Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2014) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

QMCF: QMorph Cross Field-Driven Quad-Dominant Meshing Algorithm

Bertrand Pellenard[a], Gunay Orbay[a], James Chen[a], Shailendra Sohan[a], Wa Kwok[a], Joseph R. Tristano[a]

[a] *ANSYS Inc., 275 Technology Dr, Canonsburg, PA 15317, USA*

Abstract

We propose a new method for generation of quadrilateral meshes. Our algorithm takes a triangle mesh generated on a computer-aided design surface together with a size field as input. A novel cross field, defined on vertices of the triangle mesh, is derived from both curvature directions and constraints on the boundary. Our algorithm uses a modified QMorph algorithm for quad dominant meshing, to follow size field and to align edges with the cross field previously computed.

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

Keywords: Quad-dominant meshing; Cross field; QMorph; Advancing front;

1. Introduction

Quadrilateral meshes are preferred over triangular meshes due to their numerical accuracy for both implicit and explicit finite element methods. They are also the preferred mesh in both CAGD and computer graphics. However, the automatic generation of a high quality quad mesh aligned both with boundaries and with principal curvature directions is still a computational challenge. This paper seeks to simplify this computational challenge by 1) extending current state of the art cross field computation techniques to account for prescribed constraints and 2) utilizing a cross field and size field in the QMorph algorithm such that generated quad meshes conform to both size and directions.

Problem statement. Assume a piecewise smooth surface given by CAD domain $\mathcal{D}$ (with or without boundary) and a background surface triangle mesh $\mathcal{S}$ approximating $\mathcal{D}$. We want to generate a mesh for which elements (i) are nearly square-shaped, (ii) are nearly planar, (iii) are sized in accordance to a user-specified size field, (iv) are oriented in accordance to a cross field, (v) have edges that are aligned with the

E-mail address: {bertrand.pellenard, gunay.orbay, james.chen, shailendra.sohan, wa.kwok, joe.tristano}@ansys.com

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

domain boundary. A more global requirement is that meshes should predominantly be composed of quads and regular vertices.

1.1. Previous Work

The tradeoff between local and global criteria may explain the variety of approaches proposed for the automatic generation of quadrilateral meshes: square packing [15], clustering [3,9], local and global operators [8,13], streamlining [1,10], advancing front [2,11] and parameterization [4,5,14,16].

Among these approaches, some can be classified as global as they minimize a global energy, such as the parameterization-based method ([4,5,16]). Alignment of quads with curvature directions can be controlled explicitly by first computing a smooth cross field with some curvature constraints [4,5]. But such methods can induce undesired anisotropic elements and provide a weak control on size.

Some other methods that use local operators [13] or advancing front methods are classified as local. With advancing fronts methods such as the QMorph algorithm [11], control of the size of quadrilateral is reached but the alignment of the quadrilaterals is solely based on the previous front.

1.2. Positioning and Contributions

Our approach bridges the gap between local and global methods so as to generate quad-dominant meshes. Similar to Pellenard *et al.* [12], we separate the mesh process into two steps: cross field generation and mesh generation. Our two main contributions are as follows:

1. We compute a globally and locally optimal cross field on a triangle mesh of a CAD domain. The added value compared to globally optimal cross field approach [7] is that our approach also reaches local optimality in the sense that cross field is aligned with given constraints either by boundary, curvatures or non-manifoldness.
2. We modify the QMorph algorithm such that generation of new fronts is now cross field-driven instead of growing new fronts orthogonally grown from the previous fronts. This local method advancing front is strongly driven by a globally optimal cross field.

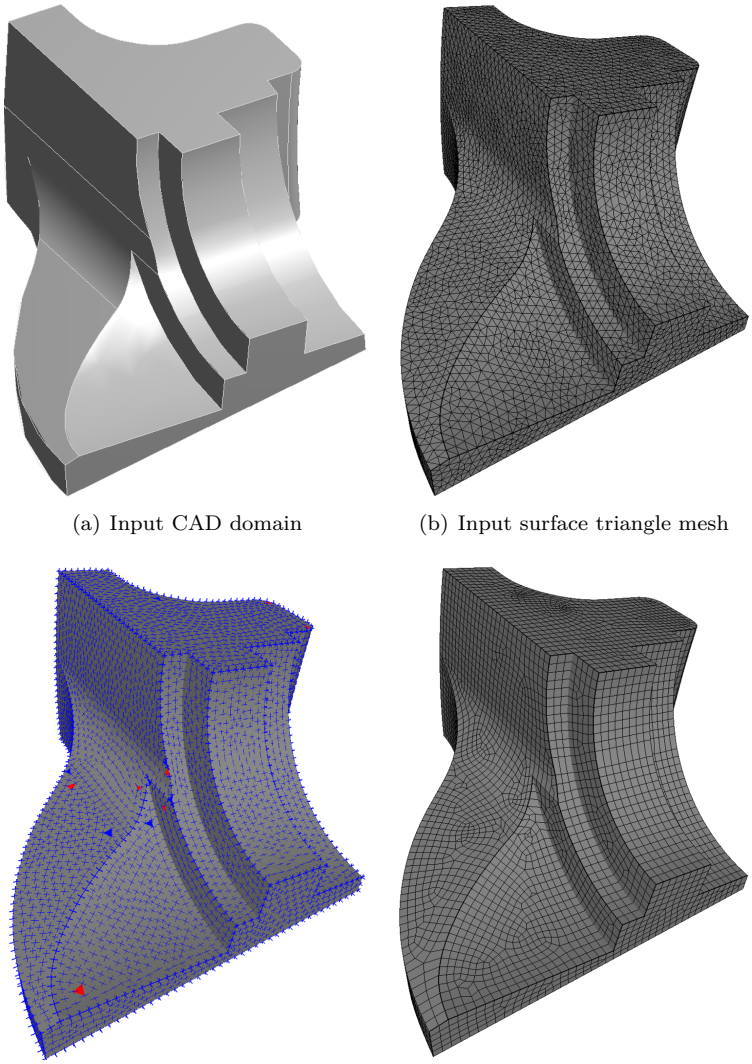
2. Overview

The input of our algorithm is a CAD domain $\mathcal{D}$ given by a boundary representation (part, body, face, loop, edge, vertex), together with a size field. We first precompute a surface triangle mesh $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ of $\mathcal{D}$. The algorithm is then decomposed into two steps. We first compute a cross field defined on vertices of $\mathcal{S}$. We then generate a quad-dominant mesh using a modified QMorph algorithm [11], such that vertex placement and computation of new fronts is cross field-driven. In our experiments, the number of triangles is nearly $1 - 2\%$. Triangles are created in our algorithm for the same reasons that in the original QMorph algorithm. Figure 1 depicts the steps of our algorithm.

