Robust Field-aligned Global Parametrization

Ashish Myles[1*] Nico Pietroni[2†] Denis Zorin[1‡]
[1]New York University [2]ISTI, CNR, Italy

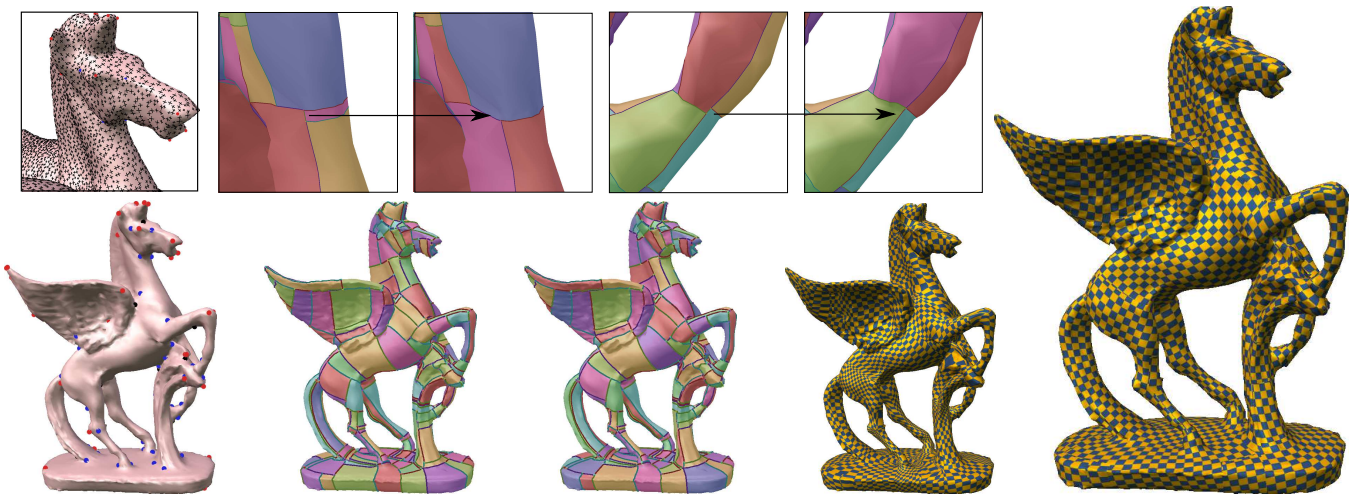

Figure 1: *Main steps of our construction, from left to right: initial field, feature-aligned inconsistent partition, collapse operations on zero chains, initial parametrization based on partition, final parametrization.*

Abstract

We present a robust method for computing locally bijective global parametrizations aligned with a given cross-field. The singularities of the parametrization in general agree with singularities of the field, except in a small number of cases when several additional cones need to be added in a controlled way. Parametric lines can be constrained to follow an arbitrary set of feature lines on the surface. Our method is based on constructing an initial quad patch partition using robust cross-field integral line tracing. This process is followed by an algorithm modifying the quad layout structure to ensure that consistent parametric lengths can be assigned to the edges. For most meshes, the layout modification algorithm does not add new singularities; a small number of singularities may be added to resolve an explicitly described set of layouts. We demonstrate that our algorithm succeeds on a test data set of over a hundred meshes.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—[Geometric algorithms, languages, and systems];

Keywords: geometric modeling, parametrization

Links: DL PDF

*e-mail: amyles@cs.nyu.edu
†e-mail: nico.pietroni@isti.cnr.it
‡e-mail: dzorin@cs.nyu.edu

1 Introduction

The goal of this paper is to present a robust global parametrization algorithm which produces results satisfying common requirements for arbitrary inputs (local bijectivity, alignment across seams, feature alignment).

We build on a commonly used set of techniques for finding a parametrization aligned to a guiding cross-field generated from surface features. These methods are typically based on minimizing a global quadratic energy measuring the deviation from the cross-field, subject to the constraint that the topological structure of the parametrization agrees with the structure of the field. Great strides were made in improving this type of methods as well as the quality and robustness of parametrization algorithms in general. Yet, even the state-of-the-art methods fail on a substantial fraction of meshes: for some methods the result may not be locally bijective, or, in the cases where nonlinear formulations are constructed to guarantee bijectivity, no feasible solution may be found.

In the latter case, failures may be due to one of three types of problems. The problem may be *connectivity-related*: connectivity of the mesh makes it impossible to map the mesh to the plane piecewise linearly and satisfy all requirements. It may be *algorithmic*: state-of-the-art efficient methods for imposing bijectivity constraints are based on restricting the space of solutions to a convex subspace, and a feasible parametrization may not be in that subspace. Finally, it may be *topological*: there are no global parametrizations with the topology specified by the field in that subspace. Our method includes solutions to all three types of problems: it can be viewed as a mesh refinement algorithm ensuring that the refined mesh is guaranteed to have a valid parametrization; as a consequence, the initial parametrization we obtain provides reference frames for convex algorithms that ensures that a feasible solution to the convex problem does exist. Finally, when topological problems are present in the field, the field is implicitly modified (including adding cones).

Our approach builds on the following well-known observation: *locally* for *any* non-singular field one can always construct a parametrization simply by following field integral lines

(parametrization by tracing). An appealing feature of this approach is that the parametrization is guaranteed to be perfectly aligned with the guiding fields and locally bijective, two difficult-to-achieve goals. This approach, however, does not directly extend to the general setting of global parametrizations with singularities. Moreover, for most fields no perfectly aligned global parametrization exists; furthermore, there may be no parametrization with *the same field topology* as the input feature field.

Instead, we use the tracing approach to build a quad partition of the surface. We then modify the partition to satisfy global parametrization constraints and finally parametrize quad by quad to obtain a locally bijective parametrization At each stage we guarantee (with limitations related to finite resolution) that for any nondegenerate input mesh, and a field satisfying weak constraints detailed in Section 4 a valid result is generated.

Overview. More specifically, our method proceeds as follows:

1. Build a quad patch partition of the mesh by tracing the field (Section 4, using robust field tracing, Section 6). This ensures feature alignment, and partitions the mesh into quads.
2. Modify this partition to be *globally consistent* (Section 7): specifically:
   (a) Determine a non-negative parametric edge length assignment for all patch sides; some edges may be forced to have zero length by the field structure (Section 7.1).
   (b) Eliminate degenerate quads from the partition using a partition simplification algorithm, which, except for a well-defined set of cases, eliminates all degenerate quads (Section 7.2).
   (c) If any degenerate quads are left, add singularities to the parametrization to eliminate the remaining degeneracies in edge assignment.
3. Obtain an initial parametrization by mapping each quad to a rectangle of the size determined by the edge length.

A final parametrization is computed by solving an optimization problem with convex constraints which is always guaranteed to have a solution.

We demonstrate the robustness of our method on a test data set of 114 models, which includes all nontrivially different manifold meshes from commonly used repositories (in greater detail the data set is described in Section 8). Our method succeeds in producing a valid initial parametrization for 100% of the models, and an optimized parametrization is obtained for all models except one. The resulting global parametrization can be used either on its own if the application does not require integer parametric cone positions (e.g., constructing T-spline based approximations), or requires minimal rounding (tiling the surface with textures). Alternatively, it can be a robust starting point for a recent method [Bommes et al. 2013].

2 Related work

A broad overview of quadrangulation and field-guided techniques is given in [Bommes et al. 2012]; we briefly mention the most important works most closely related to ours.

Previous work on field-aligned parametrization/remeshing addressed the problem of constructing a global parametrization from the field in several ways. In [Alliez et al. 2003] and [Ray et al. 2006] singularities emerge as a part of the remeshing process, and their locations and numbers cannot be controlled directly, and do not fully agree with the field. [Bommes et al. 2009] described a stiffening method that modifies the energy used to compute parametrization to eliminate flipped triangles. More recently, [Bommes et al. 2013] proposed a method based on introducing convex constraints ensuring bijectivity. While the method of [Lipman 2012] was not applied to global parametrization, it easily extends to the global parametrization setting as we show in Section 8. A number of methods starting from [Lee et al. 1998], are based on constructing coarse parametric domains by simplification or as a Morse graph of a function [Dong et al. 2006]. While these methods do not provide formal guarantees, they have structure that allows, in principle, a robust realization. Field alignment and cone placement optimization can be integrated with these techniques in a more limited way, compared to more recent methods. Conformal methods [Sheffer and de Sturler 2001; Kharevych et al. 2006; Ben-Chen et al. 2008; Springborn et al. 2008] either do not guarantee local bijectivity, or guarantee it only if a feasible solution of a constrained nonlinear optimization problem is found.

The algorithms described in this paper are drawing on two other important sources: motorcycle graph constructions [Eppstein et al. 2008; Gunpinar et al. 2014] and robust field tracing using edge maps [Jadhav et al. 2012]. On surfaces, so far, motorcycle graph constructions were applied to quadrangulations, not cross-fields. As we show in Section 4, some additional problems need to be handled in the field case.

Our algorithm is agnostic about the origin of the fields: in particular, it works with any algorithm generating piecewise constant fields on faces ([Bommes et al. 2009; Myles and Zorin 2012; Myles and Zorin 2013]); vertex-based fields can also be adapted (e.g., [Knöppel et al. 2013]). All of these are converted to our representation identical to the one used in concurrent work [Ray and Sokolov 2013]. The relationship of our algorithms to those in [Ray and Sokolov 2013] is discussed in Section 6.

There is a wealth of related work on tracing vector fields in the plane and on surfaces; recently [Szymczak and Zhang 2012] demonstrated how piecewise constant fields can be handled robustly; however, piecewise constant fields, due to existence of merging and splitting integral lines, are not suitable for our purposes.

Most commonly, for planar meshes, piecewise linear interpolation of vertex-based fields is used (e.g., considered in detail [Tricoche 2002]) As demonstrated in [Zhang et al. 2006] linear interpolation cannot be directly applied to fields on 3D meshes. We also note that Singularities of piecewise linear fields are generically on faces and are simple [Knöppel et al. 2013], although arbitrary high-order singularities can be modeled at vertices [Tricoche et al. 2000].

