Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2017) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

26th International Meshing Roundtable, IMR26, 18-21 September 2017, Barcelona, Spain

Toward a paver replacement

Ryan Viertel[a,b,*], Matt Staten[a], Braxton Osting[b]

[a] *Sandia National Laboratories[1], 1515 Eubank SE, Albuquerque, NM 87123, USA*
[b] *University of Utah, 201 Presidents Cir, Salt Lake City, UT 84112, USA*

Abstract

In recent years, cross field guided quad meshing algorithms have found success in computer graphics applications. These methods have the capability of producing high quality block structured surface meshes. As most of these algorithms have not been designed for meshing on CAD surfaces for solid mechanics and other finite element applications, little attention has been paid to common requirements in these applications such as strict boundary alignment on all geometric curves and conformation to prescribed boundary node placement. We seek to design an algorithm capable of producing high quality block structured meshes on arbitrary surfaces such that the mesh conforms to a predetermined boundary interval assignment and follows a sizing function on the interior of the domain. Such an algorithm would be an ideal replacement for unstructured methods commonly used for finite element meshing such as the paving algorithm and its variants. In this note we outline a strategy that we expect will be able to meet these requirements. For each step in the pipeline, we discuss solutions which are currently available as well as open problems that will need to be resolved.

Peer-review under responsibility of the scientific committee of the 26th International Meshing Roundtable.

Keywords: frame field, cross field, quad mesh, paving

1. Introduction and Background

Cross field guided quad meshing has been studied for computer graphics applications for nearly a decade [1]. The field has reached a level of maturity that open source implementations of robust algorithms for cross field design and quad meshing are available and have been used in commercial graphics applications [2,3]. Recently these algorithms have been applied to meshing for finite element methods [4]. In practice, however, heuristic methods such as paving [5] continue to be used for meshing CAD surfaces for solid mechanics and other finite element applications. Paving has been a workhorse in quad meshing for several decades. It has proven successful because of its robustness in meshing arbitrary surfaces, and its ability to conform to predetermined boundary interval assignments and sizing functions, however paving results in meshes with little structure, and the algorithm is not deterministic. On the other

[1] 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. SAND2017-8620 C
* Corresponding author. Tel.: +1-801-669-4887
E-mail address: rvierte@sandia.gov

Peer-review under responsibility of the scientific committee of the 26th International Meshing Roundtable.

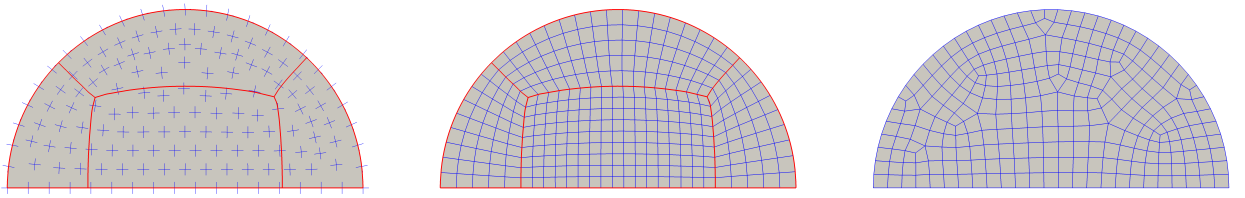

Fig. 1. Cross field guided meshing using streamlines. (**left**) A smooth boundary-aligned cross field is designed and partitioned into four-sided regions by tracing streamlines. (**center**) Boundary intervals are assigned and each region is meshed. (**right**) For comparison the same surface is meshed with the paving algorithm.

hand, cross field meshing algorithms are deterministic and produce block structured meshes, however we are not aware of any single cross field guided quad meshing algorithm that satisfies all of the common quad meshing requirements for solid mechanics and structural dynamics finite element analysis.

Our goal is to design an algorithm to generate quad meshes on arbitrary surfaces that meet the following objectives:

M.1 Block structure

M.2 Boundary aligned elements

M.3 High element quality

M.4 Elements abide a prescribed size map

M.5 Elements conform to prescribed node placement on the boundary

In addition, we want the algorithm to be fully automatic and deterministic, meaning that the resulting mesh only depends on geometry and user specifications, not on arbitrary choices made within the algorithm.

Such an algorithm would be an ideal replacement for paving in meshing for finite element analysis. We expect that an algorithm that extends existing cross field guided quad meshing algorithms will be able to satisfy objectives **M.1** - **M.5**. We define a *cross* as an unordered set of vectors $\{v_1, v_2, v_3, v_4\}$ such that $|v_i \cdot v_j|$ is zero or one, and all four vectors lie in the same plane. A *cross field* on a surface M is an assignment of a cross c to each but a finite number of points of M where each vector of c lies in the plane tangent to M at the point it was assigned. The basic idea of cross field meshing algorithms is to design a cross field that is boundary aligned and varies as smoothly as possible throughout the surface and use this field to guide the placement of quad elements; see fig. 1.