3. Cross field

A cross field is computed on the background triangle mesh $\mathcal{S}$. It is derived from the initial requirements to generate square-shaped elements and to generate regular meshes. The principle of cross field algorithms is to minimize the following energy:

$$E(\phi, p) = \sum_{e_ij \in \mathcal{E}} \left(\phi_j - \phi_i + \theta_{ij} + \frac{2p_{ij}\pi}{n} \right)^2, \tag{1}$$



(a) Input CAD domain (b) Input surface triangle mesh

(c) Cross field (d) QMorph

Fig. 1. Overview. Input CAD domain $\mathcal{D}$ (a). Input surface triangle mesh $\mathcal{S}$ computed on $\mathcal{D}$ relatively to an input sizing field (b). Cross field computed on vertices of $\mathcal{S}$ (c). There are 20 singularities in the mesh. Singularities of index 1 (respectively -1) are shown in red (respectively in blue). Final quad-dominant mesh computed with modified version of QMorph algorithm ([11]) such that generation of elements is cross field-driven (d).

where e_{ij} denotes an edge, $p_{ij} \in \mathbb{Z}$ denotes period jumps, and where angles ϕ_i specifying directions θ_{ij} at vertices is the angle between neighboring tangent frames. Our cross field algorithm is an enriched version of the algorithm described in [7]. The added value is that we are able to fix constraints on some parts of the mesh S.

In this section, we first revisit the general principle of Knöppel *et al.*'s algorithm. Then, we describe our new cross field algorithm. We finally show results for some models.

3.1. Knöppel et al.'s algorithm

The principle of curvature alignment algorithm is to minimize a Dirichlet energy E_D on $\mathcal{D}$ for cross fields ($\psi = uX$):

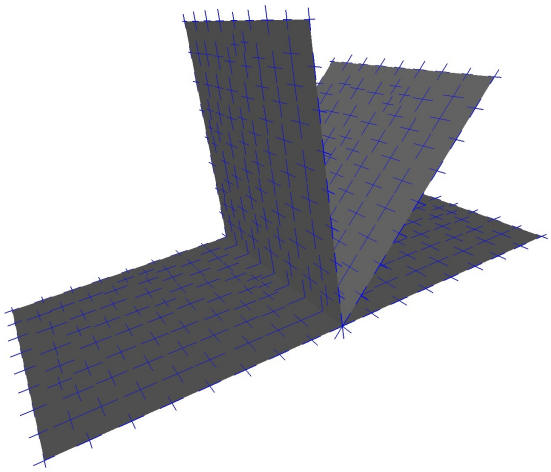


Fig. 2. Cross field for four rectangles sharing one edge. For vertices on the shared edge, four crosses are defined because of decomposition of $\mathcal{S}$ into 4 meshes.

$$E_D(\psi) = \frac{1}{2}\int_{\mathcal{D}} |\nabla\psi|^2 d\mathcal{D}, \tag{2}$$

where ∇ denotes the covariant derivative. For a surface triangle mesh, the problem of minimizing energy E_D can be turned into the matrix problem

$$Au = Mq, \tag{3}$$

where A is a Hermitian matrix representing the Dirichlet energy on $\mathcal{S}$, M is a Hermitian mass matrix on $\mathcal{S}$, q is the coefficient vector of the guidance discrete curvature field, and u is the vector of complex numbers.

3.2. Our approach

We now want to generate a cross field in best accordance with local requirements, such that alignment with boundaries or curvatures. The input of our cross field algorithm is a CAD domain $\mathcal{D}$, together with a background triangle mesh $\mathcal{S} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$. After computing the matrices as described in [7], we choose and compute constraints for the cross field and we finally solve an over-constrained linear system to get the cross field. The steps of the cross field algorithm are summarized as follows:

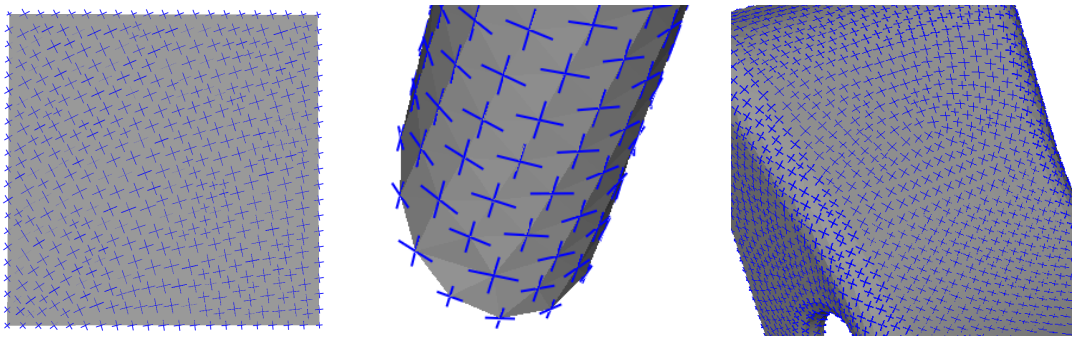
1. Compute matrices A and M, and compute discrete curvatures Mq
2. Compute constraints
3. Linear least squares solver of the over-constrained linear system

The background surface triangle mesh $\mathcal{S}$ can be non-manifold: some edges can be adjacent to more than 2 faces (see Figure 2). In this case, we decompose $\mathcal{S}$ into n triangle meshes $\mathcal{S}_0, \ldots, \mathcal{S}_{N-1}$ such that each $\mathcal{S}_i$ ($i = 0, \ldots N-1$) is manifold. Figure 2 shows the cross field computed on the four triangles meshes. We note that vertices shared by multiple faces have multiple crosses defined.

