SAND2019-5668C

COARSE QUAD LAYOUTS THROUGH ROBUST SIMPLIFICATION OF CROSS FIELD SEPARATRIX PARTITIONS

Ryan Viertel[1] Braxton Osting[2] Matthew Staten[1]

[1]*Sandia National Laboratories, Albuquerque, NM, U.S.A. rvierte@sandia.gov, mlstate@sandia.gov*
[2]*Univeristy of Utah, Salt Lake City, UT, U.S.A. osting@math.utah.edu*

ABSTRACT

Streamline-based quad meshing algorithms use smooth cross fields to partition surfaces into quadrilateral regions by tracing cross field separatrices. In practice, reentrant corners and misalignment of singularities lead to small regions and limit cycles, negating some of the benefits a quad layout can provide in quad meshing. We introduce three novel methods to improve on a pipeline for coarse quad partitioning. First, we formulate an efficient method to compute high-quality cross fields on curved surfaces by extending the diffusion generated method from Viertel and Osting (SISC, 2019). Next, we introduce a method for accurately computing the trajectory of streamlines through singular triangles that prevents tangential crossings. Finally, we introduce a robust method to produce coarse quad layouts by simplifying the partitions obtained via naive separatrix tracing. Our methods are tested on a database of 100 objects and the results are analyzed. The algorithm performs well both in terms of efficiency and visual results on the database when compared to state-of-the-art methods.

Keywords: cross fields, quad partitioning, quad meshing, surface decomposition

1. INTRODUCTION

Block structured quad meshes are often desirable because of their numerical efficiency [1], low memory requirements [2], and high mesh quality [3]. In the past, such meshes have been designed by hand or in an interactive environment [4]. Some researchers have attempted to automate this process using medial axis subdivision [5, 6]. More recent approaches to this problem use cross fields to guide construction of the decomposition [3, 7, 8, 9, 10, 11, 12]. Campen [13] provides a literature review on quad patching algorithms.

In this paper, we further develop methods for the cross field based pipeline for coarse quad layout generation summarized in algorithm 1.

Algorithm 1 Partitioning a surface M into a coarse quad layout with T-junctions.

```
Input: A triangle mesh $T$ representing a surface $M$ in $\mathbb{R}^3$.
Output: A quad layout with T-junctions $Q$ partitioning $M$ into four-sided regions.
1. Compute a smooth cross field on T.
2. Construct a quad layout with T-junctions by tracing out separatrices of the cross field.
3. Simplify the quad layout with T-junctions.
```

Other researchers have taken an approach similar to this pipeline. Kowalski et al. [3] design a cross field by solving a PDE with a constraint applied via Lagrange multipliers. They trace out separatrices from each interior singularity and boundary corner to obtain a quad layout and then mesh each region via a bilinear transfinite interpolation. Fogg et al. [11] take a similar approach, but initialize the cross field by an

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

advancing front method and then smooth it with the energy functional introduced by Hertzmann and Zorin [14]. Ray and Sokolov [15] and Myles et al. [9] both implement robust streamline tracing algorithms, taking as input a smooth cross field on a triangle mesh. They design partitions by tracing out separatrices in parallel until their first crossing with another separatrix to form a motorcycle graph [16]. In addition, Myles et al. [9] assign a parametric length to each edge using a heuristic method to achieve consistent parametric lengths. They then parameterize each region rather than building a quad mesh, and demonstrate the robustness of their algorithm on a dataset of 114 objects. Campen et al. [10] follow a similar approach to Myles et al. [9], but introduce a novel way to guarantee a consistent assignment of quantized parametric lengths on closed, orientable 2-manifolds. Viertel and Osting [17] introduce an efficient cross field design algorithm and use the Ginzburg-Landau theory to prove that separatrices of the resulting cross fields decompose a surface into quad regions, where T-junctions are only required when a limit cycle occurs.

A common theme with separatrix tracing approaches such as these is that on a discrete geometry, singularities are never perfectly aligned. In practice, this frequently causes limit cycles and very thin regions to occur within the quad layout. This is problematic for meshing because very small mesh elements are required. Further, the *base complex* of the mesh is often far more complicated than necessary, mitigating the benefits of a multi-block decomposition. Previous research addresses this problem in different ways. Tarini et al. [18] and Bommes et al. [19] both address the problem after a quad mesh is generated. They introduce grid preserving operators that simplify the base complex of a quad mesh at each step. Kowalski et al. [3] attempt to generate a coarse partition by snapping streamlines to singularities when they pass within a certain tolerance of the singularity. Myles et al. [9] identify sequences of edges in the motorcycle graph obtained via separatrix tracing that are assigned a zero parametric length and remove them via a collapse operation. Razafindrazaka et al. [12] formulate the problem of connecting singularities together as a minimum weight perfect matching problem. In a second paper, they extend this method to work on quad meshes [20].

1.1 Contributions and Outline

We follow the pipeline described in algorithm 1, taking as input a triangle mesh of a surface M. We use a per-node representation of the cross field, thus the triangle mesh is required to have a sufficient number of nodes on a given curve to accurately capture the underlying geometry (see section 3.3). In practice, an adaptive triangle mesh that has finer elements near areas of high curvature and corners and larger elements in areas that are relatively flat provides a good balance between computational efficiency and geometric fidelity. In our implementation, we use triangle meshes generated in CUBIT [21] using the TriMesh scheme with geometry adaptive interval assignment.

In section 2, we extend the diffusion generated method for cross field design of Viertel and Osting [17] to curved surfaces. This method is ideal for designing cross fields on the types of surfaces commonly found in CAD models because it is fast, robust, and has good singularity placement, especially near boundaries. In section 2.1, we review concepts from differential geometry and the literature on cross fields to formulate the problem of cross field design on a curved surface. In section 2.2, we describe a discrete formulation of the diffusion generated method for cross field design on surfaces.

In section 3 we describe the procedure we use to generate an initial quad layout with T-junctions. The initial steps closely follow the approaches used in previous works such as Kowalski et al. [3] and Fogg et al. [11]; in section 3.1 we detail our procedure for identifying singular triangles and for calculating the *ports*, i.e. directions where separatrices exit each singularity, and in section 3.2.1 we use a variation on Heun's method to trace separatrices through regular triangles. We improve upon the process in section 3.2.2, where we prove that the trajectory of a streamline in the neighborhood of a singularity is hyperbolic under a conformal map. This leads to a simple method for accurately calculating the trajectory of streamlines through singular triangles in a node-based representation of a cross field. This new method avoids tangential crossing of streamlines in the neighborhood of a singularity as is common with other methods. Finally, in section 3.3, we describe stopping conditions for separatrix tracing as well as how to deal with tangential separatrix crossings.

In section 4, we introduce a simple and robust method to simplify the partitions obtained from the methods in section 3. The algorithm iteratively employs a simple "chord collapse" operation to remove thin sections of the partition and limit cycles that occur due to singularity misalignment. Each chord collapse operation produces another valid partition, monotonically reduces the number of T-junctions, and strictly decreases the number of partition components. This method results in a coarse partition of M into quad regions that can be used as a starting point for quad meshing or parameterization methods.

In section 5, we test our method on several example surfaces and analyze the results. We note that our algorithm performs well in terms of efficiency and in visual quality as compared to other state-of-the-art methods. In section 6, we further discuss the improvements made in our pipeline as well as its limitations, and describe directions for future work.

2. CROSS FIELD DESIGN

Various methods exist to design a smooth cross field on a curved surface [14, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]. The method introduced in Viertel and Osting [17] is appealing because it is comparable in speed to the fastest methods such as those described in Knöppel et al. [28] and Jakob et al. [29], but also has distinct advantages on surfaces with boundary, which are common in CAD and meshing for FEM, because we can easily enforce Dirichlet boundary conditions unlike Knöppel et al. [28], and the singularity placement near the boundary is better than Jakob et al. [29]. In this section, we extend the diffusion generated method for cross field design to curved surfaces. We use this method of cross field design as the first step in our pipeline (algorithm 1). In section 2.1, we first review some tools from the literature that have been used previously to develop cross field design methods on surfaces. In section 2.2, we describe the implementation details of our method.

2.1 Cross Fields on Surfaces

The main difficulty in extending flat 2D methods to curved surfaces is the lack of a global coordinate system. In this section, we use concepts from differential geometry to formulate the cross field design problem on 2-manifolds. We consider a smooth, orientable 2-manifold M embedded in $\mathbb{R}^3$ and endowed with the Riemannian metric induced by the the Euclidean metric on $\mathbb{R}^3$. Let $\mathbb{T} = \{z \in \mathbb{C}\colon |z| = 1\}$ be the circle group with group operation given by complex multiplication and let $\rho(4)$ be the set of the 4th roots of unity. A *cross* is an element of $C = \mathbb{T}/\rho(4)$. There is a canonical group isomorphism $R\colon C \to \mathbb{T}$ called the *representation map* given by $R([c]) = c^4$, where c is any representative member of the equivalence class $[c] \in C$. We will refer to $u = c^4$ as the *representation vector* for $[c]$. The *inverse representation map* $R^{-1}\colon \mathbb{T} \to C$ assigns $u \in \mathbb{T}$ to $R^{-1}(u) = [\sqrt[4]{u}]$, the equivalence class of the 4th roots of u.