Meshing is typically accomplished in one of two ways. The first is to generate a parameterization that is as aligned as possible to the cross field in a least squares sense, and then use the parameterization to map a regular grid from the parameter space onto the surface [1,6,7]. A disadvantage of this method is that a globally consistent parameterization cannot always be found, and even when it can, producing a conformal mesh requires integer variables, making the problem NP-hard [7]. The libQEX mesh extraction library addresses this problem and is able to extract a mesh even from imperfect parameterizations [8].

The second type of cross field guided quad meshing methods partition a surface into four-sided regions by tracing out *streamlines* of the cross field [4,9,10]. A streamline of a cross field is a continuously differentiable parameterized curve, $\gamma(t)$ on M such that $d\gamma/dt$ is parallel to one of the four cross vectors at $\gamma(t)$ for all values of t. Viertel and Osting [10] recently examined the mathematics of the cross field design and quad meshing problems in detail, and proved under certain assumptions on the domain that it is always possible to partition a surface into four-sided regions. The resulting partition is a quad layout with T-junctions, where T-junctions occur exactly when a limit cycle appears in the cross field; see fig. 2.

Partitioning a surface into four-sided regions greatly simplifies the meshing problem as each region can be mapped with a regular grid or a pattern based technique; see section 2.3. The resulting meshes are boundary aligned, have high quality elements, and have few singularities, satisfying objectives **M.1** - **M.3**. In the following section, we outline the steps that we expect will be needed to design a robust, deterministic, fully-automatic algorithm that in addition will satisfy **M.4** and **M.5**.

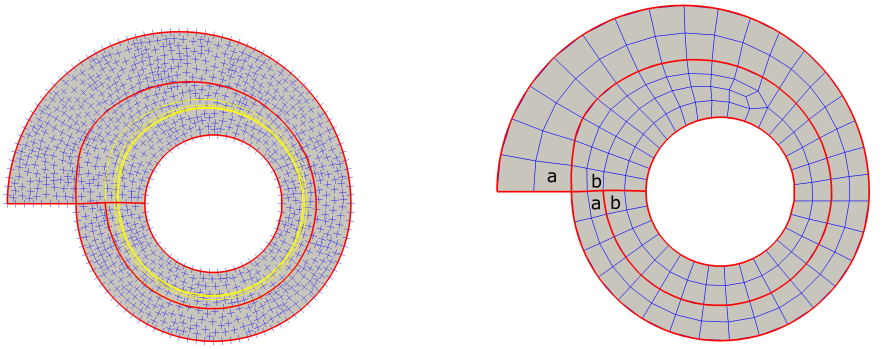

Fig. 2. A surface where the cross field contains a limit cycle. (**left**) A boundary aligned cross field is shown in blue. The quad layout obtained by the partitioning algorithm in [10] is shown in red. The yellow streamline begins at the geometric corner, follows the red curve of the partition and 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.

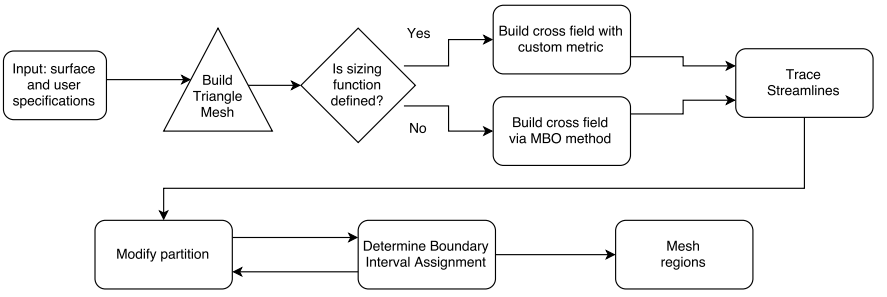

Fig. 3. Flow chart outlining the algorithm.

2. Method

In this section, we present an outline of how we expect that the algorithm will look. Figure 3 shows a flow chart connecting each of the steps of the algorithm. As input we take a surface embeddd in $\mathbb{R}^3$ as well as user input such as boundary node placement and preferred element size. As a preprocessing step we generate a triangle mesh of the surface for use in computation. A cross field will be generated on the surface and streamlines of the cross field traced out to partition the surface into a quad layout. Some modification of that quad layout might be necessary in order to conform with prespecified node locations on the boundary, and to simplify the structure of the mesh. Next, the number of intervals on each curve of the quad layout needs to be assigned in such a way to satisfy soft and hard constraints on the boundary, and the "sum even" constraint required for quad meshing. Finally, each region will be meshed using pattern-based techniques. Each of these steps is further explained in the subsections below.