To have a complete control of singularity placement, we use linear interpolation in the space of angles similar to [Li et al. 2006a]. However, we use a different representation as it is easier to trace it efficiently with guarantees on integral lines.

We should also mention that quad partitions we construct are T-meshes, and in principle can be optimized using techniques similar to [Myles et al. 2010]. Our cone insertion procedure is related to the one used in [Li et al. 2006b].

3 Background

Aligned seamless parametrizations. Our goal is to construct seamless global parametrizations of the type used in [Bommes et al. 2009]. We build a quad partition first, and the initial parametrization is defined per quad. In this case, the conditions can be formulated in a particularly transparent form, similar to [Tong et al. 2006; Dong et al. 2006]: for each quad Q we define a locally bijective map to the (u, v)-domain; on the boundary between quads Q and Q', parametric positions $\mathbf{q}$ satisfy

$$\mathbf{q}_2 = r\mathbf{q}_1 + \mathbf{t}, \tag{1}$$

for any two points $\mathbf{p}_1$ $\mathbf{p}_2$ on the common boundary, where $r = R_{k\pi/2}$ is a rotation by a multiple of $\pi/2$, $\mathbf{q}_i$ are parametric positions

with respect to Q and Q' respectively, and $\mathbf{t}$ is a *translation* between quads in the parametric domain. (Figure 2).

We emphasize that we do not consider the details of the integer-grid layout problem in the paper, which requires the additional condition that $\mathbf{t}$ is an integer vector, although, as explained in Section 7, our method allows to obtain fine quadrangulations, or serve as a starting point for the mixed-integer optimization of [Bommes et al. 2013].

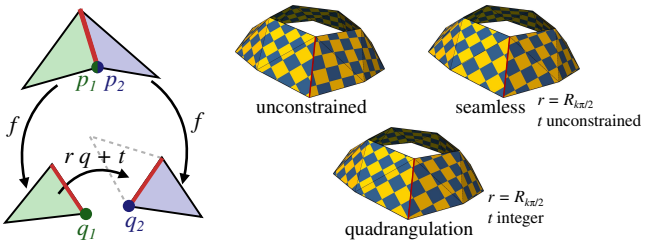

Figure 2: *Seamless parametrization*

Cross-fields. We are interested in parametrizations aligned with a *cross-field*. A cross-field is an assignment, to each point of a mesh, of a quadruple of unit vectors $\mathbf{u}, R_{\pi/2}\mathbf{u}, -\mathbf{u}, R_{\pi/2}\mathbf{u}$, which are considered equivalent if related by a cyclic permutation. Continuous vector fields on surfaces (more precisely, Lipschitz) have an important property: for each nonsingular point there is a unique integral line passing through it. Cross-fields have a similar property, except there are two orthogonal lines at each point. Moreover, the field can be locally "straightened" i.e. there is always a neighborhood that can be mapped to the plane so that the field integral lines become parallel straight lines.

This property yields a direct correspondence between local non-singular parametrizations and cross-fields: parametrization can be constructed uniquely from the field, up to a monotonic change of variables along axes, by "field straightening". In this case, integral lines of the field coincide with parametric lines.

However, globally this is in general, impossible, and the parametrization has to minimize a deviation from the field.

Obstacles to field-aligned parametrization. There are two types of singular features in a field that prevent us from converting a set of local parametrizations to a global one: *singular points* and *limit cycles.*

Singular points. Singular points of a direction (unit vector) field or a cross-field can have an arbitrarily complex structure, for vector fields formally discussed in, e.g., [Andronov et al. 1973]. We follow the definitions of [Tricoche 2002], and use the characterization of sectors introduced in [Li et al. 2006a].

If there are no integral lines passing through the singularity C, there is a neighborhood of $\mathbf{p}$ in which all integral lines are closed curves around $\mathbf{p}$. In this case, the singularity is a *center*. Otherwise, a small neighborhood of the singularity bounded by a closed curve C, can be decomposed into *sectors*, each bounded by C and two integral lines $S(t)$ and $S'(t)$ (*separatrices*) satisfying the equation $dS/dt = \mathbf{u}(S(t))$. The field direction defines an orientation on the curves S. Let s be the angular parameter along C, increasing counterclockwise. The vector field restricted to C can be also viewed as a function $\mathbf{v}(s)$, which can be represented by an angle $\alpha(s)$ with respect to the direction from $C(s)$ to the singularity. It depends on the choice of C, but we consider the limit values as the C shrinks to $\mathbf{p}$, and view limit $\alpha(s)$ as the directional variation of the field at the singularity.

Three cases are possible (Figure 3).

1. *Hyperbolic* sector: any integral line passing through the sector, other than S and S', enters and leaves the sector and does not pass through $\mathbf{p}$. For hyperbolic sectors, the rotation rate $\alpha'(s)$ is *negative* (the direction rotates clockwise).
2. *Parabolic* sector: All integral lines in the sector, including S and S' approach $\mathbf{p}$, as $t \to \infty$ or $-\infty$. For parabolic sectors, the rate of rotation is *zero*.
3. *Elliptic sector*: S and S' are parts of the same integral line starting and ending at $\mathbf{p}$, and any integral line in the sector approaches $\mathbf{p}$ for both $t \to \infty$ and $-\infty$. For elliptic sectors, the rate of rotation $\alpha'(s)$ is *positive*.

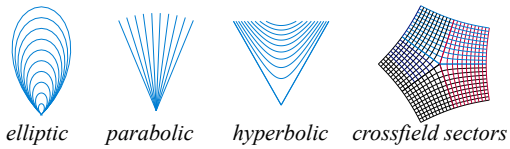

Figure 3: *Left: Three types of sectors at singularities. Right: A cross-field has overlapping sectors.*

Sectors for cross-fields. Observe that if we introduce a single cut at a singularity along a separatrix of a cross-field, the cross-field on the cut neighborhood can be separated into two vector fields, yielding two sets of overlapping sectors (Figure 3, right). Note that the rate of rotation of a cross-field coincides with the rate of rotation of the two vector fields into which it can be locally split, and the integral of α' is related to the index $I(\mathbf{p})$ of the singularity by

$$\int_C \alpha'(s)ds = 2\pi(I(\mathbf{p}) - 1)$$

As all separatrices starting at singularities become patch edges in our construction, a crucial requirement is that *the cross-field singularities have a finite number of separatrices*, i.e. no parabolic or elliptic sectors are allowed. As any singularity of index 1 or higher has to have elliptic sectors, we only accept fields with singularities of index < 1 as input.

Limit cycles. In addition to singular points, a vector or a cross-field can have *limit cycles* (Figure 4). A *limit cycle* C is a closed integral line which is a limit set of a set of other field lines.

Limit cycles are a major fundamental obstacle for constructing a global parametrization by tracing a field, as near a limit cycle the parametrization will be singular. Isolated limit cycles are *stable*: one can construct vector fields for which a significant perturbation is needed to obtain a field without cycles; in other words, not every field on the surface is close to a gradient field of a global parametrization. Many of the zero quad chains discussed in Section 7 appear due to limit cycles in the field.

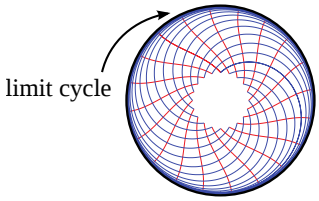

Figure 4: *A limit cycle attracts an infinite number of integral lines.*

Finally, we should mention that limit cycles are not necessary for a field to be not consistent with a global parametrization. For example, while the spiral strip shape from Figure 5 does have a limit cycle at the bottom, the first torus field does not have limit cycles, yet, it is not consistent with any seamless global parametrization. (See the supplementary document for a proof.)

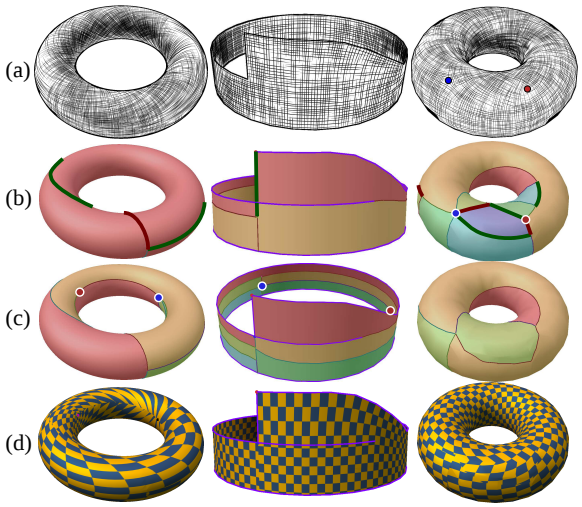

Figure 5: *(a) Three fields not consistent with any parametrization: a torus with $\pi/2$ rotation in the field; a spiral strip from [Myles and Zorin 2013], with field following the strip; and a torus with a single 3-5 singularity pair. (b) Traced T-mesh with zero-edges highlighted. (c) Resolved T-mesh, using 2-6 pair insertion for the first two and a singularity pair collapse for the third; (d) Optimized bijective parametrization using the quasi-conformal constraints of [Lipman 2012].*

4 Quad partition construction

Overview. The input to our partition construction algorithm is a mesh with a cross-field defined using a representation described in Section 5. This representation can be generated from a cross-field defined per face [Ray et al. 2008; Bommes et al. 2009], and a set of feature edges. We impose no restrictions on the choice of feature edges other than requiring boundaries curves to be marked as such; but we refine the mesh so that any triangle has at most one feature edge. We do require the field to satisfy three requirements: (1) on a triangle with a feature edge, the field is aligned to the edge; (2) no field singularity has index 1 or greater; and (3) the field rotation rate $\alpha'(s)$ between two sequential feature curves around a vertex is strictly negative. The first requirement insures that feature edges are integral lines; the latter two guarantee that the field can be converted to a field with a finite number of integral lines through each point.

The first step in our algorithm is constructing a *quad partition* of the mesh. More precisely, we define the connectivity of the partition as follows: a partition is a 2D complex, with edges formed by p.w. linear paths on the mesh, and for which each face is a polygon with 4 or more vertices; exactly 4 of these vertices are marked as face corners. We view all faces as quads, with non-corner vertices considered to be T-joints. The chain of edges connecting two corners of a face is a *T-edge* of the face (Figure 6, left).