Let T_p be the tangent space of M at a point p. The disjoint union of all tangent spaces on M is called the *tangent bundle* and is denoted TM. For each tangent space T_p we select a coordinate basis, $\{\frac{\partial}{\partial x^1}|_p, \frac{\partial}{\partial x^2}|_p\}$. We also associate with each point p of M a space homeomorphic to C, which we call the *cross space* at p. We denote this space by C_p. The disjoint union of all cross spaces of M defines a fiber bundle that we refer to as the *cross bundle*. A section of the cross bundle is called a *cross field* on M. We make the natural identification between T_p, and the complex plane by the map $(a, b) \mapsto a+ib$. In this way, we can identify a cross, $[c_p]$, in C_p as an unordered set of four orthogonal unit vectors in T_p, which we call the cross *component vectors.* This also allows us to define a representation map R_p at each point with respect to the local coordinate basis, and a representation vector $u_p = R_p([c_p])$, which we identify with the corresponding unit tangent vector in T_p. For simplicity, we will use complex notation for equations throughout the paper.

In 2D cross field design, the goal of designing a smooth cross field is often formulated as designing a harmonic representation vector field u [30, 17]. That is, to minimize the Dirichlet energy

$$E[u] \coloneqq \frac{1}{2}\int_M |\nabla u|^2 dA \tag{1}$$

with the constraint that $|u| = 1$ at each point of the domain. We note that in general, this problem is ill-posed; however, generalized solutions exist if a finite number of singular points are removed from M [33, 17].

This strategy can be extended to surfaces by replacing the gradient operator ∇ with the appropriate *connection* on the tangent bundle. The Levi-Civita connection provides a way to compare vectors on the tangent bundle that preserves the notion of inner product between tangent spaces. That is, if P_{pq} is the parallel transport function for the Levi-Civita connection between T_p and T_q, and if $v_1, v_2 \in T_p$, then

$$\langle v_1, v_2\rangle_{T_P} = \langle P_{pq}(v_1), P_{pq}(v_2)\rangle_{T_q}.$$

Visually, the effect of using this connection in equation 1 is that a minimizing vector field appears smooth (see Figure 1).

In order to extend the strategy from 2D, using equation 1 to design a smooth cross field, we seek a connection, ∇^Q, on the tangent bundle with corresponding parallel transport function, Q, appropriate for transporting *representation vectors.* We choose Q in such a way that the component vectors of the corresponding crosses are transported by P, the parallel transport function corresponding to the Levi-Civita connection. Let $\gamma_{pq}(t)\colon [0, 1] \to M$ be a Levi-Civita geodesic connecting points p and q such that $\gamma_{pq}(t_p) = p$, $\gamma_{pq}(t_q) = q$, and $t_p, t_q \in (0, 1)$. Let ϕ_p be the signed angle from the velocity vector $\gamma'_{pq}(t_p)$ to $\frac{\partial}{\partial x^1}|_p$ and let ϕ_q be the signed angle from $\gamma'_{pq}(t_q)$ to $\frac{\partial}{\partial x^1}|_q$. Then if $\phi_{pq} = \phi_p - \phi_q$, $P_{pq}(v) = e^{i\phi_{pq}}v$ gives the Levi-Civita parallel transport function, P_{pq}, in coordinates with respect to the basis $\{\frac{\partial}{\partial x^1}|_p, \frac{\partial}{\partial x^2}|_p\}$.

Then, Q_{pq}, the parallel transport function for *representation vectors* between p and q, must satisfy

$$Q_{pq}(R_p([c_p])) = R_q([P_{pq}(c_p)]). \tag{2}$$

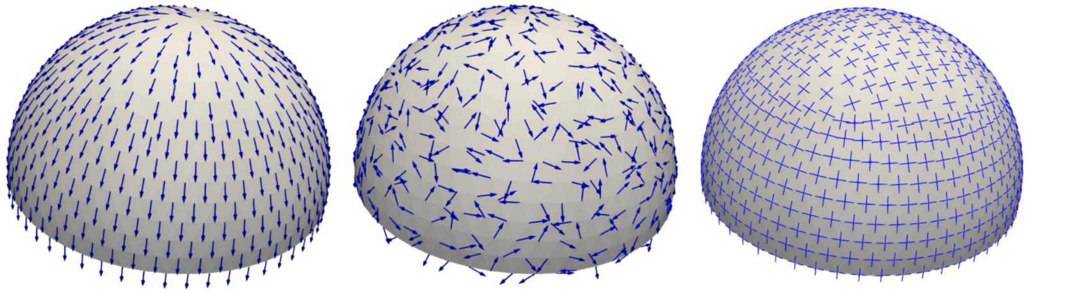

Figure 1: A comparison of fields and connections. **Left:** A smooth vector field as measured by the Levi-Civita connection. **Center:** A smooth vector field as measured by the connection ∇^Q. **Right:** The cross field corresponding to the vector field in the center.

We can write $c_p = e^{i(\theta_p + 2k\pi/4)}$ for some $k \in \{0, 1, 2, 3\}$ where θ_p is the signed angle from $\frac{\partial}{\partial x^1}|_p$ to one of the component vectors of $[c_p]$. We can now write equation 2 as

$$Q_{pq}(e^{4i\theta_p}) = e^{4i(\theta_p + \phi_{pq})}.$$

It follows that $Q_{pq}(v) = e^{4i\phi_{pq}}v$. In addition, we can define parallel transport on the cross bundle from C_p to C_q by $R_q^{-1} \circ Q_{pq} \circ R_p$. We can now use ∇^Q in equation 1 to define a smooth cross field. In the following section, we define the discrete Laplace equation corresponding to this energy and describe the diffusion generated method for cross field design in detail.

2.2 Discrete Formulation

On each node n_i of the input triangle mesh, T, a normal vector, $\vec{n}_i$, is computed as an average of the vectors normal to each adjacent face weighted by the tip angle at the node. This normal vector in turn defines the tangent space, T_i, at node n_i. We then arbitrarily select a vector in T_i, which we assign to be $\frac{\partial}{\partial x^1}|_i$. Then $\frac{\partial}{\partial x^2}|_i$ is the vector such that $\frac{\partial}{\partial x^1}|_i \times \frac{\partial}{\partial x^2}|_i = \vec{n}_i$.

Let e_{ij} be the edge connecting nodes n_i and n_j. We compute the value ϕ_i as the signed angle between the projection of e_{ij} onto T_i and $\frac{\partial}{\partial x^1}|_i$. We then store the value $\phi_{ij} = \phi_j - \phi_i$, on the edge e_{ij}.

We now define the discrete parallel transport function $Q_{ij}\colon T_i \to T_j$ by $Q_{ij}(u) = e^{4i\phi_{ij}}u$, which parallel transports representation vectors across e_{ij}. This allows us to compare two representation vectors $u \in T_i$ and $v \in T_j$ by

$$|v - Q_{ij}(u)|^2.$$

In order to state a well-defined problem, we apply a Dirichlet condition. In the case of closed manifolds, we arbitrarily fix the orientation of a single cross. In the case of a bounded manifold, we apply a Dirichlet boundary condition by fixing u_i on each boundary node. We make the convention that the cross field index of a boundary node (see Viertel and Osting [17]) is assigned according to Table 1.

Table 1: Boundary index assignment

Interior Angle	Index
$(0, \frac{3\pi}{4})$	$\frac{1}{4}$
$[\frac{3\pi}{4}, \frac{5\pi}{4}]$	0
$(\frac{5\pi}{4}, \frac{7\pi}{4}]$	$-\frac{1}{4}$
$(\frac{7\pi}{4}, 2\pi)$	$-\frac{1}{2}$

For a boundary node n_i of index 0 or $-\frac{1}{2}$, we compute the outward pointing unit normal vector of each boundary edge adjacent to n_i lying in the plane of the adjacent boundary face. We then project these vectors into the tangent plane at n_i, and bisect them with a unit vector, d_i, which averages the directions of the facet normals. Since we want to align the cross to this vector, we set $u_i = d_i^4$. For nodes of index $\pm\ \frac{1}{4}$, d_i is first rotated $\pi/4$ radians about $\mathbf{n}_i$ in the positive direction before computing u_i.

2.2.1 The Diffusion Generated Method for Cross Field Design

Intuitively, we would like to solve until stationarity the time dependent problem

$$\begin{aligned} u_t(t,x) &= \Delta u(t,x) && x \in M \\ u(t,x) &= g(x) && x \in \partial M \\ u(0,x) &= u_0(x) && x \in M \end{aligned} \tag{3}$$

with the constraint that $|u(x)| = 1$ pointwise, rather than directly solving the stationary problem with the same constraint. This key difference allows one to avoid the necessity of using a non-linear solver because the pointwise constraint can be enforced simply by normalizing the solution in between discrete timesteps (see [34, 35]). We proceed by defining a

Algorithm 2 A diffusion generated method for designing smooth cross fields

```
(i) Let $\vec{u}_0$ be the solution to $A\vec{u} = \vec{b}$. Let $\vec{u}_{-1} = \vec{0}$.
(ii) Fix $\tau$, $\delta$, and set $k = 0$.
while $\|\vec{u}_k - \vec{u}_{k-1}\| > \delta$, do
    (i) Solve the discrete diffusion equation,
        $(I - \tau A)\vec{u}_{k+1} = \vec{u}_k + \tau\vec{b}$    (6)
    (ii) Set $\vec{u}_{k+1}[2i] = \frac{\vec{u}_{k+1}[2i]}{\sqrt{\vec{u}_{k+1}[2i]^2 + \vec{u}_{k+1}[2i+1]^2}}$, $\vec{u}_{k+1}[2i+1] = \frac{\vec{u}_{k+1}[2i+1]}{\sqrt{\vec{u}_{k+1}[2i]^2 + \vec{u}_{k+1}[2i+1]^2}}$
    (iii) $k++$
end while
(iii) Set $u_i = \vec{u}_k[i] + i\vec{u}_k[i+1]$
```

discrete Laplacian operator, Δ_Q corresponding to the connection ∇^Q.

$$\Delta_Q(\vec{u})|_i = \frac{1}{|\mathcal{N}(n_i)|} \sum_{n_j \in \mathcal{N}(n_i)} (u_j - Q_{ij}(u_i)) \qquad (4)$$

where $\mathcal{N}(n_i)$ is the one-ring neighborhood of n_i and $|\mathcal{N}(n_i)|$ is the area of that neighborhood. This discrete Laplacian operator allows us to assemble a discrete diffusion equation using a backward Euler time discretization;

$$(I - \tau A)\vec{u} = \vec{u}_0 + \tau\vec{b}. \qquad (5)$$

Here $(I - \tau A)$ is a $2n \times 2n$ real symmetric positive definite matrix where n is the number of free nodes in T, and $\vec{u}$ and $\vec{b}$ are $2n \times 1$ column matrices containing the real and imaginary parts of the field. In algorithm 2, we define the iterations for cross field design.