3.2.1. Constraints

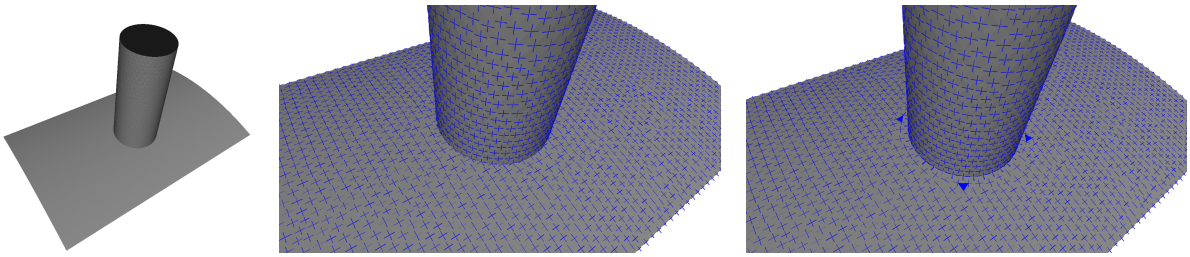
We now describe the four types of constraints we choose to introduce in the cross field.

Boundary constraints. Constraints are applied on the boundary of the mesh. The aim is to align the cross field with the boundary. It handles the two following limitations of Knöppel textitet al.'s algorithm [7]: (i) for flat surface, the smoothest field computed can deviate from expected cross field (see Figure 3(a)),



(a) (b) (c)

Fig. 3. Some limitations of Knöppel *et al.*'s algorithm ([7]). For a flat model, alignment with boundaries is not guaranteed(a). Cross field alignment with boundaries fails (b). The cross field can deviate from principal curvature directions (c).



(a) (b) (c)

Fig. 4. Influence of constraints on non-manifold edges. The model has a non-manifold edge between the flat part and the cylinder part (a). If no constraints are applied along the non-manifold edge, then the cross field does not fit the edge (b). If constraints are applied along the non-manifold edge, then the cross field fits the edge c).

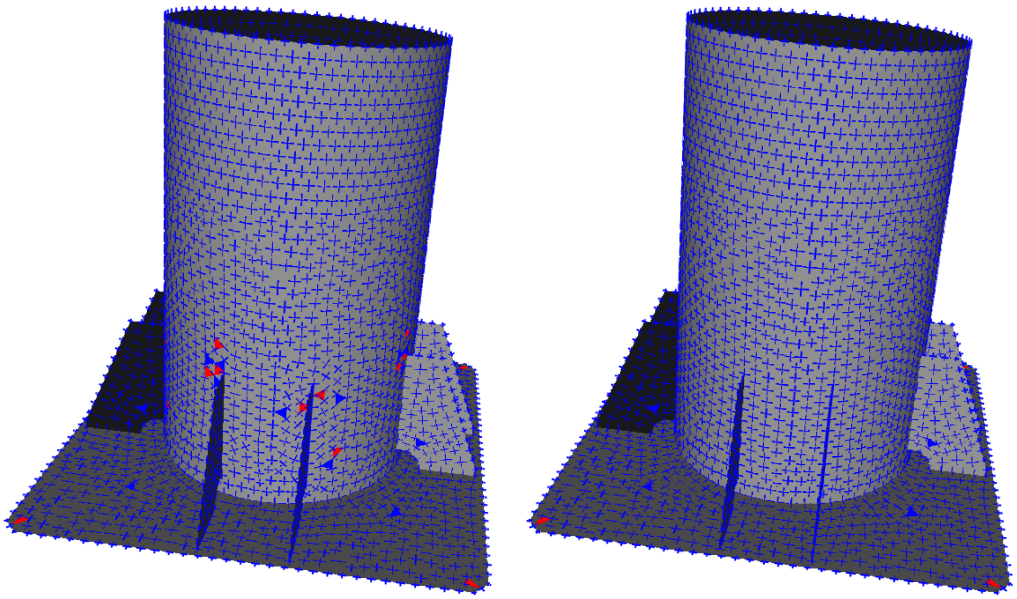
(ii)the aligned field is not necessarily aligned with the boundary (see Figure 3(b)).

Curvature directions constraints. Constraints are applied for vertices where at least one of the mean and Gaussian differ from 0. The aim is to align the cross field with the principal curvature directions. For some models, the aligned field algorithm in [7] fails to align the cross field with curvature directions (see Figure 3(c)). We also observe that introducing constraints only on the boundary and no other location can induce a cross field deviated from curvature directions at other places in order to minimize the global Dirichlet energy.

Non manifoldness constraints. Constraints are applied for vertices connected to non-manifold edges (see in the whole mesh $\mathcal{S}$). We notice that on the Figure 2, edges on the junction of the four faces are manifold if we see them locally inside the face. The edges are non-manidold if we see them in the global domain. Figure 4 shows how the cross field fit the (non-manifold) edge shared by the flat part and the outgoing cylinder.

Sharp edges constraints. Constraints are applied for vertices connected to sharp edges, *i.e.* edges whose dihedral angle is greater than a parameter we call δ and that is set by default to 30 degrees.

So, at this point, for a vertex, we have a direction vector representing the constraint. We also have a unit basis like in Knöppel *et al.*'s algorithm [7]. Then, using the tangent plane of the vertex, we compute the complex number of the direction vector in the basis defined by the unit basis.