We describe the algorithm for constructing the partition by tracing a cross-field. The algorithm requires that the discrete p.w. linear integral lines constructed from the field have a set of properties matching the properties of integral lines of a Lipschitz field, with singularities of index less than 1. Specifically we assume that:

- Singular points have a finite number of integral lines;
- Each regular point has exactly two orthogonal integral lines passing through it;
- Integral lines intersect either orthogonally, or at singularities.

(We say that two integral lines of a cross-field *intersect orthogonally* if they are tangent to orthogonal cross-field directions at their common point.)

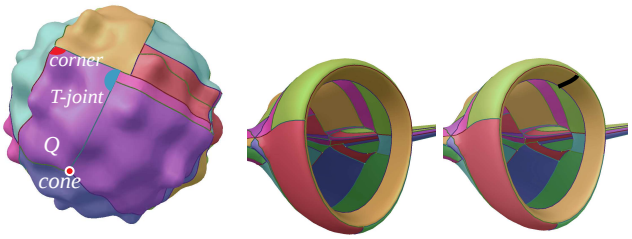

Figure 6: *Left: Example of a quad partition with notation; a T-joint vertex with respect to face Q is shown in blue, a corner of Q in red. Right: Splitting a cylindrical face by tracing perpendicular to its boundary (black curve).*

These properties are guaranteed by the construction of Section 5, so we assume that these hold in our algorithm.

Motorcycle graph algorithm. To build the partition, we use the *motorcycle graph* algorithm [Eppstein and Erickson 1999] on surfaces with some modifications to ensure termination and correctness of the resulting partition. The idea of the algorithm is to trace all integral lines emanating from singularities at equal speed in parallel, until each line terminates either meeting either another line, another singularity or a boundary/feature line. A detailed pseudocode of the algorithm is included in the supplementary document.

Termination guarantees. In this simplest form, the algorithm is not guaranteed to terminate: for example, if the cross-field has a limit cycle, and no integral line generated by the algorithm crosses this cycle orthogonally, but there is a traced line that converges to the cycle. While it is possible to set up such configurations artificially, in practice these were *never* observed. To ensure that the algorithm always completes, a more complex procedure is needed; it is described in the supplement.

The resulting graph partitions the surface into connected domains (faces); in the interior of each domain, the field is not singular. The boundary of each face consists of integral line segments. We say that two segments sharing an endpoint v form a corner of a face if they meet orthogonally; the corner is *convex* if there are no integral lines in the interior of the face passing through v.

Proposition 1. *All faces of a non-empty motorcycle graph are either quads or cylinders.*

The proof of this proposition can be found in the supplemental document.

If a cylinder partition is present, normally, tracing a field line perpendicular to the boundary (Figure 6, right) reduces it to a quad face; this may not be sufficient (e.g., if the cylinder contains a limit cycle), in which case cell partition described in the supplement needs to be used.

Refinement. Once the quad partition is obtained, the mesh is refined by inserting all edges of traced lines and triangulating faces. In addition, we ensure that each quad face has a 3-connected mesh, so that bijective parametrizations of the faces over a parametric-domain rectangles can be obtained using Tutte's maps. To achieve this, each edge with two vertices on the boundary is subdivided.

Once refinement is done, we can guarantee that the resulting mesh does have a locally bijective global seamless parametrization, perfectly aligned with feature lines and approximately with the field, *if* consistency conditions, discussed in the next section, are satisfied.

5 Traceable fields on meshes

As discussed in Section 4, the essential features of integral line tracing include: (1) a finite number of integral lines starting at each singularity; (2) exactly two orthogonal lines passing through each

regular point; and (3) distinct integral lines intersecting only at singularities or orthogonally. *We need to retain these properties in the discrete case.*

Our goal in this section is to define a field representation that ensures these properties are retained. While in the plane this can be easily achieved by linear interpolation of a vertex-based field, it cannot be easily applied to tangent vectors on meshes [Zhang et al. 2006].

In addition to the considerations above, we choose a field representation with the following properties: (4) arbitrary hyperbolic singularities are allowed for maximal flexibility in the resulting parametrization structure, and (5) all singularities are at vertices. These properties ensure compatibility with well-developed techniques for field generation [Bommes et al. 2009; Myles and Zorin 2012; Myles and Zorin 2013], which generate singularities at vertices, and place no restrictions on the structure of the resulting parametrization.

We define the cross-field per-vertex *intrinsically* as described below, with a single chart-domain direction at regular vertices, and with an initial field direction and rotation rate at singular vertices. We show how field values in charts can be translated to field values on 3D mesh triangles.

Vertex charts. Using a set of charts at vertices, we can define a per-vertex tangent space intrinsically. The chart maps can be used to map the field back to the mesh, maintaining its topological properties. However, as we will see, a single chart direction at vertices corresponds to a varying set of directions on triangles.

We define chart maps c_i for each vertex 1-neighborhood N_v. Pick an edge e incident at the vertex v, and let f_v be the isometric map of the 1-neighborhood cut along the edge e to the plane, mapping v to $[0,0]$ and e to $[\ell,0]$ where $\ell=|e|$. Let $g_v(z)=z^{2\pi/\Theta}$, where Θ is the total angle at v, and z is the complex coordinate in the plane. This map "closes the gap" left by unfolding the cut 1-neighborhood. The composition $c_v=(g_v\circ f_v)^{-1}$ is a continuous map; we use the inverse map from the chart to the surface, as we want to transfer vectors from the parametric domain to the surface. We note that the maps c_v form a conformal atlas.

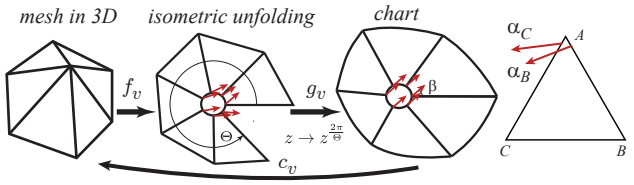

Figure 7: *Right: Chart map and its effect on a field at a vertex. Left: Field representation notation.*

Mapping fields from charts to the mesh. Suppose we have a smooth field defined on a chart $C_v=c_v^{-1}(N_v)$; then, as f_v is an isometry, we can simply look at the effects of $g_v^{-1}=z^{\Theta/2\pi}=z^q$. This map is smooth everywhere except at v.

We are interested in the behavior of the vector field near v, so without the loss of generality we assume the vector field is constant and has direction β in C_v, represented by complex number $e^{i\beta}$.

On a circle of radius ϵ, at a point with polar coordinates (ϵ,γ) the image of the vector is $\epsilon q e^{i(q-1)\gamma}e^{i\beta}$, i.e. for $q\neq 1$ (for a non-flat surface) has direction $e^{i((q-1)\gamma+\beta)}$, independent of ϵ. In other words,

At a regular vertex, on the mesh, the chart map c_v maps a single vector at a vertex to a directionally varying field at a vertex, rotating at the constant rate $q-1=\frac{\Theta-2\pi}{2\pi}$.

In the case of a singularity with a finite number n of integral lines terminating there, the field at the origin in C_v is undefined, but can be described by a rate of rotation of the direction around this vertex and a direction for an arbitrarily chosen edge.

For a valence-n singularity with uniform field rotation around it at the rate $(1-n/4)$, the image of the field under g_v is also a uniform-rotation field similar to the regular case, but with a different rotation rate.

Thus, at every vertex, we specify the field valence (4 for regular vertices) and a direction. For singularities, the direction is treated as the initial direction for rotation.

This approach can be easily extended to a more general type of singularities with arbitrarily chosen n parametric lines; in our code, we use uniform-rotation rate fields at singularities with no feature lines, and the more general formulas for points where multiple feature lines meet (see supplementary document).

Discrete field representation. The derivation above yields a way to approximate a field directly in terms of per-triangle quantities. The rotation of the field at vertices can be precisely represented by specifying two angles at each vertex of the triangle and assuming constant rotation speed for directions between. Thus, for each edge, we define two unit vectors in the planes of incident triangles such that if the two triangles are unfolded, the vectors coincide (Figure 7). In each triangle, we choose a reference direction and represent all directions encoding the field by angles with respect to the reference direction. The per-triangle data for the field consists of 6 angles denoted α_B, α_C, β_A, β_C, γ_A, γ_B, one for each (vertex,edge) pair. Suppose the barycentric coordinates of a point $\mathbf{p}$ are (u,v,w), where $u+v+w=1$. Define $r,s,t\in[0,1]$ to be angles between one of the triangle sides at each vertex and the line connecting the vertex to $\mathbf{p}$, divided by the triangle angle at the vertex. Then the angle is interpolated via

$$\theta(\mathbf{p})=((1-r)\alpha_B+r\alpha_C)u+(s\beta_A+(1-s)\beta_C)v+((1-t)\gamma_A+t\gamma_B)w$$

(angle-linear interpolation at vertices, and position-linear across the face). One may verify that $\theta(\mathbf{p})$ is a convex combination of the six coefficient angles, linearly interpolates along edges, and linearly interpolates in angle at vertices. $\theta(\mathbf{p})$ reduces to barycentric interpolation if the two per-edge vectors are equal at each vertex.

6 Resolving field topology on a triangle

The most straightforward approach to tracing the continuous field defined in the previous section is to use a standard field integration method, e.g. Runge-Kutta. For sufficiently small steps, the field lines can be approximated arbitrarily well. As long as there are no tangent integral lines, if two integral lines of a smooth field do not intersect, their sufficiently fine piecewise linear approximations do not intersect. However, for our purposes, this approach is not entirely adequate: we aim to cut out quad domains on the surface which then will be parametrized, inserting multiple edges into each triangle. Refining the mesh may be expensive especially for large meshes; it is desirable to minimize the number of line segments used to approximate an integral line in a single triangle.