3. GENERATION OF A QUAD LAYOUT WITH T-JUNCTIONS

After designing a cross field on a triangle mesh, the next step in algorithm 1 is to construct a quad layout with T-junctions. This is accomplished in two steps:

1. Determine singularity locations and ports.
2. Trace out separatrices of the cross field.

In section 3.1, we detail the first step of this process. In section 3.2, we describe our approach to streamline tracing, including our novel method for computing the trajectory of a streamline in the neighborhood of a singularity. In section 3.3, we discuss stopping conditions for the streamline tracing algorithm as well as a heuristic method for handling tangential streamline crossings that occur commonly when tracing streamlines via numerical integration.

3.1 Singularity and Port Detection

Singularity and port detection in a cross field defined per node is well documented in the literature [3, 28, 29, 11, 17]. We use methods that have been developed previously, but include a description here for completeness.

3.1.1 Matchings

Across each edge, we assume that crosses make the smallest rotation possible. This is called the *principle matching* of the crosses. If $u_i = e^{4i\theta_i}$, then the change in cross orientation between two nodes n_i and n_j, denoted Δ_{ij}, is the number between $-\frac{\pi}{4}$ and $\frac{\pi}{4}$ given by

$$\Delta_{ij} = (\theta_j - (\phi_{ij} + \theta_i))(\text{mod } \pi/2) - \pi/4.$$

3.1.2 Singularity Detection

The *index* of a triangle, t_{ijk}, with nodes n_i, n_j, and n_k is the number given by

$$\mathrm{I}\left(t_{ijk}\right) = \frac{\Delta_{ij} + \Delta_{jk} + \Delta_{ki} - \phi_{ij} - \phi_{jk} - \phi_{ki}}{2\pi}.$$

Practically, this is the number of turns that a cross makes while circulating the triangle, which is always an integer multiple of $\frac{1}{4}$. A triangle is *singular* if its index is non-zero. Summing the changes in cross orientation along each edge, we compute the total circulation of the cross around the triangle. On a flat surface, the total circulation must be a multiple of $\pi/2$; however, on curved surfaces, this is not the case. In general, a vector in the tangent bundle of M parallel transported along a closed curve does not always return to the same orientation after circulating the curve. To mitigate the effects of *holonomy* while transporting a cross around the triangle, we subtract from the total circulation the angle defect that occurs when parallel transporting a vector around the triangle via our discrete connection. This angle defect is given by $\phi_{ij} + \phi_{jk} + \phi_{ki}$. Subtracting the angle defect from the total circulation leaves us with a number that is a multiple of $\pi/2$, from which we compute the index.

After we determine singular triangles, we approximate the location of the singularity within the triangle by taking the barycenter of the triangle. We choose an arbitrary node, n, on the triangle and rotate the cross component vectors at that node into the plane of the face. We use the ray starting at the barycenter and passing through n as a reference axis, and compute the angle α that the nearest cross component vector makes with the reference axis. We then compute the angles where streamlines exit the singularity (ports) as $\alpha + \frac{2\pi k}{4-d}$ where $d/4$ is the index of the singularity (see [17]).

3.2 Separatrix Tracing

After designing a cross field on T, we build an initial quad partition by tracing separatrices of the cross field. Below, we describe the methods used for tracing separatrices in non-singular and singular triangles.

3.2.1 Non-Singular Triangles

In non-singular triangles t_{ijk}, we project the crosses on each corner of t_{ijk} into the plane of the triangle. We define a reference coordinate axis in that plane, and compute a representation vector for each cross with respect to this reference coordinate axis. We interpolate the argument of the representation vector linearly over the triangle. This is possible because in a non-singular triangle, the argument is continuous as it circulates the boundary. Using this interpolation, we trace out the streamlines using Heun's method [3].

3.2.2 Singular Triangles

Previously, no streamline tracing methods have been suitable for accurately tracing streamlines in the neighborhood of a singular triangle. Numerical integration methods such as Heun's method and other Runge-Kutta methods [36, 3, 11, 17] are inaccurate in the neighborhood of a singularity since cross directions can change arbitrarily fast. They frequently compute discretizations of the streamline that "cut the corner" rather than traversing around the singularity as they should (see Figure 2). Streamline tracing methods based on edge maps [9, 15] are guaranteed to never result in tangential crossings, but since paths are discretized by straight lines through each triangle, they are limited in their ability to resolve the path of a trajectory around a streamline by the number of triangles that meet at a singular point.

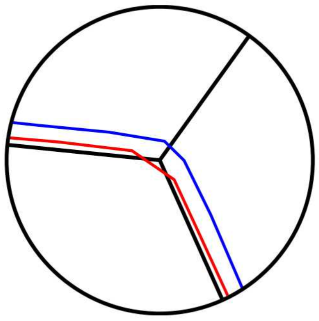

Figure 2: An illustration of a numerically traced streamline that "cuts the corner" and does not traverse around the singularity.

Here we develop a new method to accurately trace streamlines within the neighborhood of a singularity to any predefined resolution. This method guarantees that no tangential intersections of streamlines will occur within the neighborhood. We first prove that in $\mathbb{R}^2$, trajectories of streamlines through the neighborhood of a singularity are hyperbolas under a conformal transformation.

Let f be a canonical harmonic cross field (see [17]) on a domain $D \subset \mathbb{R}^2$. Let $a \in \mathbb{R}^2$ be the location of a singularity of index $\frac{d}{4}$ where d is an integer ≤ 1. Consider the open ball $B(a, r_0)$ of radius $r_0 > 0$ centered at a. We seek an approximation for the trajectory of an arbitrary streamline passing through a point $q \neq a$ in $B(a, r_0)$.

The cross field, $f(z)$, partitions $B(a, r_0)$ into $4 - d$ evenly angled sectors bounded by separatrices of the cross field [17]. In each sector, the cross field defines a local (u, v) parameterization. Let S be the open sector containing q, and let s_0 and s_1 be the separatrices bounding S ordered counterclockwise. Because the cross field defines a local (u, v) parameterization on S, there are two streamlines passing through q, one crossing s_0 orthogonally, the other crossing s_1 orthogonally. Without loss of generality, we consider γ, the streamline crossing s_1 orthogonally. This streamline crosses s_1 into S', the open sector adjacent to S that is also bounded by s_1, and then exits $B(a, r_0)$ (see Figure 3).

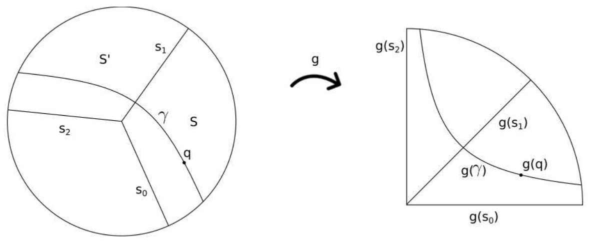

Figure 3: Streamlines in a neighborhood of a singularity become hyperbolas under a conformal map g.

By [17], the cross field in $B(a, r_0)$ can be written as

$$f(z) = e^{i(\frac{d\theta}{4} + \frac{2k\pi}{4})} + o(r)$$

where $d/4$ is the index of the singularity, $z = re^{i\theta}$, $k \in \{0, 1, 2, 3\}$, and $\theta = 0$ corresponds to s_0. For $r < r_0$ where r_0 is sufficiently small, we make the approximation

$$f(z) = e^{i(\frac{d\theta}{4} + \frac{2k\pi}{4})}$$

Streamlines of the cross field are given by

$$z' = f(z)$$

Since we are looking for the streamline crossing through s_1, we consider $k = 0$. Thus, we are looking for the set $C = \{z(t) \in B(a, r_0) \,|\, t \in (t_a, t_b)\}$ where $z(t)$ on (t_a, t_b) is a solution to the problem

$$z' = e^{i\frac{d\theta}{4}} \tag{7}$$
$$z(0) = (r_q, \theta_q) \tag{8}$$

in $D = \{z = re^{i\theta} \,|\, r \in (0, r_0),\, \theta \in (0, \frac{4\pi}{4-d})\}$.