2.1. Cross Field Design

The first step of the meshing algorithm is to design a cross field on the surface. There are several methods available in the literature, as well as open source implementations; see [2,6,12] and [11] for a recent review. Our first choice for a cross field design algorithm would be an extension of the MBO method described in [10]. In this approach, crosses are represented with the N-Rosy representation [13]. An initial cross field (typically a pointwise normalized solution to Laplace's equation) is evolved by diffusion for a short time, and then each vector is renormalized. These iterations continue until convergence. Viertel and Osting [10] use the Ginzburg-Landau theory to prove that this approach results in a harmonic cross field that can always be partitioned into four-sided regions by tracing streamlines. The MBO method is also straightforward to extend to surfaces. The main challenge is that there is no global representation

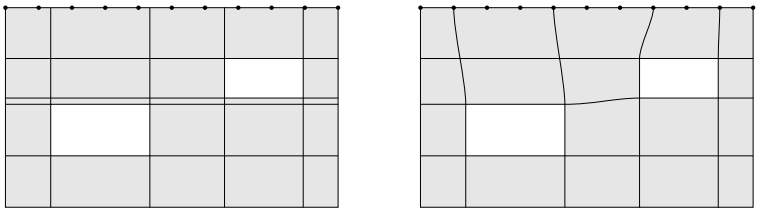

Fig. 4. Two decompositions of a geometry where the cross field is aligned with the x and y directions at every point. (**left**) The naive decomposition obtained by tracing streamlines from the boundary corners does not respect the boundary node placement at the top of the surface and results in very thin regions due to the placement of the two holes. (**right**) The modified decomposition respects the boundary node placement and simplifies the decomposition.

for crosses on non-planar surfaces. However this same issue is encountered in other cross field design algorithms for non-planar surfaces and is overcome by making use of the concept of a *discrete connection*; see [11].

Sizing Function

If a quad element sizing function is defined on the surface, a cross field can be designed that takes this sizing information into account. Panozzo et al. [14] encode such information by allowing the size of the crosses to change. After designing the field, the surface and crosses are then deformed by stretching until each cross has unit size, then the deformed surface is meshed, and the mesh is mapped back onto the original surface. Jiang et al. [15] address the problem by designing a custom Riemannian metric on the surface, and then design a smooth cross field with respect to this custom metric rather than the standard Eucliedean metric. They then generate a parameterization on the surface and use libQEX to extract a mesh. We intend to use the strategy in [15] to build cross fields that respect a sizing function, however we prefer to use streamline tracing methods to obtain a partition because it allows more control over the structure of the final mesh. Combining this type of approach with streamline tracing methods to build a quad layout is an open problem, however we expect that the extracted quad layout will be such that mapping a regular grid into each region will satisfy objective **M.4**.

2.2. Partitioning into Four-Sided Regions

After designing the cross field, the next step is to obtain a quad layout by tracing out streamlines of the cross field. For visualization purposes, streamlines are often computed using a fourth-order Runge-Kutta method. Using this approach, it is possible that computed streamlines will cross each other due to numerical inaccuracies. Myles et al. [9] and Ray and Sokolov [16] independently developed robust methods for streamline tracing that overcomes this issue, and we plan to implement one of their methods.

Partition Modification

Another challenge with streamline tracing is to determine numerical tolerances for combining streamlines or snapping streamlines to a boundary node. It is possible for two streamlines to run parallel to each other at a distance smaller than the target element width. In such a case the streamlines could be combined into one; see fig. 4. Further, in order to meet objective **M.5** it may sometimes be necessary to snap a streamline exiting a geometric boundary to a node. For example, this will be necessary when the boundary curve is pre-meshed and no new nodes can be created. It may also be desireable at times to move the location where a streamline crosses another curve of the partition in order to match a preferred element size. In this case the modification would be dependent on the boundary interval assignment described in section 2.3. Modifying the partition in a robust and efficient manner is another open problem.

2.3. Boundary Interval Assignment and Meshing

After the partition is created, it is necessary to assign intervals to the boundaries of each region. In the case where no T-junctions appear in the partition and user specified boundary constraints are consistent with mapping, the boundary interval assignment becomes trivial and each region can be meshed by mapping. When T-junctions occur, it

is no longer possible to mesh each region with a regular grid because, as can be seen in fig. 2, this leads to a condition on the number of quad cells per side such as $a + b = b$, $a > 0$, $b > 0$ which is clearly impossible to satisfy.

In cases where T-junctions occur or when user specified boundary conditions prohibit a mapped mesh, additional irregular nodes must be introduced in order to mesh each region. The location of these nodes will depend on the boundary interval assignment on each of these regions. Depending on the application and the user's preference on how these irregular nodes are distributed, an algorithm such as [17] could be modified to address the specific goals and constraints arising in this problem.