(a) (b)

Fig. 5. Importance of the choice of α coefficient in Eq. 4. On the left, keeping a non-zero coefficient for discrete curvatures for constrained vertices can induce a deviation of the cross field and then increase Dirichlet energy (see Eq. 1), compared to model on the right.

3.2.2. Linear system with constraints

We remember that the principle of Knöppel *et al.*'s approach is to solve the linear system of Eq. 3. We introduce a parameter in the right member of Eq. 3 to tradeoff of curvature alignment for constraints alignment. The linear system to solve is now:

$$Au = \alpha \bar{*} Mq, \tag{4}$$

with $\alpha \in \mathbb{R}^{+^n}$, $\bar{*}$ denotes multiplication, member by member, of two vectors and α_i equals 0 if the the vertex of index i belongs to set $\mathcal{N}_1$ and equals 1 otherwise ($\mathcal{N}_1$ denotes the set of constrained vertices and vertices whose edge path length to a constrained vertex is lower than 1). Figure 5 shows how the cross field can be less smooth if vertices close to constraints still have a discrete curvature component in the linear system.

Because of the constraints we previously introduced, the system is over-constrained. We solve it with a linear, least squares method. Without loss of generality, we can assume the first N_F columns of the matrix correspond to unconstrained (or free) vertices of that the other N_C columns correspond to constrained vertices, such that:

$$A = \left(A_F \quad | \quad A_C\right), u = \begin{pmatrix} u_F \\ u_C \end{pmatrix}.$$

A is a square matrix of size $(N_F + N_C, N_F + N_C)$, A_F is a rectangular matrix of size $(N_F + N_C, N_F)$ and A_C is a rectangular matrix of size $(N_F + N_C, N_C)$.

$$Au = \alpha \bar{*} Mq, i.e. \left(A_F u_F = \alpha \bar{*} Mq - A_C u_C\right).$$

Let $b = \alpha \bar{*} Mq - A_C u_C$. Then, $A_F^t A_F u_F = A_F^t b$, where A_F^t is the complex transpose matrix (or hermitian) of A_F. Then, we solve

$$Bu_F = b',$$

with $B = A_F^t A_F$ (square matrix), and $b' = A_F^t b'$.

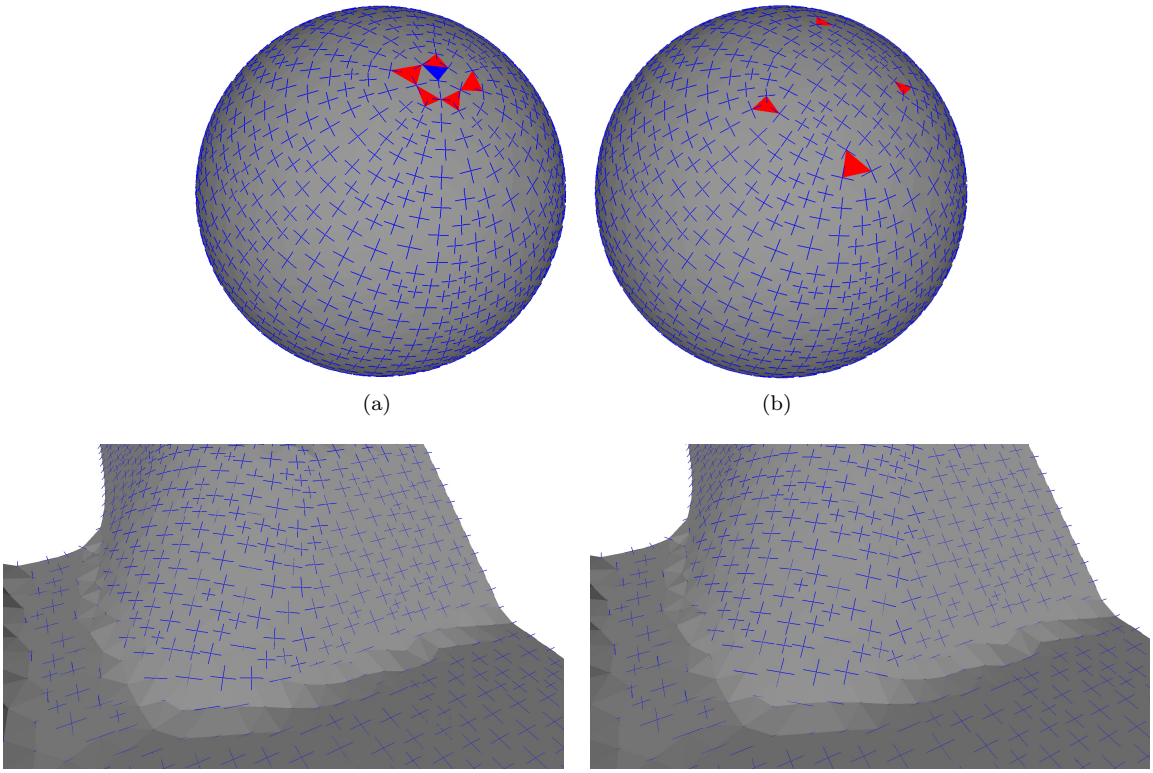


Fig. 6. Inconsistencies in curvature directions. For each, we depict in red (resp. blue) singularities of positive (resp. negative) index. Inconsistencies in curvature direction estimations (a, c). Inconsistencies removed (b,d).

3.2.3. Remove inconsistent constraints

Introducing constraints can lead to inconsistencies, typically if the curvature directions differ significantly for adjacent vertices. We choose to remove the constraints for the three vertices of a triangles whose index[1] differs from 0. We also remove the constraints for the two vertices of an edge whose adjacent crosses deviate too much. Figure 6 shows how it improves the smoothness of the cross field. On the sphere, singularities are well distributed around one pole. On the other model, the cross field is recomputed locally in a small neighborhood around inconsistencies, making it smoother than before. Figure 7 shows the results of our cross field algorithm on some examples. The output cross field is aligned with curvatures, boundaries and non-manifold edges. The cross field is also globally smooth relative to Dirichlet energy.

