Directionality fields in the generation and evaluation of quadrilateral meshes

Alice Herrera de Figueiredo, Waldemar Celes
Instituto Tecgraf/PUC-Rio, Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brasil
{herrera,celes}@tecgraf.puc-rio.br

***Abstract*—One of the main challenges in quadrilateral mesh generation is to ensure the alignment of the elements with respect to domain constraints. Unaligned meshes yield numerical problems in simulations that use these meshes as a domain discretization. There is no alignment metric for evaluating the quality of quadrilateral meshes. A directionality field represents the diffusion of the constraints orientation to the interior of the domain. Kowalski et al. use a directionality field for domain partitioning into quadrilateral regions. In this work, we reproduce their partitioning method and adapt it to reduce the final number of partitions. We also propose a new metric to evaluate the quality of a quadrilateral mesh with respect to the alignment with domain constraints.** [1]

I. Introduction

The discretization of domains in meshes is one of the main tasks in numerical simulations of engineering using the finite element method to solve systems of partial differential equations. Quadrilateral and hexahedral elements are sometimes chosen because they have better interpolation functions. When compared to triangles, quadrilateral elements are usually more stable and need less refinement. Moreover, the transformation of quadrilaterals into triangles is easier then vice-versa.

In this work, we focus on quadrilateral mesh generation and in the method of Kowalski et al. [1], which shows how to partition the domain in quadrilateral regions using the concept of directionality. Given a triangle mesh, their method generates a directionality field by solving a partial differential equation. Then it uses the directionality field to find the singularities of the field and to generate their separatrices, streamlines that will partition the domain into quadrilateral regions. As those regions respect the directionality field, any direct mapping method generates a good quadrilateral mesh, as we can see in Fig. 1.

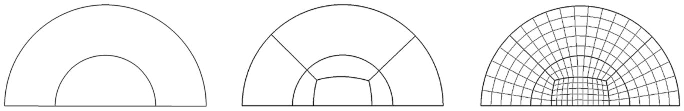


Fig. 1. Example of the domain decomposition in quadrilateral regions and the quadrilateral mesh generation [1].

The main contributions of this work are:

- A implementation of the method of Kowalski et al. [1], with the needed modifications for domains with constraints (Sections III and IV).
- A new quality metric that globally evaluates alignments of quadrilateral meshes (Section V).

II. Related Works

A. Quadrilateral mesh generation

Quadrilateral mesh generation is a field that has been widely studied. However, there are still many challenges, specially the generation of meshes aligned with boundary and interior constraints for complex domains. Aligned meshes are very important for modeling [2], to respect the shapes, and for numerics, to best capture the physical phenomenon [3].

Blacker and Stephenson [4] introduced the paving algorithm, which paves rows of elements from the boundary inward. The method respects the geometry alignment and constraints orientations. There are lot of works that use this paving method. White and Kinney [5] propose a mesh generation of element by element, instead of row by row advancing front. Recently, Park et al. [6] present a paving method in domains with open boundary constraints.

For complex constraints domains, Araújo and Celes [7] present a new automatic method for quadrilateral mesh generation based on deferred constraint insertion. Their method starts with a triangular mesh that is locally modified to satisfy each inserted constraint. At the end, they use some heuristics to convert the triangular mesh into a quadrilateral one. Pochet et al. [8] propose a direct method for quadrilateral mesh generation, which receives a geometric constraint set as input and uses a quadtree to adapt and subdivide the domain.

Direct mapping methods ensure alignment, but need to partition the domain. Many works use the medial axis or skeleton to subdivide the domain [9]. However, the medial axis is not stable: a small change in the domain can affect the skeleton. Moreover, the resulting regions normally do not have four sides, and so are not suitable to generate quadrilateral structured meshes.

Ray et al. [10] formalize the concept of n-symmetry direction field, which are objects invariant by rotation of $\frac{2\pi}{N}$. Works such as Bommes et al. [11], Tarini et al. [12] e Kowalski et

[1] This work relates to an M.Sc. dissertation.

al. [1], [13], [14] use this concept of directionality field to partition the domain in 2D or 3D into suitable regions for the quadrilateral or hexaedral mesh generation. In this work, we explore domain partition with directionality fields, based on Kowalski et al.'s papers: 2D [1] and 3D [14].