Overview. Our strategy is to construct an *edge map* [Jadhav et al. 2012]. An edge map $E(\mathbf{p})$ is a map from the triangle boundary to itself, such that if an integral line starts/respectively ends at $\mathbf{p}$, it ends/starts at $E(\mathbf{p})$. $\mathbf{p}=E(\mathbf{p})$ only at tangency points. A single step of tracing a field through a single triangle reduces to evaluating the edge map.

The map is constructed by splitting each triangle into *strips* capturing the field topology inside the triangle. Informally, each strip has two opposite segments where all lines enter and exit. The other two sides (one of which can be degenerate, i.e. a point) are field

lines. Before resolving the topology of the field on a triangle, we preprocess the mesh, refining triangles at cones, so that at most one integral line passing through a cone is contained in each triangle.

As there are no singularities inside the triangle, the overall structure of the vector field on the triangle is relatively simple: any integral line entering the triangle has to leave it at another point of the boundary, and by our assumption about vertices (no more than one integral line), no integral lines intersect inside the triangle. Lines entering at arbitrarily close locations on the boundary may exit at very different points (Figure 8).

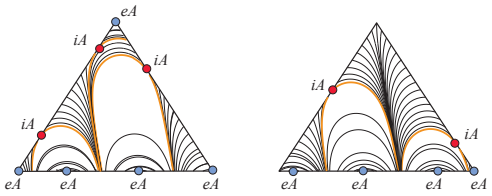

Figure 8: *Field topology on a triangle, showing two main patterns of aligned and transversal divisions.*

The *topology resolution* process constructs a *partition* for each triangle by sequentially chopping off strips, defined precisely below, along with the field approximation with piecewise linear field lines.

A triangle partition is a small mesh, with strips as faces. To distinguish from the triangle edges, we call the edges of the partition *divisions*, indicating that most of them are divisions of the triangle border into parts on which the field points either inside, outside or tangentially. We add extra divisions that cut the triangle. Geometrically, divisions can be points, line segments, or two or three joined segments on two sides of a triangle, connecting (possibly coinciding) points on the triangle border.

A division can be *transversal* (denoted T) or *aligned*. On a transversal division, the field is not parallel to the division direction and points either to the interior or exterior of the triangle. An aligned division is a part of an integral line of the approximate field. It can be a single point (vertex or non-vertex) separating *in*-transversal (iT) and *out*-transversal (oT) divisions on the boundary of the triangle or a triangle edge aligned with the field. All interior divisions, inserted to split off a strip, are aligned divisions.

Each *strip* of a triangle partition has exactly two transversal divisions, one *in* and one *out*. The field lines of the approximate field start on the *in*-transversal division and exit on *out*; the remaining edges of a strip are field lines or points on the triangle boundary.

Algorithm initialization. Initially, the triangle partition contains a single face: the whole triangle. The initial set of divisions is computed by splitting the triangle boundary at all points where the direction of the field changes with respect to the edge: some of the vertices, or points where the field is tangent. During the algorithm we maintain an *active face*; the partition at any given time contains a single active face and a collection of strips already split off. Note that initially, the sequence of divisions around the active face has a very regular structure, $iT - A - oT - A - iT \ldots$, with aligned and transversal alternating, and no two transversal of the same type in order (because by construction aligned divisions are inserted only between transversal of opposite type). *The algorithm maintains this alternating structure for the active face.*

We label the aligned divisions in a face as *interior* (iA) or *exterior* (eA), depending whether the field line containing the aligned face only intersects the face over the division, or it has other points contained in the face. The initial labeling is easily determined by checking whether the integral line touches the border of the triangle from inside or outside. We will see that all divisions we add by inserting edges are exterior to the active face.

Splitting off strips. The algorithm is based on the following proposition, proved in the supplement.

Proposition 2. *(a) The sequence of divisions of the triangle border defined as above, has exactly two more external aligned divisions than internal; (b) The sequence of division contains a subsequence $T - eA - T - eA - T - iA$, or the triangle divisions form a strip.*

The algorithm consists of a repetition of a splitting step depicted in Figure 9, left.

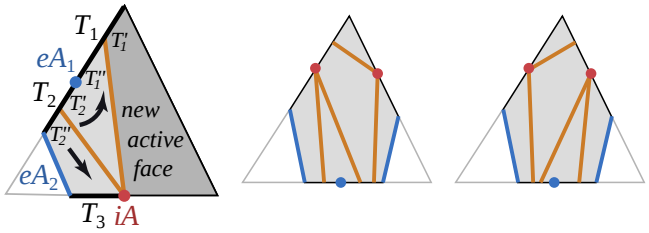

Figure 9: *Topology resolution, with the active region denoted in gray. Left: One step of the topology resolution algorithm, yielding a new active region highlighted in darker gray. Right: Ambiguous configurations.*

On each step, we find a subsequence $T_1 - eA_1 - T_2 - eA_2 - T_3 - iA$, in the sequence of active edges. It exists on initialization and we will show that at the end of the step, the same is true for the truncated face by construction.

Two new vertices A_1 and A_2 are inserted to split T_1 and T_2 into two subedges T_i' and T_i'', $i = 1, 2$, and two new divisions eA_1^n and eA_2^n are created, connecting endpoints (possibly coinciding) of division iA to T_1'' and T_2'. This splits of two strips. The new active face is obtained by replacing the whole sequence with two divisions T_1' and eA_2^n. As a result, the new active face has the number of divisions decreased by 4, with the number of each type of aligned divisions decreased by one, and the number of transversal divisions decreased by 2. This means that part (a), and as a consequence, part (b) of Proposition 2 holds for the active face. Note that the alternation of transversal and aligned divisions is preserved for the new active face.

By triangle convexity, we could never insert divisions intersecting the boundary not at endpoints transversally, or intersecting each other transversally. However, some strips generated in this way may be bounded by several divisions connecting points on the same edge resulting in a degenerate strip, which violates required field properties. The algorithm resolving such strips to nondegenerate is described below.

While the algorithm above produces a valid partition of triangle into strips, *a priori* it is unclear if it has the same topology as the original field. In the supplement, we demonstrate that the ambiguity is small, and reduces to just two cases (Figure 9, right). For our application, an arbitrary choice between the two does not have much effect, other than on field line quality. If desired in these cases, the triangle may be refined or a high-accuracy tracing may be performed to distinguish between these.

Degenerate strips. The algorithm above generates correct connectivity for strips, but in some cases these are degenerate, because *in*- and *out*-divisions are on the same edge of the triangle.

We note that at each step of the algorithm, two strips are generated (Figure 9). We observe that divisions T_2 and T_3 cannot be on the same triangle side, hence the first strip cannot be degenerate. However, T_1 and T_2 can be on the same edge. If the second strip is degenerate, we observe that eA_1 has to be a point tangent division; interior tangent divisions are always points, and have to be on different edges, so we can construct a segment connecting eA_1 and iA

(Figure 10a), separating the strip into two, with the newly inserted segment, not collinear with the T_1 and T_2, marked as *in*-transversal for one and *out*-transversal for the other strip. These strips are non-degenerate.

Once no topology resolution steps are possible, there is a single strip remaining, with two tangential divisions, and two transversal, in and out. If it is degenerate, then one of the tangential divisions is a point (it has to separate two transversal divisions on the same triangle side). Suppose at least one resolution step was done. Then the last strip has to have at least one internal segment tangential division, i.e it has to be of the form shown in Figure 10b, which also shows how it is separated into two nondegenerate strips.

Finally, if no topology resolution is needed (i.e. the triangle is a single strip from the beginning), the second tangent division can be either a vertex (on the same edge or not), or a point on an edge (same or not). These cases are shown in Figure 10c-f. After these additional splits, all strips are nondegenerate.

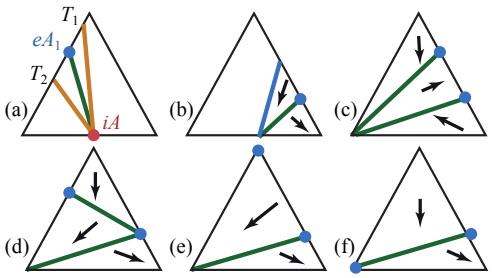

Figure 10: *Splitting strips with tangential divisions on the same triangle side. Split lines are shown in green, and arrows indicate the overall direction of field lines on all resulting strips.*

Refinement for bijectivity. If any integral lines intersect in a facet while not obeying the orientation requirements (i.e. corresponding to a non-bijective parametrization), then each such facet is added to a list of facets to refine. The maximal angle change is identified at each vertex and along each edge. If the maximal angle change is at a vertex, then the triangle is refined using bisector at the vertex. If maximal angle change is at an edge, then that edge is refined at the midpoint.

By field continuity, we eventually refine enough so that the topology of both orthogonal fields on the triangle is trivial (a single strip), in which case the bijectivity condition holds.

This happens very rarely and has so far been observed only on elephant and omotondo shapes.

Resolving geometry. It remains to explain how points are inserted to split T_1 and T_2 at the topology-resolution step. As our primary goal is robustness and efficiency rather than high accuracy, we use a simple algorithm that satisfies two constraints: (a) it is exact for a strip with a constant field; (b) the positions computed using this algorithm are continuous with respect to field changes. If these two conditions are satisfied, then we expect, for a sufficiently small deviation of a field from constant on a triangle, the error of the field approximation satisfying criteria to be of order $O(h^2)$, where h is the triangle size. (We do not include a detailed analysis of this, as the accuracy of tracing has little impact on our primary application.)

Our computation is based on minimizing the absolute total *flux* though the boundary of each strip. We observe that for a unit-length field, and linear angle interpolation from α to β relative to the direction of line segment, the flux on a line segment d of length L can be easily computed as

$$F(d) = L(\sin\beta - \sin\alpha)/(\beta - \alpha) \qquad (2)$$

reducing to $L\cos(\alpha)$ for constant field, similar to [Ray and Sokolov 2013].

For each strip, the transversal divisions may either have *known* endpoints (tangency points or triangle vertices), or *unknown* (inserted in the refinement process). By considering different cases of strips, any strip with unknown endpoints for a division, either has known endpoints for the other division, or can be virtually split into two strips (cf. Figure 10a) by a diagonal with known endpoints, resulting in two strips each with at least one division with known endpoints.

For a transversal division T, we denote the opposite division in a strip $opp(T)$.