Once the boundary intervals are determined, each region can be meshed either with a regular grid, or by using the method described in [18], which is capable of meshing N-sided regions for $N \in \{2, 3, 4, 5, 6\}$ with arbitrary boundary interval assignments that satisfy the sum even constraint.

3. Discussion and Future Directions

Cross field guided quad meshing is a promising new approach to quadrilateral meshing. Theoretical guarantees and robust implementations demonstrate that cross field guided methods are capable of building quad meshes with many desireable properties for finite element analysis. Our goal is to extend existing algorithms to be capable of satisfying each of the objectives **M.1** - **M.5**. In this note we have outlined a method which we expect will be able satisfy **M.1** - **M.5**, as well as the open problems in this pipeline which will need to be solved.

While many methods exist for designing cross fields on surfaces, little work has been done to analyze the numerical properties of these methods. One direction for future work would be a careful numerical analysis of the finite element problem arising in the discretization of the MBO method for cross field design [10]. Another future direction along these lines would be a numerical analysis of streamline tracing methods for piecewise linear cross fields.

Another interesting problem is to extend the theory of cross fields to 3D frame fields for automatic hexahedralization. This problem however is significantly more difficult, and at present, the best that can be reliably obtained is a hex dominant mesh [20,21].

References

[1] F. Kälberer, M. Nieser, K. Polthier, QuadCover - surface parameterization using branched coverings, Comp. Graph. Forum 26 (2007) 375–384.
[2] W. Jakob, M. Tarini, D. Panozzo, O. Sorkine-Hornung, Instant field-aligned meshes, ACM TOG 34 (2015).
[3] A. Jacobson, D. Panozzo, et al., libigl: A simple C++ geometry processing library, 2016. Http://libigl.github.io/libigl/.
[4] N. Kowalski, F. Ledoux, P. Frey, A PDE based approach to multidomain partitioning and quadrilateral meshing, in: Proceedings of the 21st International Meshing Roundtable, 2013, pp. 137–154.
[5] T. D. Blacker, M. B. Stephenson, Paving: A new approach to automated quadrilateral mesh generation, Int. J. Num. Meth. Eng. 32 (1991) 811–847.
[6] D. Bommes, H. Zimmer, L. Kobbelt, Mixed-integer quadrangulation, ACM TOG 28 (2009).
[7] D. Bommes, M. Campen, H.-C. Ebke, P. Alliez, L. Kobbelt, Integer-grid maps for reliable quad meshing, ACM TOG 32 (2013).
[8] H.-C. Ebke, D. Bommes, M. Campen, L. Kobbelt, QEx: Robust quad mesh extraction, ACM TOG 32 (2013).
[9] A. Myles, N. Pietroni, D. Zorin, Robust field-aligned global parametrization, ACM TOG 33 (2014).
[10] R. Viertel, B. Osting, An approach to quad meshing based on harmonic cross-valued maps and the Ginzburg-Landau theory, submitted, arXiv:1708.02316 (2017).
[11] A. Vaxman, M. Campen, O. Diamanti, D. Panozzo, D. Bommes, K. Hildebrandt, M. Ben-Chen, Directional field synthesis, design, and processing, Comp. Graph. Forum 35 (2016) 545–572.
[12] F. Knöppel, K. Crane, U. Pinkall, P. Schröder, Globally optimal direction fields, ACM TOG 32 (2013).
[13] J. Palacios, E. Zhang, Rotational symmetry field design on surfaces, ACM TOG 26 (2007).
[14] D. Panozzo, E. Puppo, M. Tarini, O. Sorkine-Hornung, Frame fields: Anisotropic and non-orthogonal cross fields, ACM TOG 33 (2014).
[15] T. Jiang, X. Fang, J. Huang, H. Bao, Y. Tong, M. Desbrun, Frame field generation through metric customization, ACM TOG 34 (2015).
[16] N. Ray, D. Sokolov, Robust polylines tracing for n-symmetry direction field on triangulated surfaces, ACM TOG 33 (2014).
[17] S. A. Mitchell, High fidelity interval assignment, International Journal of Computational Geometry & Applications 10 (2000) 399–415.
[18] K. Takayama, D. Panozzo, O. Sorkine-Hornung, Pattern-based quadrangulation for n-sided patches, Comp. Graph. Forum 33 (2014) 177–184.
[19] S. J. Owen, M. L. Staten, S. A. Canann, S. Saigal, Q-morph: an indirect approach to advancing front quad meshing, Int. J. Num. Meth. Eng. 44 (1999) 1317–1340.
[20] D. Sokolov, N. Ray, L. Untereiner, B. Lévy, Hexahedral-dominant meshing, ACM TOG 35 (2016).
[21] X. Gao, W. Jakob, M. Tarini, D. Panozzo, Robust hex-dominant mesh generation using field-guided polyhedral agglomeration, ACM TOG 36 (2017).