B. *Quality Metrics for Quadrilateral Meshes*

There are many different ways to evaluate the quality of a quadrilateral mesh. It is possible to evaluate the vertex valence. A quadrilateral mesh is *regular* when the valence of every vertex is 4. Another quality metric is the minimum angle between quadrilateral edges. The aim is to measure the number of elements that have the minimum angle closer to 90° and thus closer to squares. The Jacobian quality metric analyzes the deviation of a mesh element from the ideal element: it measures how much the element has deformed from parametric space to object space.

These metrics are local: they evaluate the element locally, but do not evaluate the mesh globally. None of these metrics evaluates mesh alignment. This work proposes a new metric to evaluate mesh alignment, as explained in Section V.

III. Reference Work

The first part of this work was to revisit Kowalski et al.'s work [1]. In this section we explain with detail their original method.

Kowalski et al. [1] present an algorithm to partition a 2D domain Ω into regions suitable for quadrilateral mesh generation. Given a uniform triangulation of Ω, the algorithm has the following steps, shown on Fig. 2:

a,b: Generate a directionality field on Ω, propagating the representation vectors by solving a PDE.
c: Find singularities and separatrices used to partition Ω in quadrilateral regions.
d: Generate a quadrilateral mesh using structured mapping in each quadrilateral region.

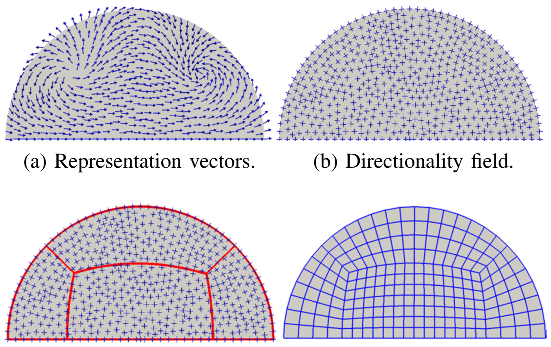

(a) Representation vectors.
(b) Directionality field.
(c) Quadrilateral regions partition.
(d) Quadrilateral mesh.

Fig. 2. Kowalski et al method [1].

A. *Directionality field*

Directionality is the concept that guides Kowalski et al. [1]. The creation of a directionality field aims to propagate the orientations of the constraints to the interior of Ω. The directionality is continuous on the straight corners, which makes them continuous in quadrilateral regions.

A regular vertex P in the interior of a quadrilateral mesh is four-valent and the four incident edges can be seen as two sets of opposite vectors. The tangents of these curves in P can be described as a *cross*.

A vector $\vec{v}_0$ is formed by using θ, which is an angle between one of the cross's tangent and a fixed direction (Kowalski et al. [1] use the x axis): $\vec{v}_0 = (\cos\theta, \sin\theta)$. Using $\vec{v}_0$ it is possible to build the directionality. A *directionality* of $\theta \in [0, \frac{\pi}{2})$ is defined as:

$$C_\theta = \left\{ \vec{v}_k = \left(\cos\left(\theta + \frac{k\pi}{2}\right), \sin\left(\theta + \frac{k\pi}{2}\right) \right)^T, 0 \leq k \leq 3 \right\}$$

Given θ of $\vec{v}_0$, the angles of the following vectors $\vec{v}_1, \vec{v}_2, \vec{v}_3$ are:

$$\theta_{k+1} = (\theta_k + \frac{\pi}{2}) \bmod 2\pi \quad \text{where} \quad \theta_0 = \theta, \quad 0 \leq k \leq 2$$

As the directionality has four vectors, Kowalski et al. [1] created the *representation vector* to make the interpolation easier. Thus, the representation vector $\vec{v}_r$ of the directionality C_θ is given by:

$$\vec{v}_r = (\cos(\theta_r), \sin(\theta_r)) \quad \text{where} \quad \theta_r = (4\theta_0) \bmod 2\pi$$

As a consequence, when the representation vector rotates by 2π, the corresponding directionality will only rotate $\frac{\pi}{2}$. When two directionalities differ by a π rotation, they will have the same representation vector, as shown in Fig 3.

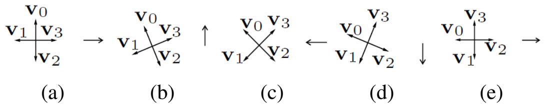


Fig. 3. Directionalities and their representation vectors [1].