Furthermore, every initial division on the boundary has known endpoints and a corresponding flux, and any division with unknown endpoints is a subdivision of a boundary division. If a transversal division T has subdivisions T_i, $i = 1 \dots n$, with unknown endpoints $\mathbf{p}_i$ in the interior of T, we want $\sum_i F(T_i)$ to be equal to $F(T)$, and, at the same time each $F(T)$ close to $-F(opp(T)) = G_i$, which can be approximated using Eq. 2. All fluxes of subdivisions of T have the same sign, which, we assume to be positive. Then we define each flux as w_i^2, to ensure it does not change sign, and minimize $\sum_i (w_i - G_i^{1/2})^2$, subject to $\sum_i w_i^2 = F(T)$, yielding a simple solution $w_i^2 = F(T)G_i / \sum_i G_i$. Once fluxes on an edge are known, the point positions are easily deduced from the flux formula.

Finally, once positions of all unknown points are determined, the map of the triangle boundary to itself is defined per strip, with one-to-one p.w. linear map of the in transversal division of the strip to the *out*-transversal division. If both transversal divisions are line segments then the map is linear. Otherwise one of the transversal divisions consists of two or three segments on different triangle edges, and we use fluxes through each segment to split the opposite transversal division.

For the proposed scheme, properties (a) and (b) above are satisfied: (a) can be checked directly; (b) requires a bit more work, and an outline of the proof is provided in the supplementary document.

Floating-point accuracy. For topological correctness, it is necessary to ensure all initial tangency points are distinct, and do not coincide with vertices and point positions $\mathbf{p}_i$ computed using formulas above are in the correct order along the triangle boundary. All geometric operations require floating point computations, and correctness conditions may be violated in extreme cases due to limited accuracy of calculations. The use of exact predicates would be the most reliable approach but new ones would have to be developed in addition to those already available. In our current implementation, we use scaled integer barycentric coordinates in the range $0 \dots 2^{32}$, and explicitly enforce all nondegeneracy conditions. In extreme cases, this may be impossible (e.g., in the case of a very short division of length 2^{-32}, with several subdivisions). Such cases are very rare and we have encountered them only on artificial examples.

Comparison to [Ray and Sokolov 2013]. Concurrent work [Ray and Sokolov 2013] uses an identical field representation. The topology resolution algorithm is somewhat different (a single strip is eliminated each time, inserting a transversal division, instead of inserting tangent divisions as we do; due to (near-) uniqueness of decomposition the resulting edge map unavoidably is similar. The main differences include: we analyze decomposition uniqueness, introduce additional refinement steps to avoid degenerate strips, and the algorithm for refinement to ensure bijectivity on a triangle. The algorithm for computing positions of inserted points is similar.

7 Consistency and simplification

The result of the motorcycle graph construction algorithm is a partition of the surface into quads, possibly with T-joints. In general, a seamless global parametrization mapping each quad of this mesh to a rectangle in the parametric domain does not exist. In this section, we describe an algorithm for modifying the quad partition to guarantee the existence of a global parametrization.

7.1 Partition consistency

If each quad Q_i of the partition is mapped to a rectangle, the basic condition required for existence of a global seamless parametrization is that the parametric edge lengths on two opposite T-edges of Q_i are equal. If the lengths of edges forming two opposite T-edges are b^0_j, $j = 1 \dots n_0$, and b^1_j, $j = 1 \dots n_1$, respectively, the *consistency conditions* simply say that the parametric lengths are equal on two sides:

$$\sum_{j=1}^{n_0} b^0_j = \sum_{j=1}^{n_1} b^1_j \tag{3}$$

subject to the constraints $b^k_j > 0$. We call a set of non-negative (but not necessarily positive) parametric lengths *consistent* if these satisfy 3 for any two opposite T-edges of rectangle.

To determine a unique parametric length assignment, we compute the length ℓ_k of each partition edge on the surface, and minimize the quadratic energy

$$E_{\text{len}} = \sum_{k=1}^{n} (b_k - s\ell_k)^2 \tag{4}$$

where an auxiliary variable s allows for edge scaling. As this system is homogeneous, it always has a solution for $b_k \geq 0$, but some edge length may be forced to be zero. This situation can often be viewed as a discrete version of a limit cycle (cf. Figure 11).

If a surface is partitioned into quad patches with no T-joints, the constraint system is trivial, and a solution with no zero edges always exists. For the general case, we aim to minimize the number of b_k forced to be zero. In principle, algorithms exist to find a maximal consistent subsystem, but have exponential complexity. Instead, we use a heuristic, which in practice yields sets of zero b_k variables of very small cardinality. The idea is to remove all positive lengths from the energy (4), and iteratively identify additional edges that can be nonzero. The supplemental document presents the details of the algorithm.

The consistency condition (3) ensures that if one of the opposite sides of a quad is zero, the other is also zero, i.e. the quad is completely degenerate (*zero quads*); otherwise, both sides retain non-zero parametric length.

7.2 Topological partition simplification

A remarkable property of a partition with *consistent* (i.e. satisfying Eq. 3) but possibly degenerate parametric length assignments is that if we construct a mesh by identifying the opposite sides of quads with a zero parametric dimension, the result is still a valid quad mesh, unless the process results in a handle on the surface collapsing to an edge or a point.

This suggests the idea of *partition simplification*: quads with zero parametric dimensions are merged with nearby quads to form a new partition for which non-zero edge assignments are possible (Figure 15).

Zero quad chains. We start with a purely topological description and explain how the topological operations are performed geometrically at the end.

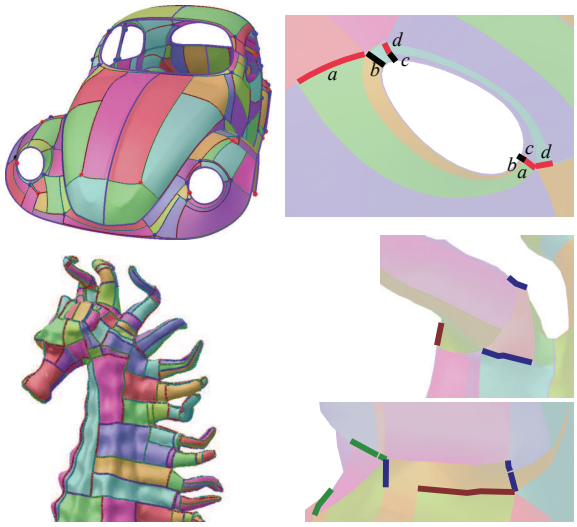

Figure 11: *Examples of zero edges highlighted on the right. Zero chains in the Beetle model are associated with a limit cycle at the border: zero chains are shown in red, and nonzero edges forcing zero edges in black. Consistency conditions require $c = b + a$, and $b = c + d = b + a + d$. From non-negativity, it follows that $a = d = 0$.*

To define simplification operations more precisely, we introduce several definitions.

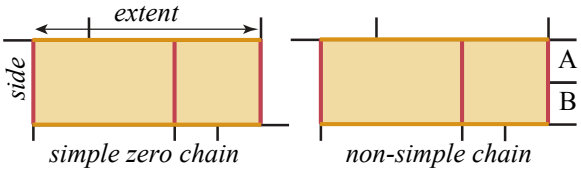

Figure 12: *Simple and non-simple quad chains and notation. Collapsing a non-simple quad chain cannot be done without collapsing quads A and B.*

We say that a sequence of quads, with each quad, except first and last, sharing opposite T-edges with the previous and next quads is *a quad chain*; a maximal chain is a chain that cannot be further extended (Figure 12). Note that we do not assume that quads in a chain do not repeat. In *a zero chain* the common edges all have zero parametric length. The *extent* sides of zero-chain quads are the sides which are *not* a part the zero-edge sequence (note these also can have zero lengths as a part of another chain). A chain is *simple*, if its side edges have at least one endpoint that is a T-joint. As described below, a simple chain can be removed by a collapse process while maintaining a quad partition. The *extended valence* of a vertex is its valence plus the number of quads in which it is a T-joint; that is, the valence of a vertex after all T-joints are extended.

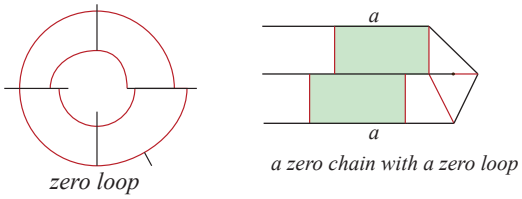

Figure 13: *An example of a zero loop (zero edges shown in red); a quad chain containing a zero loop, sides marked a are identified; collapsing this chain may result in a topology change.*

A zero loop is a closed sequence of zero elements, each element being either a zero edge or a zero quad; two sequential elements share a vertex; if one of the elements is a quad, the shared vertex is on an extent side, and vertices shared with the previous and next elements are on opposite extent sides. An *admissible zero chain* does not contain zero loops, and two sequential quads in a chain are always different.

Maximal simple zero chain collapse. The basic simplification operation that we use removes a single zero chain from the partition. It iterates over quads R_j, $j = 0, \dots n$, of the chain.

For each quad, two operations are performed (Figure 14).

(1) T-joints on the extent sides of R_j are extended to intersect with opposite sides, to get a sequence of new quads Q_i (to simplify notation, we use a single index for these, starting from the first subquad of Q_0). Quads Q_i do not have T-joints on extent sides.

(2) All resulting quads Q_i are sequentially collapsed. To define the operation more precisely, we denote the zero edge of Q_i common with Q_{i-1}, by $e_i = (v_i, w_i)$; for each zero edge, we define a collapse direction by choosing which of the two vertices is removed. We label vertices so that w_i is always removed. We explain the choice of w_i below (for meshes satisfying Proposition 4, the choice is not important). Q'_i and Q''_i are adjacent to Q_i on two sides, across its extent edges f_i and g_i, respectively.