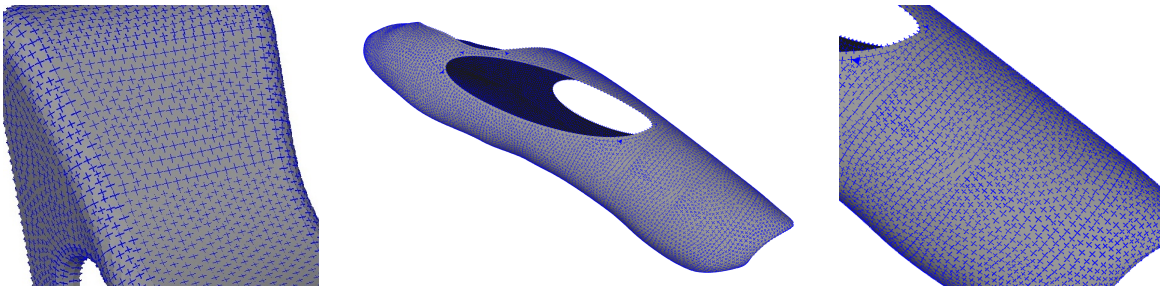
4. Modified QMorph

The QMorph algorithm must now be modified to use the cross field when creating new quadrilaterals and new fronts.

4.1. Principle of the original QMorph algorithm [11]

The QMorph quadrilateral mesh generation algorithm contains the following main steps:

[1] We refer the reader to [7] for the definition of the index



(a) (b) (c)

Fig. 7. Cross field results. Cross field is aligned with both curvatures and boundaries (a). Cross field is aligned with curvatures, boundaries, and with non-manifold edges (b). Zoom of middle figure (c).

1. Creation of a background surface triangle mesh $\mathcal{S}$ which covers the entire domain $\mathcal{D}$, as we described in Section 2. It will continually change and shrink while quadrilaterals are formed as fronts advance to interior of the domain. An edge of the quadrilateral mesh can directly come from the initial mesh $\mathcal{S}$ or from edges created through operators on $\mathcal{S}$. In this way, robustness is improved because there is no edge to edge intersection checking, contrary to Paving algorithm [2].
2. Front definition: The initial front is the set of edges of $\mathcal{S}$ that are adjacent to only one triangle.
3. Front edge classification: Each front is initially classified according to angles into one of three states: $0-0$, $0-1$, $1-0$ (following the terminology of Owen *et al.* [11]). The state of a front edge defines how the edge will eventually be used in forming a quadrilateral. Angles between adjacent front edges determine the state of an individual front.
4. Front edge processing: Each front edge is individually processed to create a new quadrilateral mesh from triangles in $\mathcal{S}$. The current front always defines the interface between the quadrilaterals and the triangles. Figure 8(a) shows the process on the front $N_A - N_B$. Front edges are handled differently according to their current state classification. Once a new quadrilateral is created, the front is redefined and the adjacent front edge states are updated, through the following sub-steps:

   (a) Side edge definition: Selecting a front edge from the edge front lists, side edges are defined. Side edges may be defined by using an existing edge, by swapping the diagonal of adjacent triangles, or by splitting triangles to create a new edge. In Figure 8(b), side edge $N_B - N_C$ shows the use of an existing edge, while the side edge $N_A - N_D$ was formed from a local swap operation.
   (b) Top edge recovery: The final edge on the quadrilateral is created by an edge recovery process. It modified locally the triangulation using local edge swaps to enforce an edge between the two nodes at the ends of the two side edges. Edge $N_C - N_D$ in Figure 8(c) was formed from a single swap operation. Any number of swaps may be required to form the top edge.
   (c) Quadrilateral formation: The last quadrilateral is formed by merging any triangles bounded by the selected front edge, and the newly created side edges and top edge as shown in Figure 8(d).
   (d) Local smoothing: The mesh is smoothed locally to improve both quadrilateral and triangle element quality as shown in Figure 8(e).
   (e) Local front reclassification: The front is advanced by removing edges from the edge front lists and adding edges to the front lists that have one triangle and one quadrilateral adjacency. New front edges are classified by state. Existing fronts that may have been adjusted in the smoothing process are reclassified.

5. Front edge processing (step 4) continues until all edges on the edge front lists have been deleted.

Quite often, a quadrilateral mesh generated with the algorithm is not suitable for many applications. For example, for crash test analysis, an analyst prefers an uniform mesh with quadrilateral edges aligning with

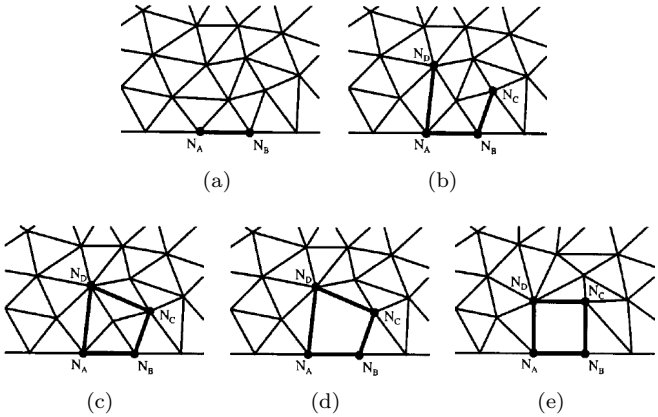


Fig. 8. Steps demonstrating process of generating a quadrilateral from front $N_A - N_B$ in the original QMorph algorithm [11]. Initial front (a). Side edge definition (b). Top edge recovery (c). Quadrilateral formation (d). Local smooth (e). This figure is taken from Owen *et al.* [11].