Kowalski et al. [1] solve a diffusion problem to propagate information from the constraints to the entire domain:

$$\begin{cases} J(u) = \int_\Omega |\nabla u|^2 dx \\ u(x) = u_0(x) \quad \forall x \in \partial\Omega \\ |u(x)| = 1 \quad \forall x \in \Omega \end{cases}$$

B. *Singularities*

Next, Kowalski et al. [1] detect the singularities of the directionality field. A singularity occurs when the field is not defined and has a zero representation vector. A singularity can be inside the triangle or on an edge.

Given a triangle T with P_1, P_2, P_3 as vertices, $v(P_i)$ as the representation vector of P_i and θ_i as the angle between $v(P_i)$ and the x axis, the Poincaré index [1], [15] of T is given by:

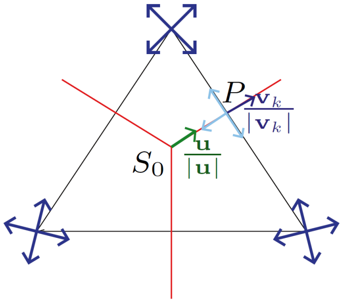
$$i_s = \frac{\Delta\theta_{12} + \Delta\theta_{23} + \Delta\theta_{31}}{2\pi}$$

where $\Delta\theta_{ij} = \theta_j - \theta_i - \pi$. The triangle T contains a singularity when $i_s = \pm 1$. When $i_s = 0$, the triangle T does not contain a singularity.

C. Separatrices

After finding the singularities, the next step is to define the separatrices that will form the quadrilateral regions. Separatrices are streamlines that start at a singularity point. A streamline of a vector field is a curve whose tangent at any point has the same direction as the vector field at this point. On a streamline of a directionality field, the tangent at a point has the same direction as one of its directionality vectors at the point.

The first step to generate a separatrix is to find the intersection point in the edge of a triangle that contains the singularity. Given an edge e of a triangle T that contains a singularity S_o, a point P in e and a vector $\vec{u} = P - S_o$, the separatrix will intercept e at P when one of the vectors $\vec{v}_k$ of P's directionality has the same direction and orientation of the vector $\vec{u}$, as in Fig. 4.



P
$\frac{\mathbf{v}_k}{|\mathbf{v}_k|}$
S_0
$\frac{\mathbf{u}}{|\mathbf{u}|}$

Fig. 4. Separatrices originated from the singularity S_0 intersection with the triangle that contains S_0. [1].

The next step is to generate further segments on the separatrix. This is done by a numerical integration method. Kowalski et al. [1] use Heun's method, a second order Runge–Kutta variant [16], shown in Fig. 5.

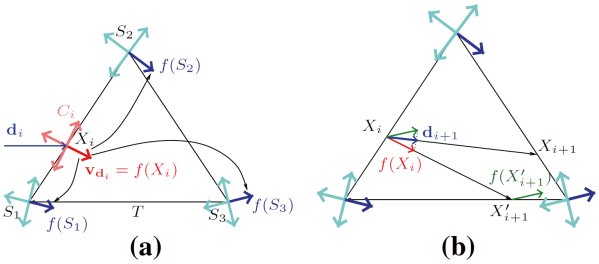


Fig. 5. Triangle integration process to define the separatrices [1].

The integration method integrates over the triangles. Each integration step starts at a point of a triangle and ends at another point of the same triangle. Each step begins at a point X_i that is on an edge of a triangle T. This point has an input direction $\vec{d_i}$ and the aim is to find the output direction $\vec{d}_{i+1}$ of T at X_{i+1} point.

The representation vector in X_i is linearly interpolated from the representation vectors of the vertices S_i of T. They then define a vector field $Y = f(X)$ over T, which is the vector of the directionality that has the small angle compared to the direction $\vec{d_i}$ denominated $v_{\vec{d_i}}$.

Knowing that $v_{\vec{d_i}} = f(X_i)$, to find the output point X_{i+1}, we need to find a intermediate point X'_{i+1}:

$$X'_{i+1} = X_i + hf(X_i)$$

From $f(X'_{i+1})$ and $f(X_i)$, it is possible to find the output direction $\vec{d}_{i+1}$:

$$\vec{d}_{i+1} = \frac{f(X_i) + f(X'_{i+1})}{2}$$

Then:

$$X_{i+1} = X_i + h\vec{d}_{i+1}$$

The integration ends when the streamline reaches another singularity or the border of the domain Ω.