A single intermediate step of the collapse operation works as follows: **Initial state:** Q_{i-1} is a triangle, with the edges e_i, f_{i-1} connecting v_{i-1} and w_i, and g_{i-1} connecting v_{i-1} and v_i. All edges e_j for $j < i$ removed, and opposite sides identified. **Collapse:** Remove edge f_{i-1} merging Q_{i-1} with Q'_{i-1}. Collapse edge e_i, turning Q_i into a triangle. Label v_i as a T-joint. **Final state:** Configuration similar to the initial one, with $i-1$ replaced with i.

At the first step, there is no Q_{i-1} to remove, and at the last step, no Q_i becomes a triangle.

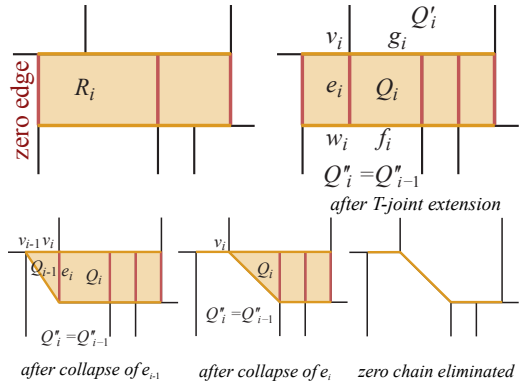

Figure 14: *Notation and steps of collapsing a zero-chain, with extent edges in orange and zero edges in red.*

A more trivial operation is a collapse of a single zero edge not belonging to any chain (an *isolated zero edge*). Such an edge always has an end point which is a T-joint v; otherwise, it would be a T-edge, and the opposite T-edge in quad would have to be zero. It can removed on its own, simply merging it with the edge with which it shares v.

Partition simplification algorithm. The overall algorithm is just:

While there are zero edges find an isolated zero edge or a simple zero chain, and apply the collapse operation to it.

Its behavior is described by two propositions. The first guarantees that the algorithm can always proceed.

Proposition 3. *If a mesh has zero edges, there is either an isolated zero edge or a simple zero T-chain.*

The second proposition describes the set of partitions that can be made consistent by the algorithm:

Proposition 4. *Suppose a partition satisfies the following conditions: (1) all vertices in the partition have extended valence 3 or greater; (2) there are no valence 3 cones connected by a zero path; (3) zero quad chains contain no zero loops. Then any simple zero chain can be collapsed, and the resulting mesh has the same properties.*

Requirement (1) and (2) are needed because collapsing a vertex of valence less than 3 with a vertex of valence less than 4 may result in vertices of valence 1, i.e. quads with two edges glued to each other, which cannot be collapsed. These conditions ensure that no vertices of valence below 3 appear as a result of collapses. Requirement (3) is needed to avoid topological degeneracies; it implies that if one follows a parametric line along a zero edge of a zero chain, eventually a non-zero quad is reached. The conditions of the proposition are sufficient but not necessary for removing all zero chains. The proofs are in the supplemental document.

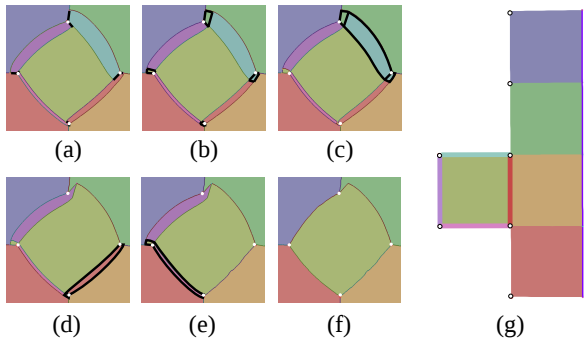

Figure 15: *Collapsing several zero chains. (a) Partition with zero edges highlighted in black; (b) simultaneous T-joint extensions; (c)-(e) sequential zero chain collapses; (f) result with no zero chains. (g) The image of the surface in the parametric domain conceptually remains the same throughout. The four zero-chains map to the four line segments highlighted around the light green quad.*

The constraints of the proposition are relatively weak (there were 4 cases out of 114 for which these were not satisfied, as discussed in Section 8); we discuss below the modifications to the algorithm that need to be made to handle the general case.

Algorithm for general meshes. The partition simplification algorithm can be applied to any mesh, not just those satisfying the conditions of Proposition 4, but with no guarantee that a zero quad chain can be collapsed. In this case, we try to maximize the number of collapsed chains, and resolve the rest by adding cones.

Algorithm modifications. For meshes not satisfying the conditions of Proposition 4, a specific heuristic for choosing vertices w_i affects the outcome. Additionally, the stopping criterion for the algorithm is not the elimination of all zero chains – rather, reaching a state when no remaining zero chain can be collapsed.

If sharp features are present, the endpoints of the path cannot be relocated, and the feature valence (number of feature edges at a vertex) should not change. This requires selecting a vertex which is not a feature endpoint as w_i, if one is available. If one vertex is on a border, the non-border vertex has to be used as w_i.

A zero path is not collapsed if (1) there is a zero edge e_i whose collapse results in an extended valence 2 vertex; (2) a zero loop is present; or (3) the endpoints of an edge are feature vertices, except in the case when the edge is a feature edge, and one endpoint is not a feature endpoint.

Cone insertion. In the final state, there may be a number of uncollapsible quad chains remaining. To fully eliminate zero edges

and obtain a valid parametrization, we observe that for each chain, removing the constraints corresponding to the side quads makes it possible to choose nonzero edge lengths for the chain. We remove constraints one by one, and re-solve for the parametric length each time, until no zero parametric lengths are left. This state is always achieved because additional constraints are removed at every step. The detailed algorithm is included in the supplemental document.

We say that a quad is non-compatible if one of the constraints was removed for it. If two constraints are removed for a quad, we call it doubly non-compatible. We replace each non-compatible quad with 4 quads, inserting a 2-6 pair, and a doubly non-compatible quad with 7 quads, inserting 2 pairs (Figure 16). This is the simplest way to resolve edge incompatibility for a quad. The quality of the resulting parametrization depends on the placement of singularities.

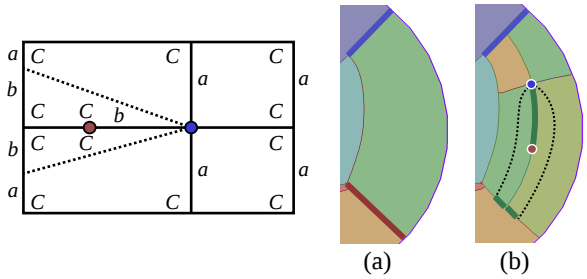

Figure 16: *Cone insertion: a single 2-6 pair makes it possible to have different parametric lengths a and $a + b$ on two sides of a quad. Only split into 4 quads marked with solid lines is needed, if the (extended) valence 6 cone is labeled as a non-corner vertex in 2 faces. In two resulting faces corners are marked C.*

7.3 Updating partition geometry

The simplification algorithm presented so far describes how partition connectivity can be changed to make it consistent. New edges appear at two points of the algorithm: when T-joints are extended, and when diagonal edges are introduced when a zero quad Q_i is split between two adjacent quads. We describe how the curves corresponding to these edges are introduced on the surface, and how corresponding parametric lengths are assigned.

Tracing new edges. Any new edge connects points (corners, T-vertices or vertices introduced by T-joint extension) of a quad of the current T-mesh.

New vertices and edges created by extension also have to be placed on the original geometry. If v_2 is a new vertex on a T-edge e, added on a non-zero T-edge by extending a T-joint v, it splits e on the surface in the same ratio as v splits the opposite edge of the quad. If the vertex is added on a zero T-edge, it splits the T-edge in half. This ensures that all newly added vertices have distinct positions on the mesh.

To trace newly added edges on the surface, we parametrize bijectively each quad on a rectangle, using a positive-weights discrete harmonic map, and trace a straight line between the end points to be connected in the parametric domain, which we then remap to the mesh, cutting it along the traced lines. This procedure guarantees correctness of the T-mesh topology embedded in the surface (i.e. edges intersect only at endpoints and the part of the surface bounded by face edges is simply-connected). It is applied both for T-joint extension and computing new edges when a zero quad is collapsed.

Updating parametric lengths. For quad collapse, the assignment of parametric lengths is simple: existing edges retain their parametric lengths, and if a diagonal new edge is introduced, it is assigned the parametric length of the pair of opposite edges of Q_i collapsed to it. However, T-joint extensions require splitting edges and assigning parametric lengths to different parts. The process is straightforward as long as T-joint positions are distinct and the extent edges do not have zero parametric lengths themselves. A specific method for resolving ambiguity extension is not that important for the algorithm as long as T-mesh consistency is preserved, and no T-joints are left on the sides of zero chain quads. We use the following assignment. For each T-joint u_1 on side f we extend it across the quad to side g. The new vertex u_2 has the same parametric distance from an arbitrarily chosen side of the quad as u_1. Suppose there are m vertices at the same parametric location as u_1 and n vertices at the same parametric location as u_2; and u_1 is preceded by k T-joints along f. The connectivity is resolved by connecting u_1 to vertex number k at location u_2 on g; if $k > n$, a new vertex is inserted to the right and is assigned the same parametric location.

8 Evaluation and Discussion

Test data set. As the focus of our work is on robustness, we have tested our algorithm on a comparatively large data set. We have started with *all* manifold mesh models from the AIM@Shape database, and added models from the Stanford shape repository and several commonly used ones in related works. We have excluded two categories: height fields representing terrains, as all of these have similar behavior and can be trivially parametrized by projection, and spline surface control meshes.

While the performance of our main set of algorithms (field tracing, partition simplification and cone insertion) is fast, the final-stage constrained cone and quadratic optimization is slow, so all meshes with face count above 100k were downsampled to 100k faces using the AIM@Shape online decimation tool. We obtained a dataset of 114 meshes.