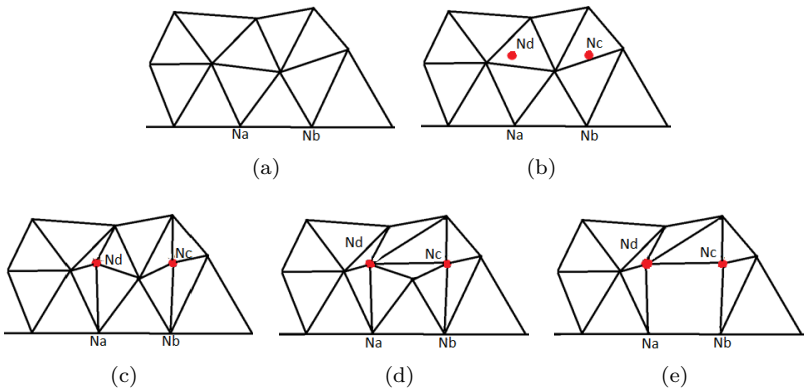


Fig. 9. Steps demonstrating process of generating a quadrilateral from font $N_A - N_B$ in our approach. Initial front (a). Side edge definition (b). Top edge recovery (c,d). Quadrilateral formation (e).

feature lines and local curvatures. Given a planar face with curve boundary edges and/or interior holes, the algorithm is unable to deliver a uniform mesh, shown in Figure 10(a). This is due to several shortcomings in the algorithm:

1. Mesh size: At the beginning, the initial triangulation stipulates to the size field along the surface. By steps 4a, 4b and 4d, the triangulation is constantly changed and modified. The triangles in the interior region are distorted and stretched. The size field implicated by the initial triangulation is destroyed after generating several layers of quadrilaterals.
2. Quadrilateral edge alignment: In step 4a, a quadrilateral is formed with the edges from the triangle mesh. It is then smoothed locally in step 4d. These steps cannot guarantee alignment of edges with feature lines or local curvatures and conformance to the local size field.

4.2. Modified QMorph

We propose a novel idea to modify the algorithm described in Section 4.1. First, we maintain no edge-to-edge intersection checking. Second, we generate edges based on the local size and cross fields. As a result, a newly created quadrilateral will respect the local size field and the local cross field. Consequently, edges

will be aligned with feature lines and with principal curvatures directions. Furthermore, we do not need to perform local smoothing. It can be expected that the final quadrilateral mesh also respects the size and cross fields in most regions, except around front collisions.

In this section, the QMorph cross field-driven quadrilateral mesh generation algorithm with cross field is discussed. The proposed approach is able to generate a quadrilateral mesh conforming to the size field and to the cross field without sacrificing the robustness of the original QMorph algorithm. The proposed modifications to the original QMorph are highlighted:

1. Background surface triangle mesh. Unlike for the original QMorph algorithm, there is no need for the mesh size to respect the size field, any triangle mesh can be used. The use of the background mesh has also changed, such that:

   (a) it is used to locate which triangle contains the new end point to define a side edge. The new point and its corresponding side edge creation will be discussed below;
   (b) the edges from the triangulation, *i.e.* side edges, are still used during the generation of quadrilaterals. In this way, there is no edge to edge intersection checking.

2. Side edge definition: When selecting a front edge from the edge front lists, side edges are defined. The location of the new point is computed based on the local size field and cross field. The final location is defined in such a way that its side edge will respect both size field and cross field. The size field and cross field can be stored with a spatial data structure; for example the ADTree (alternating decision tree) [6]. Figure 9 illustrates how to define a side edge $N_A - N_D$ from a front edge $N_A - N_B$, to minimize the deviation of edges generated from the size field and from the cross field. Point N_B is a weighted combination of size and cross field.
3. A similar approach is applied to define side edge $N_B - N_C$, based on the size field and the cross field at N_D, N_B and N_C. Once the top edge N_C-N_D is created by edge swapping, a quadrilateral is formed by its nodes and edges, which conform to the local size and cross fields. There is no need to apply any quality improvement operations, like step 4.d (local smoothing) in the original QMorph algorithm, except when near to front collision regions. The global mesh relaxation is still useful in optimizing the end quality of the mesh but it is not nearly as time consuming as the initial mesh is of much higher quality. We are really interested in relaxation near front collisions.

Figure 10 shows how our new approach provide better meshes for both uniform and varying size field. We refer the reader to Table 1 to compare mesh quality.

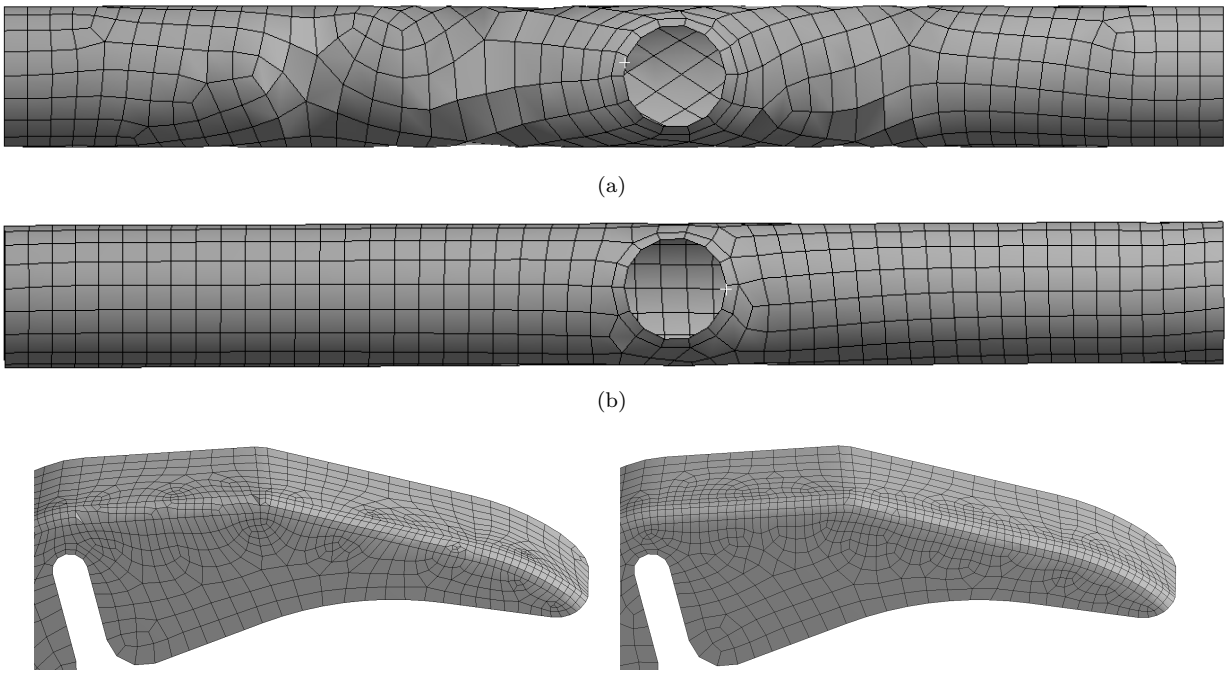
A keypoint is that the point each time a new point N_D is computed with size and/or direction fields, it is then projected on the surface $\mathcal{S}$.