D. Quadrilateral Mesh Generation

To generate the quadrilateral regions, geometric singularities are added. They are the corners of $\partial\Omega$ and they are the seeds for separatrices using the same method. Thereby, the set of separatrices partitions the domain into quadrilateral regions. This is possible because the resulting regions do not contain singularities.

IV. Implementation and Results

In this section we explain how we have implemented Kowalski et al.'s method [1], discussing the chosen adaptations. To represent the model's mesh, we used TopS [17], a library that provides a compact topological data structure for finite element meshes.

A. Directionality field

We used the same concept of directionality and representation vector. So the first step is to propagate the constraints information through all the domain. In our work, we considered constraints inside and on the border of the domain, shown in red in Fig. 6. We find the representation vector for each vertex from the vertex's directionality that is formed by the edge's normal and the edge's tangent, as in Fig. 7.

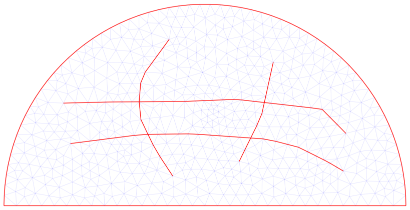


Fig. 6. Model with its constraints.

We used the program gHEM [18] to propagate the directionality from the constraints to the interior. The program simulates diffusion by solving a heat conduction problem.

We propagate the representation vectors of the constraints for each component: component x and component y. The input to gHEM is a triangular mesh and scalar values (the components of the representation vector) at the constraint vertices: those values will be the temperature. We used the program Pos3D [19] to view the result of the scalar field for each component, as shown in Figs. 8 and 9.

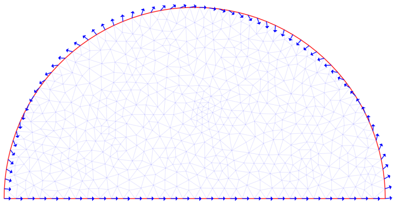


Fig. 7. Representation vectors prescribed from the constraints in red

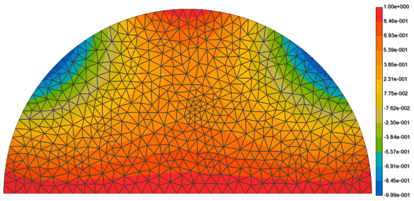


Fig. 8. x axis field.

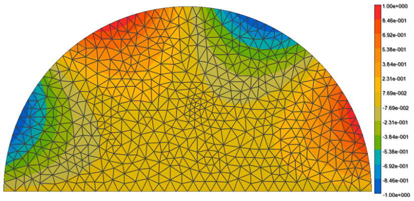


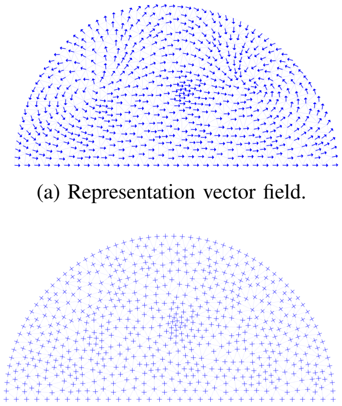
Fig. 9. y axis field.

This procedure differs from the original [1] which imposes the restriction that the vectors need to be unitary; otherwise Kowalski et al. [1] say there will be errors and variations in the directionality orientations. We did not encounter those limitations when using gHEM. At the end of the diffusion, we have a value for each component (x and y) that we use to build the representation vector at each vertex of the mesh. Fig. 10 illustrates the result of this propagation.

B. Singularities

To define the value of the field in a point P inside a triangle ABC, we used barycentric interpolation. We defined the representation vectors of the triangle vertices as "points". Knowing that the singularities occurs when the directionality field is undefined and has a zero representation vector, we simply need to find the representation vector $\vec{v}_p = (0, 0)$ for P to be a singularity point (Fig. 11).

Compared to Kowalski et al.'s work [1], this method has the advantage that it gives the exact position of the point P that has a zero representation vector. Fig. 12 shows the representation vector magnitude, the red color is magnitude 1 and the blue color is the magnitude 0. The dark blue region is exactly the region that has a singularity, which was already expected.