The cross-fields on the meshes were generated using [Bommes et al. 2009], with its default parameter choices for feature detection. Additionally, cones separated by one edge were collapsed to reduce the number of cones on noisy meshes. While the resulting field quality was not suitable for high-quality parametrization, no further field modifications were done since our goal is to test the robustness of the method to field behavior, *Detailed information on the data set and test results is included in the electronic supplement.*

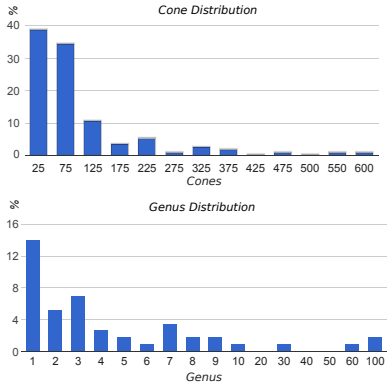

Figure 17: *Histograms of cone numbers (top) and genus (bottom) in our data set.*

Basic statistics on the data set are shown in Figure 17. As we observe, both mesh genus and cone counts vary in a broad range. In addition, the triangle aspect ratio was as high as 10^6.

Comparisons. We compare to two state-of-the art methods, both based on minimizing a quadratic energy with constraints: the trisector constraints of [Bommes et al. 2013] (IGM) and the quasiconformal cone constraints of [Lipman 2012] (BD) using the cross-field

as the initial frame field. In all figures, our method is referred to as RGP. We also compare to the stiffening technique of [Bommes et al. 2009] (MIQ). Although less reliable and predictable than constrained-based methods, it has the advantage that it does not fail completely, as convex solvers often do, if no bijective solution is found.

As discussed in Section 7, the final parametrization is obtained in the same way, but the frame is extracted from the initial parametrization. In both cases, the quasiconformal constraint bound was set to 0.99 to minimize failures of the method of [Lipman 2012] on the original mesh.

Bijectivity. Our key result is that, for *all* 114 models our method has produced locally bijective initial parametrizations. In comparison, IGM fails for 29 models, and BD fails for 20 models. Figure 25 shows some examples of problems.

Furthermore, using our bijective parametrization as a starting point, the final BD-constrained parametrization was successfully computed for all but one model, for which there was a single non-bijective triangle (vhskin). This is apparently due to the poor conditioning of the resulting system, preventing the cone solver (MOSEK) from solving the problem with sufficient accuracy.

Examples comparing initial and final parametrizations are shown in Figure 18.

We were unable to identify precisely the reasons why BD and IGM do not find a feasible solution for certain meshes. There appears to be no particular correlation with mesh quality (either nearly-degenerate triangles, or very high Gaussian curvature values), but higher correlation with field smoothness and complexity. Some examples are shown in Figure 19, and the supplementary material includes detailed statistics of mesh edge lengths and vertex total angle distributions, as well as additional images of meshes with problematic parametrizations.

Distortion comparison. We compare the distortion of the parametrizations produced by the three methods only on *meshes for which IGM and BD do not fail*.

Overall, we observe that the parametrizations we compute have as good or lower distortion than previous work. The small improvement in distortion in many cases can be explained by the added degrees of freedom, especially near cones where the distortion is typically highest.

Effects of collapsing zero chains. As discussed in Section 7, zero chain collapse can be thought of as a local topology-preserving field modification. While this collapse affects field quality (the operation is purely topological with no attempt to preserve field directions), the last stage uses the original field for the BD-constrained parametrization fit, and the field quality is recovered. This can be seen in Figure 21.

Cone insertion. While T-collapse is needed relatively commonly (39 meshes), only four meshes required cone insertion (raptor50k, dancer2, twirl, brain).

As an additional demonstration of robustness, we include an artificial example: the random mesh example from [Bommes et al. 2013] (Figure 23). Interestingly, compared to some of the "real" meshes from our dataset, the random mesh example proved to be easy: on such random meshes, collapses are rarely required, and no cone insertion was needed. We observe that random triangles sampled from a sphere generally have a good aspect ratio, and cones do not have very high valence. Another extreme example is the parametrization of a 2-torus with a single singular point.

Dependence on field quality. All tests were done on fields generated automatically, without taking isometry into account and with no manual improvement. Our observation is that for fields with penalty for high curl (we used the method of [Myles and Zorin 2012]), the need for zero-chain collapses drastically decreases. For example, four cases that originally required cone insertion, no longer require it after penalizing high curl (see Figure 22). Furthermore, only 6 out of 39 models originally requiring zero-chain collapse still needed it. In many cases, this comes at the expense of feature alignment.

Figure 18: *Comparison of the initial parametrizations computed from the consistent partitions (left) and the final parametrizations (right).*

Timings. Most of the algorithm is sublinear, as it does not need to visit every triangle. As each quad patch needs to be parametrized, the performance is mostly determined by the speed of the linear

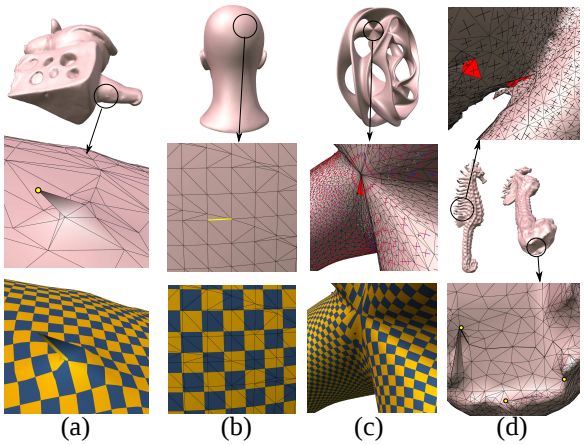

Figure 19: *(a)–(b) A mesh with a "spike" and a mesh with nearly-degenerate triangles for which both BD and IGM produce good results. (c) A mesh with neither bad triangles or spikes for which BD and IGM fail. (d) On this mesh, the "spikes" are located in places far from inverted triangles.*

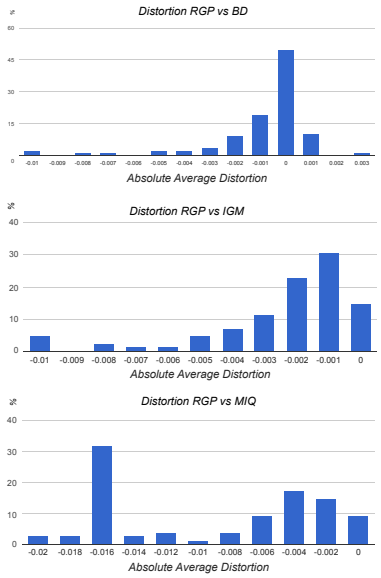

Figure 20: *Distortion comparison: the histograms show the distribution of mean-square average distortion difference, RGP minus BD, RGP minus IGM, and RGP minus MIQ with stiffening.*

solver. Figure 24 shows how total time is distributed. Overall, the cost is negligible compared to the cost of the constrained solve. The timing outliers are relatively noisy meshes with a large number of cones.

Limitations. In one case (twirl), our method does have to eliminate one zero chain by cone insertion, necessitated by a zero loop. Using [Lipman 2012] constraints, on the other hand, produces a locally bijective parametrization with no foldovers. This confirms that the class of inconsistent T-meshes that can be made consistent without adding cones is broader than the one given by Proposition 4.

9 Conclusion

The presented set of algorithms provides a way to compute feature-aligned parametrizations robustly as demonstrated on a set of shapes an order of magnitude larger than most previous work. The

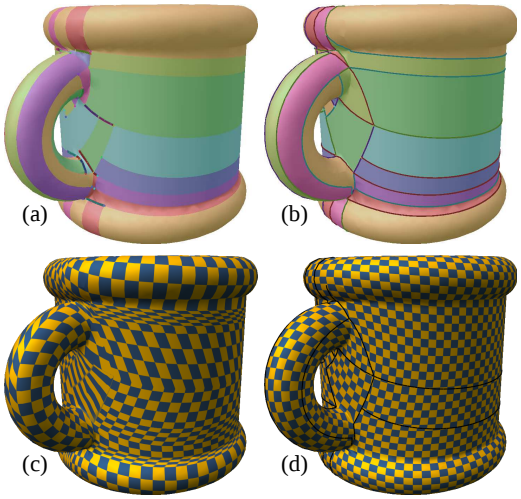

Figure 21: *Effects of zero-chain collapse on the field. Zero edges are highlighted in (a). (b) shows the result of collapse (note the pentagonal-shaped face, which is actually a logical quad). In (c) observe distortion of the initial parametrization in this area which disappears after optimization in (d).*

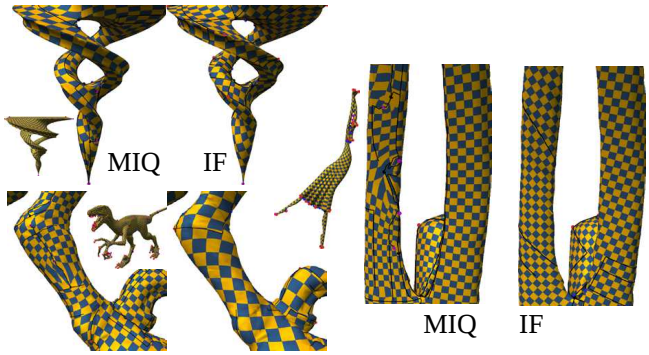

Figure 22: *Three out of four models requiring cone insertion. The original result is on the left of each. On the right, the results for more isometric (but less aligned) fields generated using [Myles and Zorin 2012], for which no cone insertion was needed.*

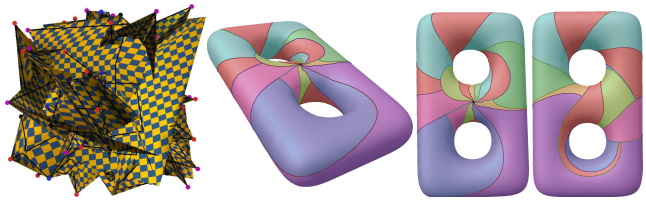

Figure 23: *Left: A random mesh sampled from a sphere. Right: Partition for a torus with a single singularity.*

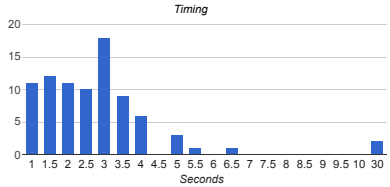

Figure 24: *Total time of tracing, collapse and cone insertion.*

algorithm for computing the initial parametrization is efficient, and does not have a significant impact on overall running time.