4.2.1. Quadrilateral patterns around cross field singularities.

Special patterns are created around singularities of the cross field: 3 divisions and 5 divisions for positive and negative index singularities respectively. For cases where a singularity is close to the boundary, the pattern edges that might be too closed to existing boundary edges are removed to avoid difficulties in later processing. The pattern orientation is computed by minimizing an energy similar to Eq. 1, such that the edge deviation to the cross field is minimal.

5. Results

Figure 12 illustrates the behavior of our algorithm on real models like the compressor model, the marathon model, and the tower bridge model. Our method behaves well even when aligning the cross field with curvature principal directions as well as on nearly planar areas where boundary constraints on the cross field



(a)

(b)

(c) (d)

Fig. 10. Comparison of final mesh without (a,c) and with cross field (b,d). QMCF generates mesh with a better alignment with principal curvature directions.

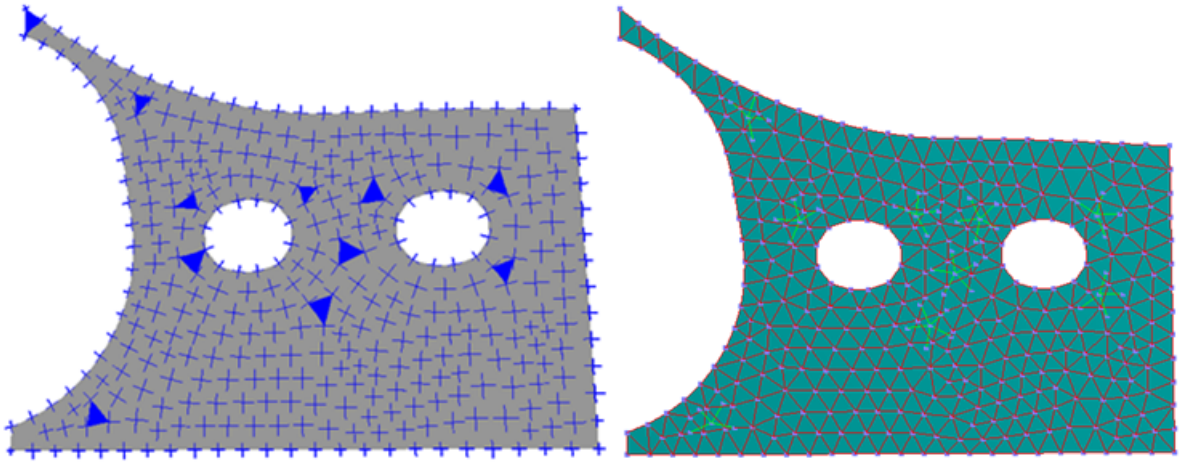


Fig. 11. Creation of patterns at the singularities of the cross field. 11 singularities of index -1 in the cross field (left). First edges of the quad mesh in red (center). Final mesh (right).

lead to a well aligned mesh with boundaries. Figure 1 is a sanity check on a typical model in the mesh community: the Fandisk. The mesh provided by our algorithm is well aligned with both sharp features and principal curvature directions.

Quality. During our experiments, we observed that the quad-dominance of the output mesh is nearly 98-99%. We observed that the quadrilateral meshes generated by our new algorithm are generally better