(a) Representation vector field.

(b) Directionality field.

Fig. 10. Result of the propagation.

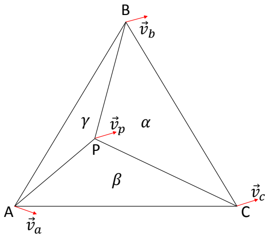


Fig. 11. We used barycentric interpolation to find the singularity point in a triangle.

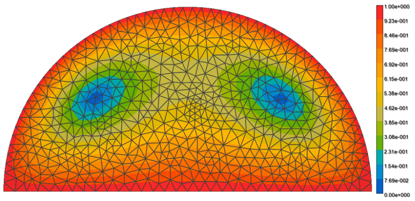


Fig. 12. Representation vector magnitude.

C. Separatrices

To generate the separatrices, we used the same method as Kowalski et al. [1]. However, instead of using only the triangle that contains the singularity, we used the two triangles that share the edge closest to the singularity point as in Fig. 13. This was the solution to numerical imprecision errors when the singularity was very close to an edge.

D. Domains with constraints partition

One of our focus was to investigate the subdivision of complex domains, such as geologic domains, which also have constraints in its interior. We aim to decompose the domain into quadrilateral regions so that the prescribed directionality

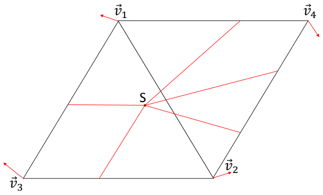


Fig. 13. We've used two triangles to calculate the separatrices' start.

does not introduce discontinuity in the field. Fig. 14 shows a domain that has interior constraints (in red) and the separatrices (in blue). The constraints, together with the separatrices, form quadrilateral regions.

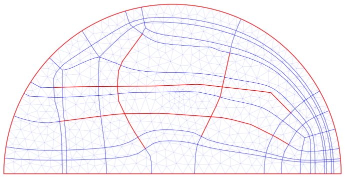


Fig. 14. Domain with interior constraints and the resulting subdivision.

The result shows that the separatrices originated from geometric singularities have followed the directionality field. Consequently, it was possible to subdivide the domain in quadrilateral regions suitable for a direct mapping, as shown in Fig. 15. This mesh was generated with Sigma2D [19].

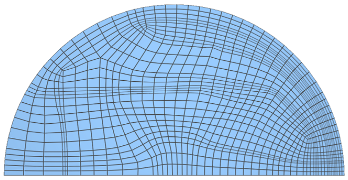


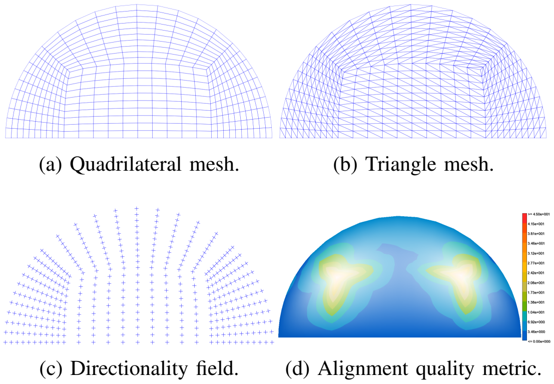
Fig. 15. Quadrilateral mesh generated from the regions of Fig. 14.

V. A Quality Metric For Quadrilateral Mesh Alignment

An application of the directionality fields is to measure the alignment of quadrilateral meshes, hence their quality. As mentioned before, no metric exists for analyzing the quality of quadrilateral mesh alignment. In the second part of this work, we proposed a new metric to evaluate the alignment in quadrilateral meshes, divided in the following steps, show in Fig. 16:

a: Input: Quadrilateral mesh Q of Ω together with its constraints
b: Triangle mesh T created from Q
c: Directionality field of Ω
d: Output: Alignment quality metric

The method starts with a quadrilateral mesh Q as input. The next step is to convert it into a triangle mesh T. For that we subdivide each quadrilateral in two triangles using one of the diagonals. With T, we generate the directionality field of the domain using our method described in Section IV-A. With the directionality at each vertex, we evaluate the quality described as follows.



(a) Quadrilateral mesh.
(b) Triangle mesh.
(c) Directionality field.
(d) Alignment quality metric.