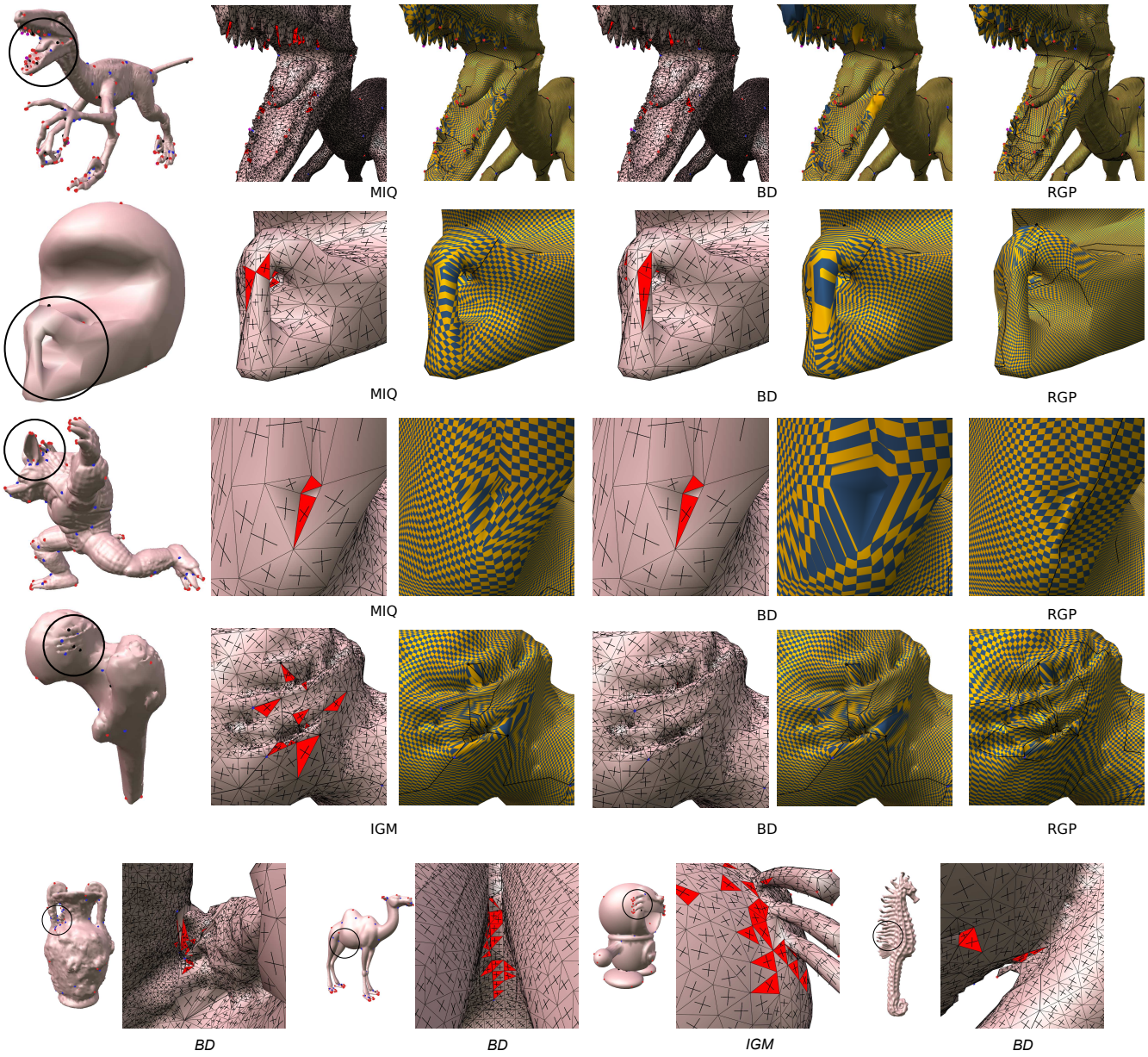

Figure 25: *Examples of non-bijective parametrizations (inverted triangles shown in red) produced by MIQ with stiffening, IGM and BD methods, compared to the same areas parametrized using RGP. For the femur model, the folded triangles for the BD method are inside a tunnel and not visible. The lower row shows additional examples of inverted triangles for different methods.*

There are many directions for improvement that are easy to explore: first, the motorcycle graph algorithm is the simplest but not the best way to construct a quad partition; methods similar to [Myles et al. 2010] can be used for further improvement. Second, while the partition simplification by quad chain collapse handles most cases requiring field modification, this is not the only possible operation for eliminating zero edges. We conjecture that as long as the holonomy type of the field is compatible with a quadrangulation, one can construct a nondegenerate quad partition. Last, but not least, combining our method with a rounding technique like [Bommes et al. 2013] is essential for quadrangulation.

Acknowledgements

This work was partially supported by NSF awards ACI-1047932 and IIS-0905502.

References

ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LÉVY, B., AND DESBRUN, M. 2003. Anisotropic polygonal remeshing. In *ACM Transactions on Graphics (TOG)*, vol. 22, ACM, 485–493.

ANDRONOV, A. A., LEONTOVICH, E., GORDON, I., AND MAIER, A. 1973. *Qualitative theory of second-order dynamic systems*. Israel Program for Scientific Translations Jerusalem.

BEN-CHEN, M., GOTSMAN, C., AND BUNIN, G. 2008. Conformal flattening by curvature prescription and metric scaling. *Computer Graphics Forum 27*, 2, 449–458.

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

BOMMES, D., LVY, B., PIETRONI, N., PUPPO, E., SILVA, C., TARINI, M., AND ZORIN, D. 2012. Quad Meshing. Eurographics Association, Cagliari, Sardinia, Italy, M.-P. Cani and F. Ganovelli, Eds., 159–182.

BOMMES, D., CAMPEN, M., EBKE, H.-C., ALLIEZ, P., KOBBELT, L., ET AL. 2013. Integer-grid maps for reliable quad meshing. *ACM Trans. Graph. 32*, 4.

DONG, S., BREMER, P., GARLAND, M., PASCUCCI, V., AND HART, J. 2006. Spectral surface quadrangulation. *ACM Trans. Graph. 25*, 3, 1057–1066.

EPPSTEIN, D., AND ERICKSON, J. 1999. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete and Computational Geometry 22*, 4, 569–592.

EPPSTEIN, D., GOODRICH, M. T., KIM, E., AND TAMSTORF, R. 2008. Motorcycle graphs: canonical quad mesh partitioning. In *Computer Graphics Forum*, vol. 27, Wiley Online Library, 1477–1486.

GUNPINAR, E., MORIGUCHI, M., SUZUKI, H., AND OHTAKE, Y. 2014. Feature-aware partitions from the motorcycle graph. *Computer-Aided Design 47*, 85–95.

JADHAV, S., BHATIA, H., BREMER, P.-T., LEVINE, J. A., NONATO, L. G., AND PASCUCCI, V. 2012. Consistent approximation of local flow behavior for 2d vector fields using edge maps. In *Topological Methods in Data Analysis and Visualization II*. Springer, 141–159.

KHAREVYCH, L., SPRINGBORN, B., AND SCHRÖDER, P. 2006. Discrete conformal mappings via circle patterns. *ACM Trans. Graph. 25* (April), 412–438.

KNÖPPEL, F., CRANE, K., PINKALL, U., AND SCHRÖDER, P. 2013. Globally optimal direction fields. *ACM Transactions on Graphics (TOG) 32*, 4, 59.

LEE, A., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. MAPS: multiresolution adaptive parameterization of surfaces. In *SIGGRAPH 1998*, 95–104.

LI, W.-C., VALLET, B., RAY, N., AND LÉVY, B. 2006. Representing higher-order singularities in vector fields on piecewise linear surfaces. *Visualization and Computer Graphics, IEEE Transactions on 12*, 5, 1315–1322.

LI, W., RAY, N., AND LÉVY, B. 2006. Automatic and interactive mesh to T-spline conversion. In *Symposium on Geometry Processing*, Eurographics Association, 200.

LIPMAN, Y. 2012. Bounded distortion mapping spaces for triangular meshes. *ACM Transactions on Graphics (TOG) 31*, 4, 108.

MYLES, A., AND ZORIN, D. 2012. Global parametrization by incremental flattening. *ACM Transactions on Graphics (TOG) 31*, 4, 109.

MYLES, A., AND ZORIN, D. 2013. Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics (TOG) 32*, 4, 105.

MYLES, A., PIETRONI, N., KOVACS, D., AND ZORIN, D. 2010. Feature-aligned T-meshes. *ACM Trans. Graph. 29*, 4, 1–11.

RAY, N., AND SOKOLOV, D. 2013. Tracing cross-free polylines oriented by a n-symmetry direction field on triangulated surfaces. *arXiv preprint arXiv:1306.0706*.

RAY, N., LI, W., LÉVY, B., SHEFFER, A., AND ALLIEZ, P. 2006. Periodic global parameterization. *ACM Trans. Graph. 25*, 4, 1460–1485.

RAY, N., VALLET, B., LI, W., AND LÉVY, B. 2008. N-Symmetry direction field design. *ACM Trans. Graph. 27*, 2.

SHEFFER, A., AND DE STURLER, E. 2001. Parameterization of Faceted Surfaces for Meshing using Angle-Based Flattening. *Engineering with Computers 17*, 3, 326–337.

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

SZYMCZAK, A., AND ZHANG, E. 2012. Robust morse decompositions of piecewise constant vector fields. *Visualization and Computer Graphics, IEEE Transactions on 18*, 6, 938–951.

TONG, Y., ALLIEZ, P., COHEN-STEINER, D., AND DESBRUN, M. 2006. Designing quadrangulations with discrete harmonic forms. *Symposium on Geometry Processing*, 201–210.

TRICOCHE, X., SCHEUERMANN, G., AND HAGEN, H. 2000. Higher order singularities in piecewise linear vector fields. In *The Mathematics of Surfaces IX*. Springer, 99–113.

TRICOCHE, X. 2002. Vector and tensor field topology simplification, tracking, and visualization. In *PhD. thesis, Universität Kaiserslautern*, Citeseer.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2006. Vector field design on surfaces. *ACM Transactions on Graphics (TOG) 25*, 4, 1294–1326.