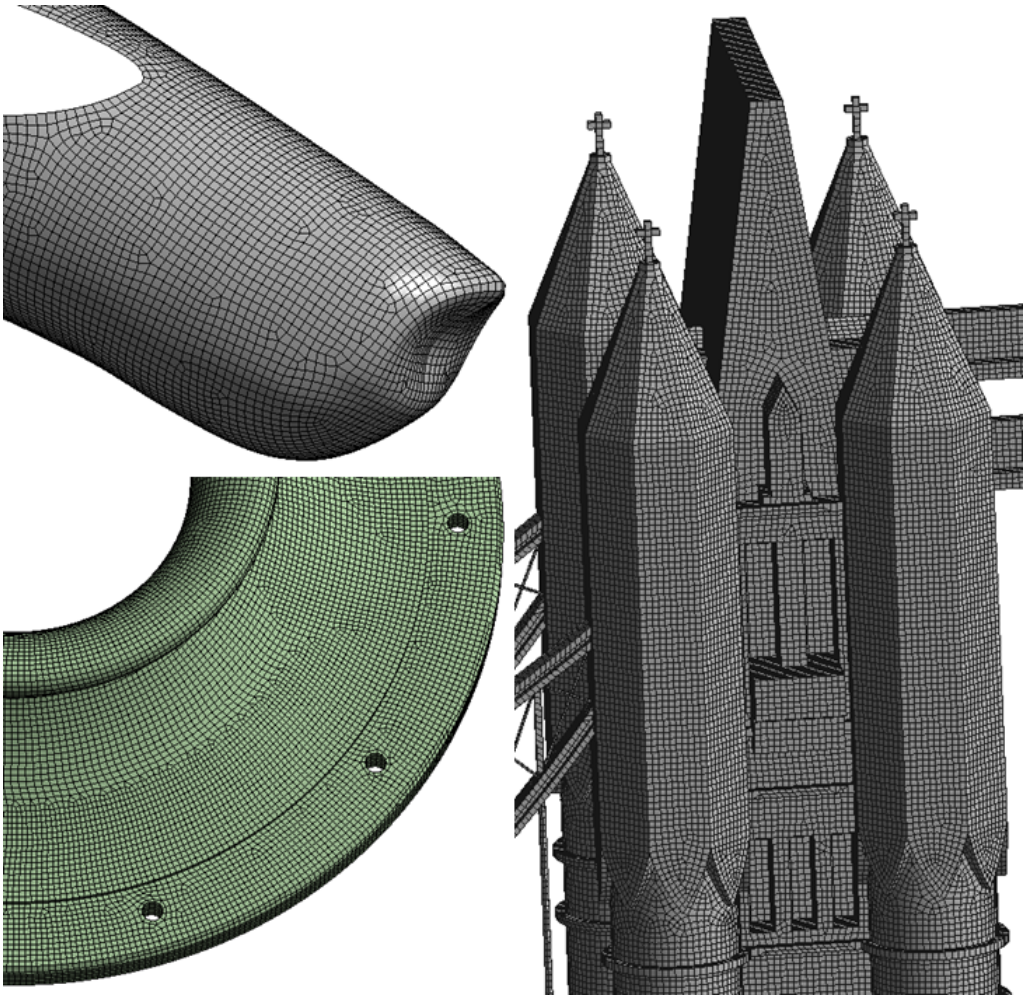


Fig. 12. Several meshes generated with our approach. Final mesh is well-aligned with both curvatures and boundaries.

Model	Jacobian ratio	Aspect ratio	Orthogonal quality
Fandisk (Figure 1)	(1.162/**1.11**)	(1.17/**1.09**)	(0.983/**0.985**)
Cylinder (Figure 10(a,b))	(1.19/**1.07**)	(1.25/**1.09**)	(0.97/**0.99**)
Figure 10(c,d)	(1.39/**1.3**)	(1.38/**1.24**)	(0.943/**0.965**)

Table 1. Quality improvment from original QMorph to QMCF. All quality scores are average, and target value is 1. Left and right values are quality scores for original QMorph and QMCF respectively.

than those generated by the original QMorph algorithm for conformance with the size field, cross field, and for orthogonality, Jacobian and aspect ratio. Table 1 shows how our algorithm improves quality of the output mesh.

Implementation details. Our algorithm is implemented in C++. The complexity of our algorithm does not depend on the number of singularities. The overall time is slightly increased compared to regular QMorph due to the computation and interrogation of size field and cross field.

6. Conclusion

We introduced a novel algorithm for generation of quad-dominant meshes. A cross field is first generated on a triangle mesh of the domain such as it is aligned with both principal curvatures directions, boundaries, sharp edges and non-manifold edges. Then, a novel approach based on original QMorph algorithm is described. The generation of new fronts are now cross field-driven instead of orthogonally grown from the previous fronts. The added value of our approach is its capability to generate meshes aligned with principal curvature directions (see Figure 12(c) and Figure 1(d)).

Acknowledgments

We are grateful to Sarang Dalne and Krishna Samavedam for experiments. We also thank Anatole Wilson and Joseph Borella for feedback.

References

[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Trans. Graph.*, 22(3):485–493, 2003.

[2] Ted D Blacker and Michael B Stephenson. Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32(4):811–847, 1991.

[3] I. M. Boier-Martin, H. E. Rushmeier, and J. Jin. Parameterization of triangle meshes over quadrilateral domains. In *Symposium on Geometry Processing*, pages 197–208, 2004.

[4] D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3), 2009.

[5] David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. Integer-grid maps for reliable quad meshing. *ACM Trans. Graph.*, 32(4), July 2013.

[6] Y. Freund and L. Mason. The alternating decision tree algorithm. *Proceedings of the 16th International Conference on Machine Learning*, pages 124–133, 1999.

[7] Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. Globally optimal direction fields. *ACM Trans. Graph.*, 32(4), 2013.

[8] Y.-K. Lai, L. Kobbelt, and S.-M. Hu. An incremental approach to feature aligned quad dominant remeshing. In *Proceedings of the ACM symposium on Solid and physical modeling*, SPM '08, pages 137–145, 2008.

[9] B. Lévy and Y. Liu. L^p centroidal voronoi tessellation and its applications. *ACM Transactions on Graphics*, 29(4), 2010.

[10] M. Marinov and L. Kobbelt. Direct anisotropic quad-dominant remeshing. In *Pacific Conference on Computer Graphics and Applications*, pages 207–216, 2004.

[11] S. J. Owen, M. L. Staten, S. A. Canann, and S. Saigal. Q-morph: an indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering*, 44(9):1317–1340, 1999.

[12] B. Pellenard, J.-M. Morvan, and P. Alliez. Anisotropic rectangular metric for polygonal surface remeshing. In *Proceedings of the International Meshing Roundtable*, pages 367–384, 2012.

[13] N. Pietroni, M. Tarini, and P. Cignoni. Almost isometric mesh parameterization through abstract domains. *IEEE Trans. Vis. Comput. Graph.*, 16(4):621–635, 2010.

[14] N. Ray, W. C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Trans. Graph.*, 25(4):1460–1485, 2006.

[15] K. Shimada and J.-H. Liao andT.i Itoh. Quadrilateral meshing with directionality control through the packing of square cells. In *IMR*, pages 61–75, 1998.

[16] Y. Tong, P. Alliez, D. Cohen-Steiner, and M.Desbrun. Designing quadrangulations with discrete harmonic forms. In *Symposium on Geometry Processing*, pages 201–210, 2006.