Fig. 16. Quality metric steps.

The metric is composed by two factors: *deviation* d and *reliability* α. The deviation represents how misaligned the vertex is compared to its directionality. For this evaluation we compare each edge incident to the vertex with the vectors $\vec{v}_k$ of the directionality. For each edge e_i, we find which $\vec{v}_k$ has the smallest angle (θ_{min_i}) with e_i. The deviation will be the largest θ_{min_i}, that is $d = \max\{\theta_{min_i}\}$ (Fig. 17).

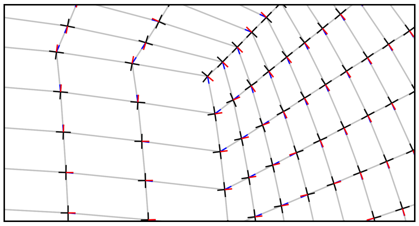


Fig. 17. Deviation representation. Mesh edges in gray and directionality in black. The blue and red vectors are the one that we use to find the angle of the closer vertex to the edge.

The reliability α means how much we can trust the deviation measure in each vertex. The directionality is undefined at the singularities, and so it is impossible to evaluate the alignment there. On the other hand, the directionality is prescribed on the constraints and so is completely trustworthy. Therefore, we can say that the reliability is 0 at the singularities and 1 on the constraints. These values are consistent with the representation vector magnitude (Fig. 18), so we can use it as the value for the reliability: $\alpha = ||\vec{v}_r||$.

To visualize the metric, we compose d and α in one figure, in which the deviation is expressed by the color of each vertex and the reliability is the transparency of the color. Fig. 19 shows this visualization: the smaller the deviation, the closer the color is to blue; the larger the deviation, the closer the color is to red.

Fig. 20 and Fig. 21 show two ways to analyze the metric. Fig. 20 has histograms that represent the distribution of the deviation d considering only vertices that have reliability α

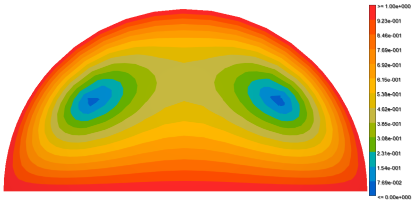


Fig. 18. Representation vector magnitude (α).

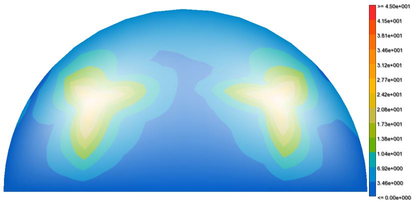


Fig. 19. Alignment quality metric, deviation (d) is the color of each vertex and reliability (α) the transparency of the color.

in a chosen interval. The first interval, $[0.8, 1.0]$, is the more reliable and its color is more opaque. For the other intervals, the reliability decreases and the color gets less opaque. Fig. 20 reveals the good quality of the mesh: the small deviations (good quality) are mostly reliable values, while the larger deviations (poor quality) occur in regions of low reliability.

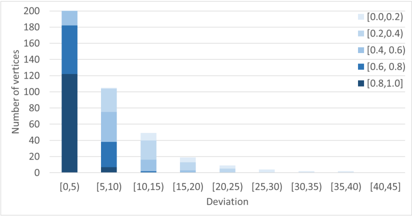


Fig. 20. Deviation graph quantized in α intervals.

Fig. 21 is a frequency histogram of the deviation, where the occurrences are weighted by the reliability. The height of each bar is the sum of the deviation reliability values. Fig. 21 shows that the graph condenses correctly the information of the first graph, allowing an evaluation of the quality mesh distribution.

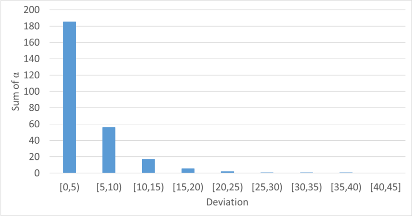


Fig. 21. Condensed bars graph; the height of the bar is weighted by α.

VI. Conclusion

In this work, we used and modified a method proposed by Kowalski et al. [1]. The method generates quadrilateral meshes by using a directionality field. Our work focused on regions with constraints and how to treat this regions in domain partitioning, respecting the directionality alignment.