Proposition 1. *$C = \{(x+iy)^{-(4-d)/8} \,|\, xy = A,\, x \in I_x\}$ for some constant A on some interval I_x.*

Proof. Consider a differentiable curve in D given by $z(t)$ for $t \in (a, b)$. Consider the function $g(z) = z^{(4-d)/8}$ that maps D to $\tilde{D}$, the sector of the upper right quadrant given by $\{w = \rho e^{i\varphi} \,|\, \rho \in (0, \rho_0 = r_0^{(4-d)/8}),\, \varphi \in (0, \frac{\pi}{2})\}$ (see Figure 3). Let $w(t) = g(z(t))$ for $t \in (a, b)$. Taking the derivative of both sides, we have

$$w'(t) = g'(z(t))z'(t)$$

since $g'(z) \neq 0$ in D, we have

$$\begin{aligned} \arg\left(g'(z(t))z'(t)\right) &= \arg g'(z(t)) + \arg z'(t) \\ &= \left(\frac{4-d}{8} - 1\right)\theta + \arg(z'(t)) \end{aligned}$$

In the case that $z(t)$ is a solution of equation 7, we have

$$\begin{aligned} \arg w'(t) &= \frac{d\theta}{4} + \left(\frac{4-d}{8} - 1\right)\theta \\ &= -\frac{(4-d)\theta}{8} = -\varphi \end{aligned}$$

Thus $w'(t) = \alpha(t)e^{-i\varphi}$ for some function $\alpha(t)$. Writing $w(t) = x(t) + iy(t)$, we have $x'(t) = \alpha(t)\cos(\varphi)$, $y'(t) = -\alpha(t)\sin(\varphi)$. Thus

$$\frac{dy}{dx} = -\tan(\varphi) = -\frac{y}{x} \implies y = \frac{A}{x}$$

for some constant A. This equation describes the family of hyperbolas in the first quadrant with asymptotes at $\varphi = 0$ and $\varphi = \frac{\pi}{2}$. The curve from this family passing through the point $g(q) = \rho_q e^{i\varphi_q}$ is given by $\{x + iy \,|\, xy = A_q\}$, where $A_q = \rho_q^2 \sin\varphi_q \cos\varphi_q$. The curve C can be recovered by taking the inverse image of this set under the mapping g, that is $C = \{(x+iy)^{-(4-d)/8} \,|\, xy = A_q,\, x \in I_x\}$ where I_x is an interval such that $(x+iy)^{-(4-d)/8} \in B(a, r_0)$ for $x \in I_x$. □

Proposition 1 provides a simple method for computing the trajectory of a streamline through a singular triangle. We make the assumption that within the triangle, the estimate

$$f(z) \approx e^{i\left(\frac{d\theta}{4} + \frac{2k\pi}{4}\right)}$$

holds. Here again $\theta = 0$ corresponds to the nearest separatrix clockwise from q. Making this assumption, we simply compute points along the hyperbola $xy = A_q$, and take the inverse image of each point. We use these points as discretization points of the streamline so long as they lay within the singular triangle. Since hyperbolas are convex, and g^{-1} preserves the order of points along rays, in order to guarantee that two streamlines don't intersect tangentially, it is sufficient to evaluate the points of the hyperbola along predefined rays from the singular point.

3.3 Partition Construction and Tangential Crossings

By the Poincaré-Bendixson theorem for manifolds [37], streamlines of a cross field on a bounded manifold M can do one of the following:

1. Connect one or more singularities in a homoclinic or heteroclinic orbit
2. Exit the boundary
3. Approach a limit cycle
4. Approach a limit set that is all of M. In this case, M must be a torus.

Because we are tracing out separatrices on a discrete mesh, in practice, they never line up perfectly with another singularity, so homoclinic and heteroclinic orbits will never occur. Thus streamlines in the discrete case can only either exit the boundary, or continue forever approaching either a limit cycle, or a limit set that is all of M. In order to generate a quad partition via separatrix tracing, separatrices that continue forever must be cut off after crossing some separatrix orthogonally. In practice, we use two stopping conditions: separatrices are traced until they either exit the boundary or cross the same separatrix more than once. The second condition is a simple way to eliminate the possibility of tracing out a separatrix forever, but can potentially create T-junctions on separatrices that would eventually exit the boundary.

When tracing streamlines using a numerical method such as Heun's method, there is no guarantee that streamlines won't cross each other or exit the boundary tangentially. This becomes especially problematic along boundaries of meshes where the underlying geometry has high curvature but few triangles, resulting in few crosses that are actually aligned with the discrete boundary of the triangle mesh. Tangential crossings are problematic because the regions produced via separatrix tracing are no longer guaranteed to be four-sided. Assuming a sufficiently fine triangle mesh along the boundary such that no separatrices exit tangentially, we observe in practice that tangential crossings on the interior typically occur in one of two cases. The first case is when one or more separatrices that approach a limit cycle are traced out for several rotations around the limit cycle. This problem is virtually eliminated by our approach of cutting off separatrices after they cross the same separatrix more than once.

The second case where tangential crossings occur is when there is a very small misalignment of singularities, such that two different separatrices follow virtually the same path. If the two separatrices are heading in opposite directions when the crossing occurs, then this problem can easily be fixed by cutting both separatrices off at the tangential crossing and combining them into a single separatrix, now connecting the two singularities in a heteroclinic orbit. If both separatrices are traveling in the same direction when they cross tangentially, there is no analogous simple operation to combine the two. However, we have observed that in practice, this almost always occurs when one of the separatrices passes very near the singularity where the other began. To mitigate the occurrence of tangential crossings when both separatrices are traveling in the same direction, we add a third stopping criteria for tracing separatrices: we cut off any separatrix inside of a singular triangle that crosses one of the separatrices leaving the singularity orthogonally. This third stopping condition greatly reduces the number of tangential crossings that occur when both streamlines are traveling in the same direction.

4. PARTITION SIMPLIFICATION

The misalignment of singularities when tracing out separatrices often results in small regions and limit cycles in the initial partition that would not exist if the separatrices coincided. In this section, we present a robust algorithm to simplify the partition obtained by naive separatrix tracing. The central step in the algorithm is an operation that extends the chord collapse operation for quad meshes [38, 39] to quad layouts with T-junctions.

A *chord* in a quad mesh is a maximal sequence of quads, $q_1, q_2, \ldots, q_n$ such that q_i is adjacent to q_{i+1}, and q_{i-1} and q_{i+1} are on opposite sides of q_i. Figure 4 shows a chord of a quad mesh highlighted in blue. A partition obtained from streamline tracing is a quad layout with T-junctions, or a *T-layout* for short. We say that each component of a T-layout has four total *sides*, 2 pairs that are opposite each other. A side consists of at least one edge or more when T-junctions occur on that side. A *chord of a T-layout* is a maximal sequence of components, $c_1, c_2, \ldots, c_n$ such that c_i is adjacent to c_{i+1}, c_{i-1} and c_{i+1} are on opposite sides of c_i, and no T-junction exists between c_i and c_{i+1}. Figure 5 top shows various chords in a T-layout.

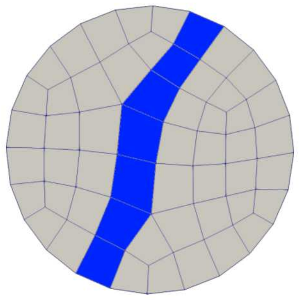
Figure 4: A chord in a quad mesh.

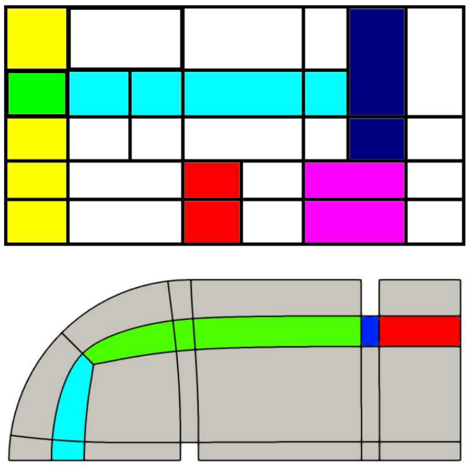
Figure 5: Illustrations of chords in two T-layouts. **Top:** Chords in a T-layout shown in various colors. The yellow and cyan chords overlap on the green component, illustrating how each component is part of two chords. **Bottom:** The four patches of a chord highlighted in cyan, green, blue, and red.

We call the set of edges shared by two partition components in a chord the *transverse rungs* of a chord. In the case that a chord begins or ends at a T-junction or on a boundary, we also include the first and last set of edges as transverse rungs of the chord. We also say that a chord has two *longitudinal sides* that consist of all the edges of the partition components that are orthogonal to the transverse rungs.

A *patch* is a maximal subset of consecutive components of a chord such that singularities only occur on the first and last transverse rungs. A chord is partitioned into one or more patches, and singularities can occur only on the corners of patches. Figure 5 bottom shows the patches of a chord.

Our definition of chord collapse on a T-layout is mo-

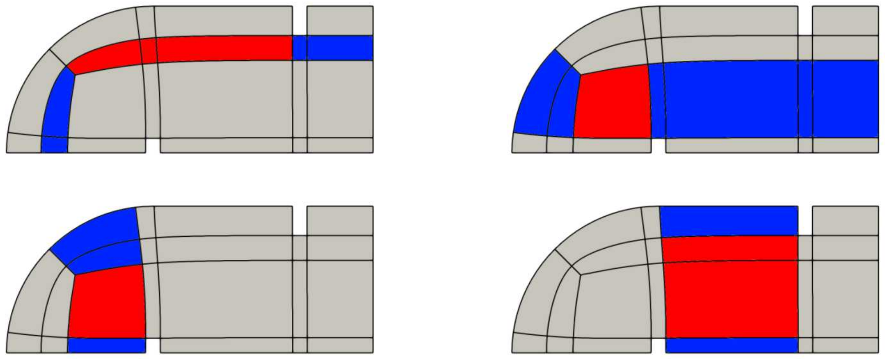

Figure 6: The four collapsible chords of a partition. Zip patches are highlighted in red and non-zip patches are highlighted in blue.

tivated by the goal of simplifying the partition by removing one separatrix from each of two singularities, and then connecting singularities together by a single curve. We say that a patch of a chord in a T-layout is *collapsible* if it satisfies the following:

1. No singularities are connected across any transverse rung of the patch.
2. No singularity is connected to a boundary across any transverse rung of the patch.
3. If the patch starts or ends at a T-junction, then one of the following must be satisfied:
   (a) The node opposite the T-junction on the same transverse rung is a singularity.
   (b) The node opposite the T-junction on the same transverse rung is another T-junction with the same orientation.
   (c) The node on the opposite corner of the patch from the T-junction is a singularity.

