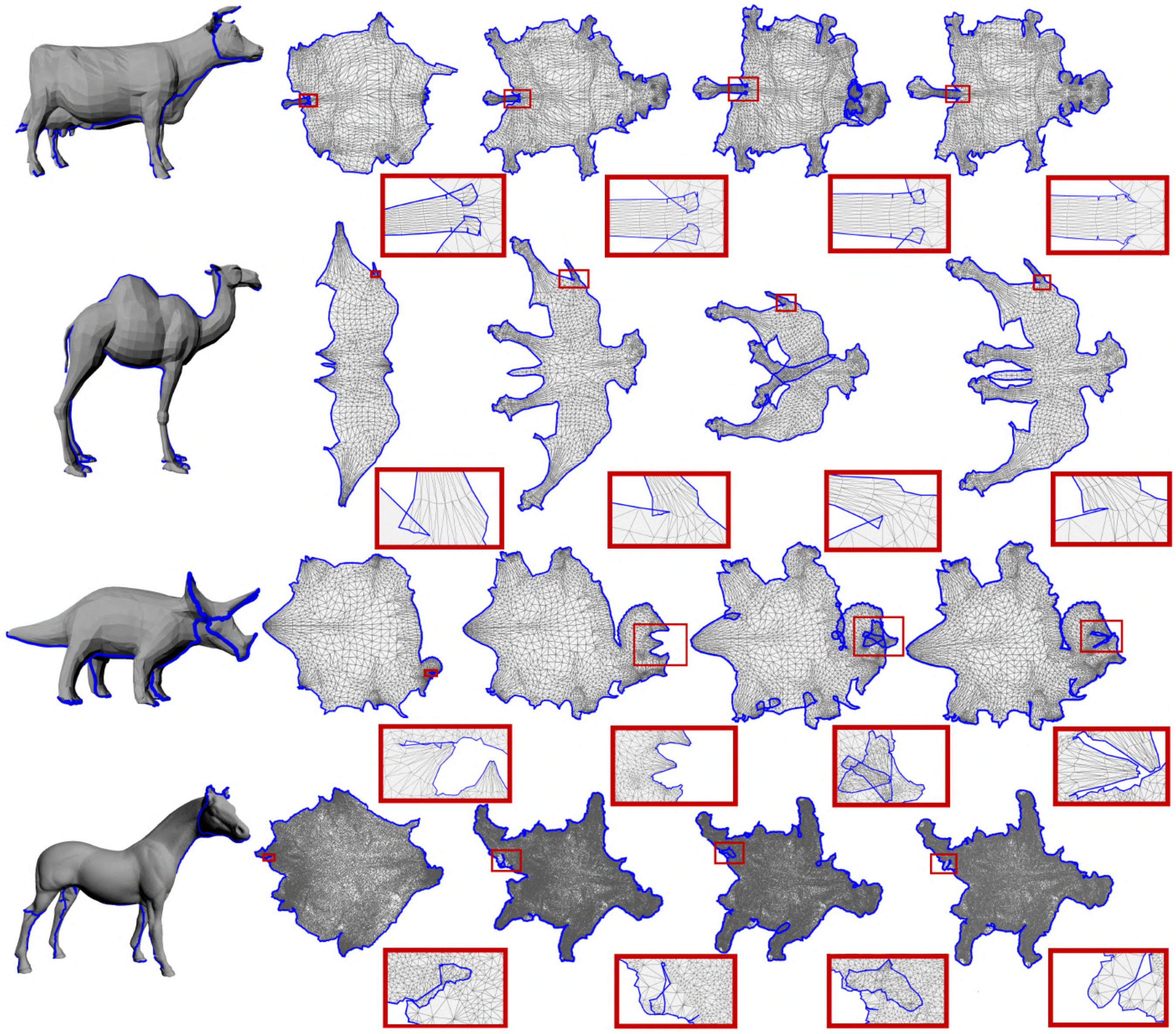

Figure 11: *A comparison of widely used parameterization methods applied to different models. The methods from left to right are: spectral conformal parameterization, ABF++, ARAP, and ours.*