We then proposed a quality metric for the alignment of quadrilateral meshes. This metric uses the directionality to calculate the angle deviation of each vertex and the magnitude of the representation vector as a reliability criterion. The results show that meshes can have a good quality locally and a bad quality globally. Our metric reflects this in a direct way.

Acknowledgments

This work was sponsored by CAPES (Grant CAPES PROEX 0487 process 1503654) and by Instituto Tecgraf/PUC-Rio.

References

[1] N. Kowalski, F. Ledoux, and P. Frey, "A PDE based approach to multidomain partitioning and quadrilateral meshing," in *Proceedings of the 21st international meshing roundtable*. Springer, 2013, pp. 137–154.

[2] D. Bommes, B. Lévy, N. Pietroni, C. Silva, M. Tarini, and D. Zorin, "State of the art in quad meshing," in *Eurographics STARS*, 2012.

[3] K. Shimada, J.-H. Liao, T. Itoh *et al.*, "Quadrilateral meshing with directionality control through the packing of square cells." in *IMR*, 1998, pp. 61–75.

[4] T. D. Blacker and M. B. Stephenson, "Paving: A new approach to automated quadrilateral mesh generation," *International Journal for Numerical Methods in Engineering*, vol. 32, no. 4, pp. 811–847, 1991.

[5] D. R. White and P. Kinney, "Redesign of the paving algorithm: Robustness enhancements through element by element meshing," in *6th International Meshing Roundtable*, vol. 10, 1997, p. 830.

[6] C. Park, J.-S. Noh, I.-S. Jang, and J. M. Kang, "A new automated scheme of quadrilateral mesh generation for randomly distributed line constraints," *Computer-Aided Design*, vol. 39, no. 4, pp. 258–267, 2007.

[7] C. Araújo and W. Celes, "Quadrilateral mesh generation with deferred constraint insertion," *Procedia Engineering*, vol. 82, pp. 88–100, 2014.

[8] A. Pochet, W. Celes, H. Lopes, and M. Gattass, "A new quadtree-based approach for automatic quadrilateral mesh generation," *Engineering with Computers*, vol. 33, no. 2, pp. 275–292, 2017.

[9] T. Tam and C. Armstrong, "2d finite element mesh generation by medial axis subdivision," *Advances in Engineering Software and Workstations*, vol. 13, no. 5, pp. 313 – 324, 1991.

[10] N. Ray, B. Vallet, W. C. Li, and B. Lévy, "N-symmetry direction field design," *ACM Trans. Graph.*, vol. 27, no. 2, pp. 10:1–10:13, May 2008.

[11] D. Bommes, H. Zimmer, and L. Kobbelt, "Mixed-integer quadrangulation," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 77:1–77:10, Jul. 2009.

[12] M. Tarini, E. Puppo, D. Panozzo, N. Pietroni, and P. Cignoni, "Simple quad domains for field aligned mesh parametrization," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6, p. 142, 2011.

[13] N. Kowalski, F. Ledoux, and P. Frey, "Automatic domain partitioning for quadrilateral meshing with line constraints," *Engineering with Computers*, vol. 31, no. 3, pp. 405–421, 2015.

[14] ——, "Smoothness driven frame field generation for hexahedral meshing," *Computer-Aided Design*, vol. 72, pp. 65–77, 2016.

[15] X. Tricoche, G. Scheuermann, and H. Hagen, "Continuous topology simplification of planar vector fields," in *Proceedings of the Conference on Visualization '01*, ser. VIS '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 159–166. [Online]. Available: http://dl.acm.org/citation.cfm?id=601671.601695

[16] S. D. Conte and C. W. D. Boor, *Elementary Numerical Analysis: An Algorithmic Approach*, 3rd ed. McGraw-Hill Higher Education, 1980.

[17] W. Celes, G. H. Paulino, and R. Espinha, "A compact adjacency-based topological data structure for finite element mesh representation," *International Journal for Numerical Methods in Engineering*, vol. 64, no. 11, pp. 1529–1556, 2005.

[18] Tecgraf/PUC-Rio, "gHEM," http://www.tecgraf.puc-rio.br/pt/software/sw-tectos.html.

[19] ——, "Sigma 2D e 3D (MTOOL e Pos-3D)," http://www.tecgraf.puc-rio.br/pt/software/sw-sigma.html.