We say that a chord is *collapsible* if all of its patches are collapsible.

The first and second conditions prevent the possibility of having to combine two singularities into a single one or move a singularity to the boundary. They reflect an assumption that throughout the simplification process, we would like to keep the singularity set of the cross field and only modify the connectivity of the singularity graph. The third condition prevents the introduction of new T-junctions when collapsing chords or other invalid configurations such as a node with only two edges meeting at a corner. Figure 6 shows the collapsible chords for a given quad layout.

Given these assumptions, we can define a collapse operation on a collapsible chord. We define this operation by defining two sub-operations on patches. On a collapsible chord, any patch will either have singularities on opposite corners, or it will have one or two singularities only on one longitudinal side. We refer to the former as a *zip* patch, and the latter as a *non-zip* patch. The green patch in Figure 5 bottom is a zip patch and the other 3 are non-zip patches.

The collapse operation on a non-zip patch is to simply delete the edges on the longitudinal side without any singularities. The operation on a zip patch is to remove both longitudinal sides of the patch and replace them with a single line that connects the two singularities together. In practice, we take a weighted average of the two sides, figuratively "zipping" the two edges together to form the new line. If any T-junctions occur on a side that is deleted during a collapse, the hanging separatrix is simply extended after the collapse operation until it crosses the next separatrix. Figure 7 illustrates 3 consecutive chord collapses used to simplify a partition. The next chord to be collapsed in each frame has its zip patches highlighted in red and its non-zip patches highlighted in blue.

This collapse operation effectively replaces the two longitudinal sides of a chord with a single curve passing through each of the singularities on either side. It is easy to see that the resulting graph is still a T-layout because the local connectivity at singularities is not changed and other crossings of separatrices are either unaffected or simply removed (see the proof of theorem 5.4 in [17]). We summarize this section with the following proposition,

Proposition 2. *Each chord collapse operation removes a chord from the T-layout, resulting in another T-layout with the same irregular nodes on the boundary and interior. A series of collapses monotonically*

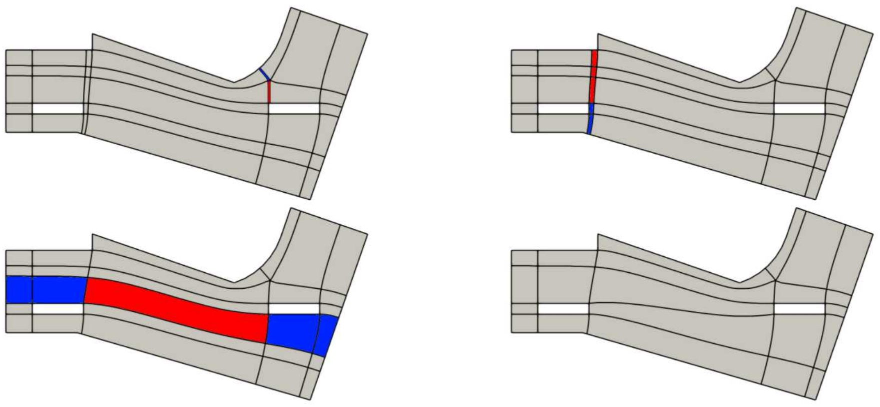

Figure 7: Three consecutive chord collapses simplify the quad layout. Zip patches to be collapsed at each step are colored in red and non-zip patches are colored in blue.

reduces the number of T-junctions in the layout, and strictly decreases the number of partition components.

This simple operation forms the core of our partition simplification algorithm. As Figures 7 and 8 illustrate, repeated application of this operation has the potential to dramatically simplify a T-layout obtained from separatrix tracing. We take a greedy approach, collapsing first the thinnest chord that satisfies all conditions for collapse. The full loop is described in algorithm 3.

Algorithm 3 Partition simplification

```
Let $\Gamma$ be the set of collapsible chords of the partition
while $|\Gamma| > 0$ do
  if No chords meet the conditions for collapse then
    Stop.
  else
    Collapse the chord with the smallest minimum width
    Determine new set of collapsible chords $\Gamma$
  end if
end while
```

4.1 Conditions for Collapse

It is not always beneficial to collapse every collapsible chord. Figure 6 highlights four collapsible chords in a partition, but by most measures, it would only be beneficial to collapse the thinnest of the chords, since collapsing the others would lead to severe deformation in the newly created partition components adjacent to the zipped edge. The decision of whether to collapse is also application dependent. For example, in the final chord collapse in Figure 7, the difference in length on opposite sides of the regions adjacent to the zipped separatrix may outweigh the cost of a slightly more complex partition, depending on the application.

A complete exploration of how different applications might benefit from various collapse conditions will not be treated here, rather we only describe the conditions used in our examples. We define an *energy* for each patch and we subsequently define the energy of the chord as the minimum energy of any of its patches. The collapse condition evaluates to true if the energy is positive and false if the energy is negative.

For a non-zip patch, we set the energy to a positive constant value. The exact value is not important, this simply reflects the notion that collapsing a non-zip patch is not detrimental to the overall quality of the partition.

For zip patches, let w be the mean of the length of each transverse rung of the patch. Let l be the mean of the length of each longitudinal side of the chord. The energy for the patch is then defined as

$$e = \frac{\pi}{8} - \arctan\frac{w}{l}.$$

If the zip patch were perfectly rectangular, then $\arctan\frac{w}{l}$ would be equivalent to the angle that the diagonal makes with the base of the rectangle. In rough terms, this condition prevents chord collapses that result in a large deformation of the angles that separatrices make at singularities.

We found this particular collapse condition and the heuristic of collapsing thinnest chords first to produce quads with more rectangular corners than other collapsing strategies that we tried. Figure 9 shows a comparison between the results of collapsing a given initial partition using the strategy that we describe versus the strategy of greedily collapsing chords via our chord collapse operation using an energy analogous to that used

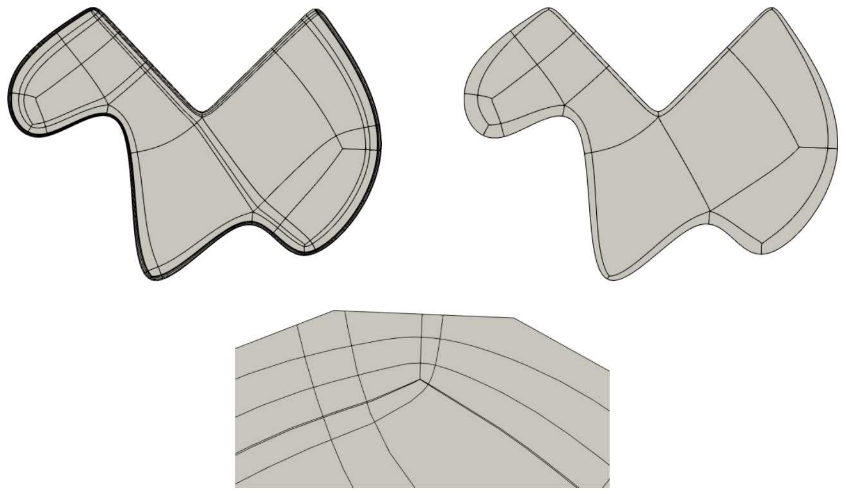

Figure 8: Before and after partition simplification. **Top Left:** The initial partition obtained by tracing separatrices. **Top Right:** A simplified partition after 10 chord collapses. **Bottom:** A close up of the top left corner of the geometry reveals extremely small components that occur because of misalignment in singularities in the initial partition.

in Tarini et al. [18] and Razafindrazaka et al. [12].

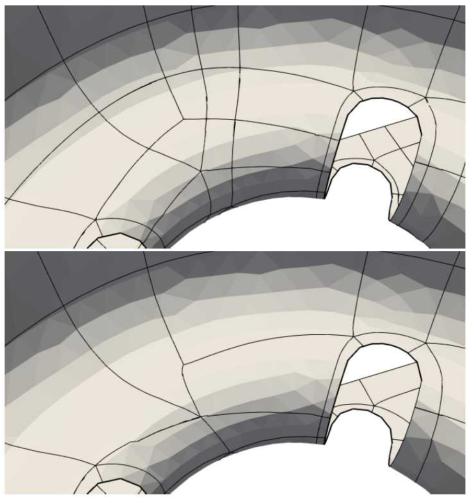

Figure 9: A comparison of collapse strategies. **Left:** The partition obtained by collapsing an initial partition according to the strategy defined in algorithm 3. **Right:** Result of collapsing the same partition using a greedy strategy collapsing chords in the order of highest to lowest energy using an energy analogous to that used in Tarini et al. [18] and Razafindrazaka et al. [12]. This strategy is over-aggressive in collapsing chords and we conclude that the energy does not work well with the chord collapse approach.

While we found the strategy of collapsing chords according to the conditions specified in this section to work well in our examples, it is a simple matter to substitute the sorting function and conditions for collapse in this algorithm with whatever is deemed appropriate for the application at hand.

5. NUMERICAL EXPERIMENTS

We tested our algorithm on 100 triangle meshes of surfaces with boundary derived from CAD models. For the diffusion generated method, we used a timestep $\tau = 1/\lambda_1$ where λ_1 is the first eigenvalue of the matrix A. We continued the iterations until $\|\vec{u}_k - \vec{u}_{k-1}\| < \sqrt{2n} \times 10^{-6}$ where n is the number of free nodes in the mesh. All examples were run on an Intel Core i5-2420m on a single thread.

In Figures 10 to 12 and Table 2, we present nine example models that are representative of the models used and results obtained in our experiment. Table 2 shows data for the number of nodes in the triangle mesh, timing for the diffusion generated method, streamline tracing, and partition simplification methods, the number of components and T-junctions before and after simplification, and the total number of chord collapses performed. Figures 10 to 12 show the initial partition obtained via streamline tracing on the left and the final simplified partition on the right. The models in the table are shown in the same order as they appear in the figures, and the horizontal lines in Table 2 identify the cutoffs between figures. All of the models except for the “faceplate” model are from a test suite used for development of the CUBIT software [21]. The “faceplate” model is the faceplate of the motor from the fan model at `https://grabcad.com/library/electric-fan-model-1`.

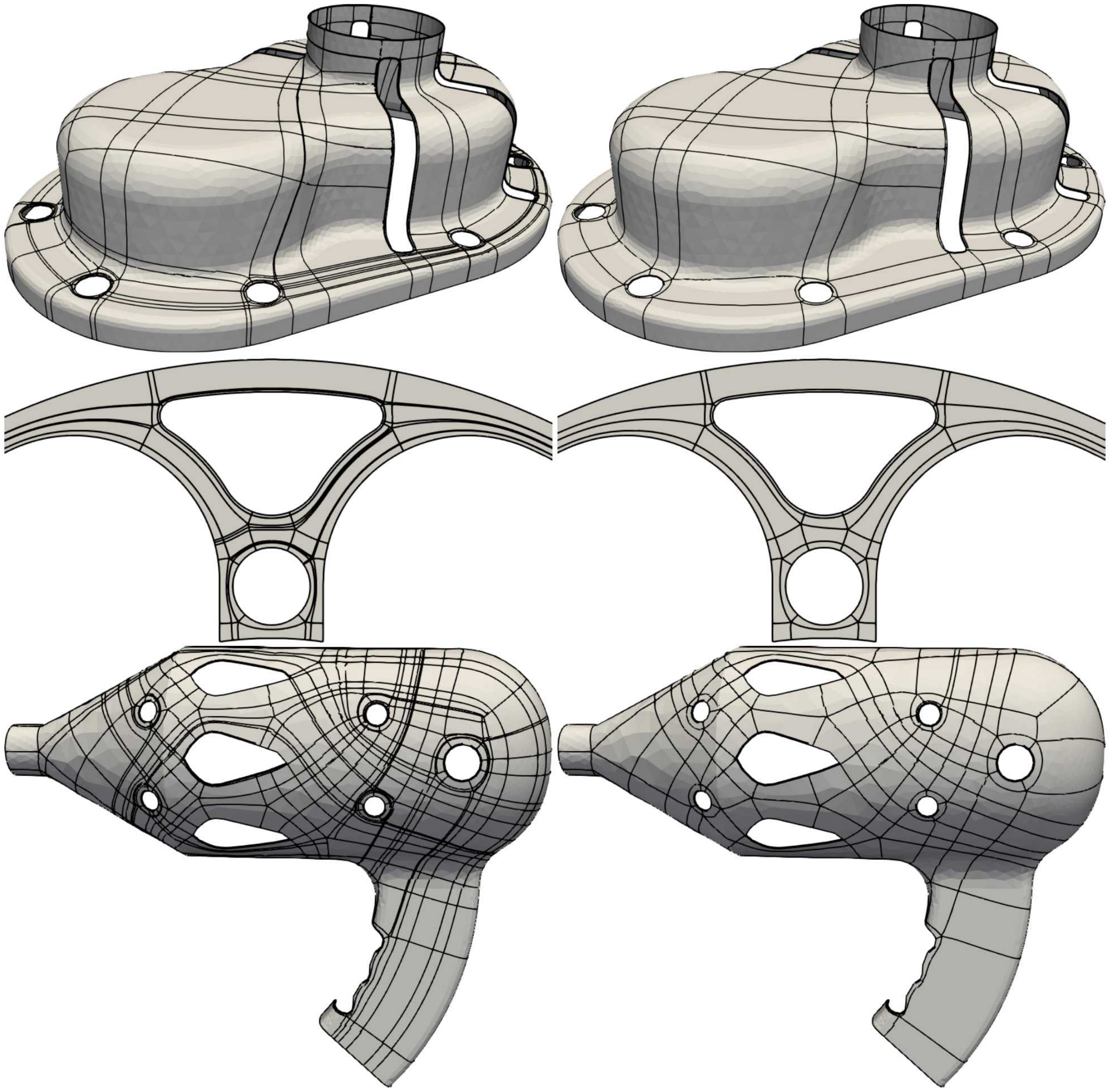

Figure 10: Examples of partitions simplified by our algorithm. The models from top to bottom are *cognit*, *chainr5*, and *gluegun*. **Left:** The initial partition obtained by tracing streamlines of the cross field obtained via the diffusion generated method. **Right:** A simplification of the partition on the left via our method.

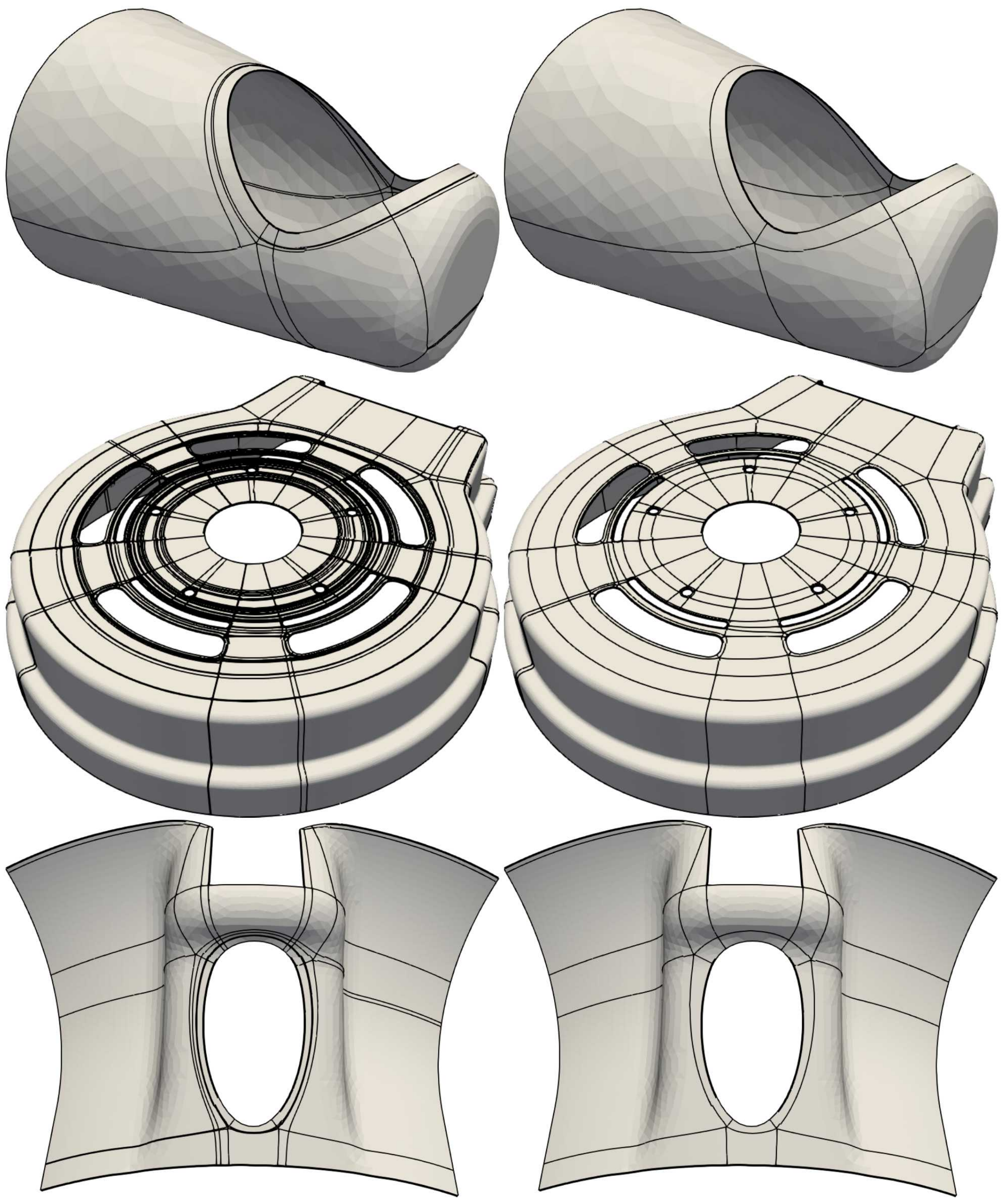

Figure 11: More examples showing the models *sprayer*, *faceplate*, and *part29*. See the caption for Figure 10.

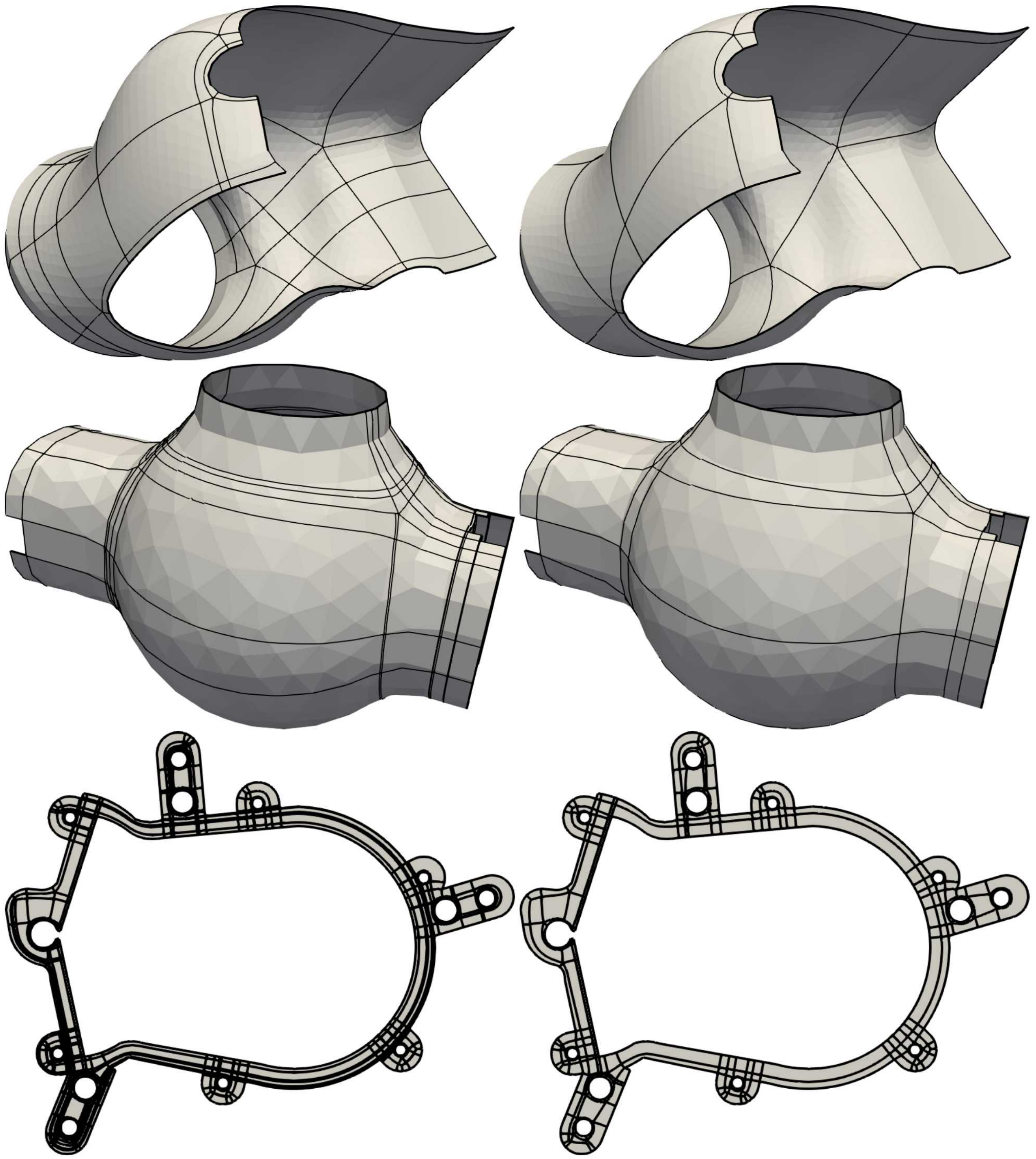

Figure 12: More examples showing the models *test1*, *engine2*, and *pump*. See the caption for Figure 10.

Overall, we observe that our algorithm performs well both in terms of efficiency and results. The timings reported for our cross field design method are comparable to those for the fastest cross field design methods [28, 29]. The timings for partition simplification reported in this paper are approximately an order of magnitude faster than those reported in [18] and [12] on similar sized models. Visually, the coarseness of the final quad layouts appear to be comparable across all three methods; however, a better comparison using the same models with each method is needed.

Out of the database of 100 models that we tested, eight models still had T-junctions after the simplification process. On four of those models, the T-junctions could be removed by simply continuing to trace the streamlines until they reached the boundary; see Figure 13 top. On the other four, there was at least one T-junction where the corresponding streamline approached a limit cycle. In each case that we observed, all T-junctions could have been removed from the initial partition by collapsing the chords in a different order (Figure 13 bottom), which suggests that perhaps a better collapse order would prioritize or even require collapsing chords that end in T-junctions.

In contrast to our method, the method of Razafindrazaka et al. [12] only considers port matchings that result in an all quad mesh, and in the cases where our method failed to remove T-junctions, would have succeeded. The main downside to their approach is that there is no way to guarantee that any solution exits to the perfect matching problem. Indeed, on the surface shown in Figure 14, there is no all quad layout for the set of singularities defined by the cross field. Where their method would fail, ours still produces a quad layout with a T-junction, which can still easily be meshed with a pattern-based technique.

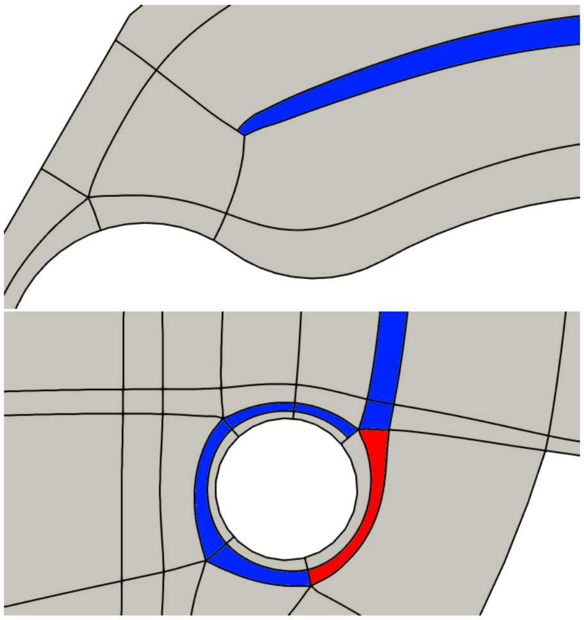
Figure 13: T-junctions not eliminated by our method. **Top:** A T-junction that can be removed simply by tracing the streamline until it reaches the boundary. The T-junction occured here because the separatrix intersected another separatrix while passing through a singular triangle. A collapsing a collapsible chord (highlighted in blue, full chord not shown) would remove the T-junction as well, but the energy condition for collapse was not met. **Bottom:** A T-junction resulting from a singularity approaching a limit cycle. Zip and non-zip patches of the chord adjacent to the T-junction are marked in red and blue respectively. The colored chord is not collapsible because further up the chord a singularity is opposite a transverse rung (not shown). In the initial partition, this T-junction could have been removed by a chord collapse without this obstruction.

6. DISCUSSION

In this paper, we have further developed three parts of the pipeline described in algorithm 1: an efficient method for high-quality cross field design, a method to accurately compute the trajectory of streamlines in the neighborhood of a singularity that avoids tangential crossings, and a robust partition simplification algorithm. We implemented a pipeline including these improvements, and executed our code on a database of 100 CAD surfaces. The resulting quad and T-layouts can be used as a starting point for meshing and parameterization.

The diffusion generated method is well suited for cross field design on CAD surfaces because it results in smooth boundary aligned cross fields with good singularity placement near the boundaries. It is also comparable in speed to the fastest cross field design methods; however, a more in-depth analysis is needed to fully compare the results.

Our novel method for tracing the trajectories of streamlines near singular points is simple and allows for accurate computation of streamlines near singularities while avoiding tangential crossings. Our implementation is, however, limited by our choice of a node-based cross field representation, as no methods exist to guarantee that streamlines away from singularities will not cross tangentially with such a representation. Our method could be improved by extending it to work in conjunction with streamline tracing methods such as Ray and Sokolov [15] and Myles et al. [9].

Our partition simplification algorithm is based on a simple chord collapse operation and is guaranteed to strictly decrease the number of partition components at each step as well as monotonically decrease the

Table 2: Basic data for our pipeline on several models

Model	n	Cross Field (s)	Tracing (s)	Simp. (s)	Components Before	Components After	T-junctions Before	T-junctions After	Chord Collapses
cognit	7274	0.444	0.274	0.450	583	190	86	0	72
chainr5	4781	0.105	0.318	0.494	440	176	88	0	61
gluegun	1842	0.074	0.124	0.254	725	189	36	0	45
sprayer	954	0.037	0.027	0.011	29	12	6	0	4
faceplate	47655	7.036	0.989	2.270	1500	227	120	0	101
part29	3265	0.121	0.051	0.023	66	22	3	0	6
test1	2703	0.115	0.044	0.027	47	19	1	0	5
engine2	564	0.025	0.053	0.057	194	63	11	1	15
pump	2592	0.067	0.249	0.821	1014	303	88	4	80

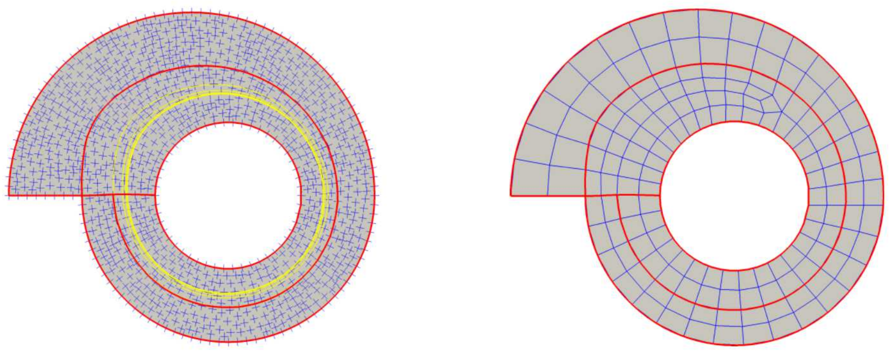

Figure 14: A surface where the cross field contains a limit cycle, resulting in a T-junction that cannot be removed without the introduction of additional singularities. **Left:** A boundary aligned cross field is shown in blue. The T-layout obtained by tracing out streamlines is shown in red. The yellow streamline begins at the geometric corner, follows the red curve of the partition, and continues on to converge to a limit cycle. **Right:** Because of the limit cycle, a pair of 3- and 5-valent nodes is needed to mesh the region adjacent to the T-junction.

number of T-junctions. It works directly on streamlines, so it does not require pre-meshing like methods such as Tarini et al. [18] and Bommes et al. [19], or computation of a Poisson parameterization like Razafindrazaka et al. [12]. Furthermore, it is more deliberate in its attempt to simplify the mesh than the partition simplification in Myles et al. [9], which is primarily intended to ensure global consistency of the parameterization.

Razafindrazaka et al. [12] formulate partition simplification as a minimum weight perfect matching problem on a graph. While construction of the graph (and consequently the solution space) in their method relies on heuristic choices, the solution space that they construct is at least as big as what could be obtained by considering all possible chord collapses in any order by our method. While they are able to find a globally optimal solution, we rely on a greedy algorithm to simplify the partition. At first glance, it appears that our method is at a disadvantage because of this; however, the two methods are hard to compare because it is not clear how the edge weights used in the formulation of Razafindrazaka et al. [12] reflect objectives such as minimizing the number of quad components, or maximizing the minimal width of a chord which can be pursued directly via our method. A specific application with objectively stated goals and a common set of models is needed for a clear comparison between the quality of the quad layouts generated via Razafindrazaka et al. [12] and our method.

The main downside of our partition simplification method is that it does not completely eliminate T-junctions from the layout. It is not always possible to produce a quad layout with no T-junctions for a given set of singularities, so an important direction for future work is to develop a method that uses strategic insertion or removal of singularities in order to guarantee that the elimination all T-junctions from a given input T-layout is possible. Such a result would be beneficial for applications in meshing for FEM, surface reconstruction into CAD from image data or geometries generated via topology optimization, and in constructing spline bases for isogeometric analysis.

ACKNOWLEDGEMENTS

B. Osting is partially supported by NSF DMS 16-19755 and 17-52202. R. Viertel and M. Staten are supported by Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] Lindquist D.R., Gilest M.B. "A Comparison of Numerical Schemes on Triangular and Quadrilateral Meshes." *11th International Conference on Numerical Methods in Fluid Dynamics*, pp. 369–373. Springer Berlin Heidelberg, 1989

[2] Sandia National Laboratories. "Trilinos 12.12.1.", 2017

[3] Kowalski N., Ledoux F., Frey P. "A PDE Based Approach to Multidomain Partitioning and Quadrilateral Meshing." *Proceedings of the 21st International Meshing Roundtable*, pp. 137–154. Springer Berlin Heidelberg, 2013

[4] MegaCads. "MegaCads 2.5.", 2001

[5] Tam T.K.H., Armstrong C.G. "2D Finite Element Mesh Generation by Medial Axis Subdivision." *Adv. Eng. Software*, vol. 13, no. 5, 313–324, Sep. 1991

[6] Gould J., Martineau D., Fairey R. "Automated Two-Dimensional Multi-Block Meshing Using the Medial Object." *Proceedings of the 20th International Meshing Roundtable*, pp. 437–452. Springer Berlin Heidelberg, 2012

[7] Fogg H.J., Armstrong C.G., Robinson T.T. "Multi-Block Decomposition Using Cross-Fields." *Adaptive Modeling and Simulation 2013*. International Center for Numerical Methods in Engineering (CIMNE), 2013

[8] Bommes D., Campen M., Ebke H.C., Alliez P., Kobbelt L. "Integer-Grid Maps for Reliable Quad Meshing." *ACM Trans. Graph.*, vol. 32, no. 4, 98:1–98:12, 2013

[9] Myles A., Pietroni N., Zorin D. "Robust Field-Aligned Global Parametrization." *ACM Trans. Graph.*, vol. 33, no. 4, 135:1–135:14, 2014

[10] Campen M., Bommes D., Kobbelt L. "Quantized Global Parametrization." *ACM Trans. Graph.*, vol. 34, no. 6, 192:1–192:12, Oct. 2015

[11] Fogg H.J., Armstrong C.G., Robinson T.T. "Automatic Generation of Multiblock Decompositions of Surfaces." *Int. J. Numer. Meth. Engng.*, vol. 101, no. 13, 965–991, 2015

[12] Razafindrazaka F.H., Reitebuch U., Polthier K. "Perfect Matching Quad Layouts for Manifold Meshes." *Comput. Graph. Forum*, vol. 34, no. 5, 219–228, Aug. 2015

[13] Campen M. "Partitioning Surfaces Into Quadrilateral Patches: A Survey." *Comput. Graph. Forum*, vol. 36, no. 8, 567–588, 2017

[14] Hertzmann A., Zorin D. "Illustrating Smooth Surfaces." *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 517–526. ACM Press/Addison-Wesley Publishing Co., 2000

[15] Ray N., Sokolov D. "Robust Polylines Tracing for N-Symmetry Direction Field on Triangulated Surfaces." *ACM Trans. Graph.*, vol. 33, no. 3, 30:1–30:11, 2014

[16] Eppstein D., Goodrich M.T., Kim E., Tamstorf R. "Motorcycle Graphs: Canonical Quad Mesh Partitioning." *Proceedings of the Symposium on Geometry Processing*, pp. 1477–1486. Eurographics Association, 2008

[17] Viertel R., Osting B. "An Approach to Quad Meshing Based on Harmonic Cross-Valued Maps and the Ginzburg–Landau Theory." *SIAM J. Sci. Comput.*, vol. 41, no. 1, A452–A479, Jan. 2019

[18] Tarini M., Puppo E., Panozzo D., Pietroni N., Cignoni P. "Simple Quad Domains for Field Aligned Mesh Parametrization." *ACM Trans. Graph.*, vol. 30, no. 6, 142:1–142:12, Dec. 2011

[19] Bommes D., Lempfer T., Kobbelt L. "Global Structure Optimization of Quadrilateral Meshes." *Comput. Graph. Forum*, vol. 30, no. 2, 375–384, Apr. 2011

[20] Razafindrazaka F.H., Polthier K. "Optimal Base Complexes for Quadrilateral Meshes." *Comput. Aided Geom. D.*, vol. 52-53, 63–74, Mar. 2017

[21] Sandia National Laboratories. "The CUBIT Geometry and Mesh Generation Toolkit 15.3.", 2017

[22] Merriman B., Bence J.K., Osher S.J. "Diffusion Generated Motion by Mean Curvature." *The Computational Crystal Grower's Workshop*, pp. 73–83. AMS Selected Lectures in Mathematics, 1993

[23] Palacios J., Zhang E. "Rotational Symmetry Field Design on Surfaces." *ACM Trans. Graph.*, vol. 26, no. 3, 55:1–55:10, Jul. 2007

[24] Ray N., Vallet B., Li W.C., Lévy B. "N-Symmetry Direction Field Design." *ACM Trans. Graph.*, vol. 27, no. 2, 10:1–10:13, 2008

[25] Ray N., Vallet B., Alonso L., Lévy B. "Geometry-Aware Direction Field Processing." *ACM Trans. Graph.*, vol. 29, no. 1, 1:1–1:11, 2009

[26] Bommes D., Zimmer H., Kobbelt L. "Mixed-Integer Quadrangulation." *ACM Trans. Graph.*, vol. 28, no. 3, 77:1–77:10, Jul. 2009

[27] Crane K., Desbrun M., Schröder P. "Trivial Connections on Discrete Surfaces." *Comput. Graph. Forum*, vol. 29, no. 5, 1525–1533, 2010

[28] Knöppel F., Crane K., Pinkall U., Schröder P. "Globally Optimal Direction Fields." *ACM Trans. Graph.*, vol. 32, no. 4, 59:1–59:10, 2013

[29] Jakob W., Tarini M., Panozzo D., Sorkine-Hornung O. "Instant Field-Aligned Meshes." *ACM Trans. Graph.*, vol. 34, no. 6, 189:1–189:15, 2015

[30] Vaxman A., Campen M., Diamanti O., Panozzo D., Bommes D., Hildebrandt K., Ben-Chen M. "Directional Field Synthesis, Design, and Processing." *Comput. Graph. Forum*, vol. 35, no. 2, 545–572, 2016

[31] Huang Z., Ju T. "Extrinsically Smooth Direction Fields." *Comput. Graph.*, vol. 58, 109–117, 2016

[32] Beaufort P.A., Lambrechts J., Henrotte F., Geuzaine C., Remacle J.F. "Computing two dimensional cross fields - A PDE approach based on the Ginzburg-Landau theory." *Proceedings of the 26th International Meshing Roundtable*, pp. 219–231. Elsevier Ltd., 2017

[33] Bethuel F., Brezis H., Hélein F. *Ginzburg-Landau Vortices*, vol. 13 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhäuser Boston, 1994

[34] Ruuth S.J., Merriman B., Xin J., Osher S. "Diffusion-Generated Motion by Mean Curvature for Filaments." *J. Nonlinear Sci.*, vol. 11, no. 6, 473, 2001

[35] Laux T., Yip N.K. "Analysis of Diffusion Generated Motion for Mean Curvature Flow in Codimension Two: A Gradient-Flow Approach." *Arch. Rational Mech. Anal.*, vol. 232, no. 2, 1113–1163, May 2019

[36] Zhang E., Mischaikow K., Turk G. "Vector Field Design on Surfaces." *ACM Trans. Graph.*, vol. 25, no. 4, 1294–1326, 2006

[37] Schwartz A.J. "A Generalization of a Poincaré-Bendixson Theorem to Closed Two-Dimensional Manifolds." *Amer. J. Math.*, vol. 85, no. 3, 453–458, 1963

[38] Borden M.J., Benzley S.E., Shepherd J.F. "Hexahedral Sheet Extraction." *Proceedings of the 11th International Meshing Roundtable*, pp. 147–152. Springer-Verlag, 2002

[39] Daniels J., Silva C.T., Shepherd J., Cohen E. "Quadrilateral Mesh Simplification." *ACM Trans. Graph.*, vol. 27, no. 5, 148:1–148:9, Dec. 2008