Frame Fields for CAD models

David Desobry, François Protais, Nicolas Ray, Etienne Corman, Dmitry Sokolov

Frame Fields for CAD models

David Desobry[1], François Protais[1], Nicolas Ray[1], Etienne Corman[1], and Dmitry Sokolov[1]

INRIA Nancy-Grand Est, 54000 Nancy, France

Abstract. Given a triangulated surface, a unit length tangent vector field can be used to orient entities located on the surface, such as glyphs or strokes. When these entities are invariant under a $\pi/2$ rotation (squares, or curvature hatching), the orientation can be represented by a frame field i.e. four orthogonal tangent unit vectors at each point of the surface. The generation of such fields is a key component of recent quad meshing algorithms based on global parameterization, as it defines the orientation of the final facets. State-of-the-art methods are able to generate smooth frame fields subject to some hard constraints (direction and topology) or smooth constraints (matching the curvature direction). When we have a surface triangular mesh, and a vector defined on each facet, we can't directly know if all the vectors are colinear. We first have to define the (so called) parallel transport of every edge to compare the vectors on a common plan.
When dealing with CAD models, the field must be aligned with feature edges. A problem occurs when there is a low angle corner formed by two colliding feature edges. Our solution not only defines the parallel transport to obtain smoothed frame fields on a surface triangular mesh, it also redefines the parallel transport wherever there is a low angle corner, to smooth a frame field as if these corners' angles were $\pi/2$.

Keywords: Frame Fields · CAD models · Geometry Processing

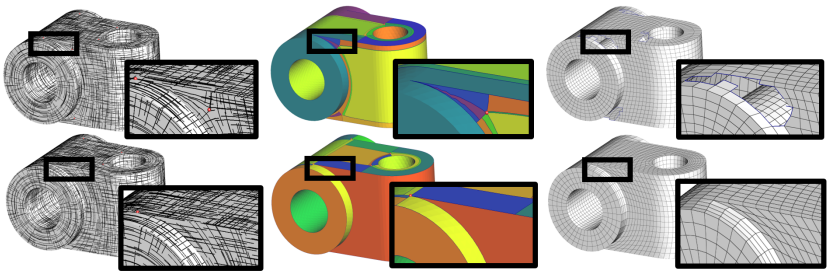


Fig. 1: Producing a frame field with minimal rotation on a CAD model is not sufficient for quad meshing (upper row). Our algorithm allows to handle those sharp edges configurations.

1 Introduction

State-of-the-art quandrangular mesh generation proceeds in two steps: first a guiding frame field is computed, second a parametrization representing the quads is extracted. The frame field defines the orientation of the quads at each point of the domain, whereas the parameterization step determines the vertex positions of the quads. So, frame fields are an intermediate result in the pipeline of quad mesh generation. A high quality quad mesh typically has low distortion and few singular vertices (valence $\neq 4$). Frame fields are produced by a numerical optimization that maximizes the smoothness of the field, naturally preventing creation of singularities: in the vicinity of a singular point a field has a high curvature that is penalized by the optimization.

CAD models are particular surfaces because they have a network of feature edges that must be preserved during the remeshing process. Thus, a feature edge of the input triangle mesh must be a feature edge of the output quandrangular mesh. Unfortunately, these edges can meet with acute angle making it impossible to fit a perfect quadrangle in such a sharp corner, as illustrated in Figure 2. As a consequence, the guiding frame field computed as an intermediate step cannot be turned into a quad mesh and the standard quad meshing method fails. Typical failure cases are shown in Figure 9.

These problematic sharp corners are omnipresent in CAD models as they appear naturally from fillets and chamfers – common designs for mechanical pieces. In the ABC dataset [12] (a collection of one million Computer-Aided Design models) we found that more than 65% of the models have potentially problematic sharp corners. In Thingi10k [22], which is not specialized in CAD models, around 35% of the dataset present at least one sharp corner. This is why non-orthogonality and metric deformation has been a common subject of research recently. In this paper, we propose an easier and much faster way to deal with these problematic cases than previous works.

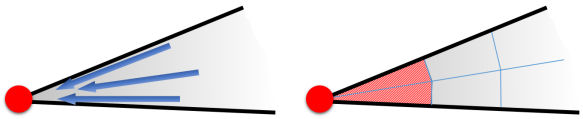


Fig. 2: If two colliding feature edges produce a low angle corner constraint, a classical frame field will align with both feature edge as if they were parallel. It produces a 1/2 singularity that can't be meshed with valid quads.

1.1 Contribution

Our solution automatically produces high quality orthogonal frame field aligned with feature edges with a topology compatible with quad meshing on CAD

models. It then relaxes the orthogonality of the frame field to match even more the feature edges configurations that require non-orthogonality.

These two steps are performed by solving two linear systems, so the impact on performances compared to standard orthogonal frame field generation is very low. Previous works [9, 5, 7] are providing some solutions to solve these problems, but they are much harder to implement, require non-linear optimizations that highly impact performances, and need fine parameter tuning to results close to ours.

The rest of the paper is organized as follows: after a review of related work, we formalize the problem in §2, recall the mathematical background for orthogonal frame field generation § 3.3, and present our first contribution to avoid degenerate fields § 3. In § 4, we further optimize the field curvature by relaxing its orthogonality. We evaluate the quality of our field in § 5 for producing quad meshes.

1.2 Related Work

Frame field generation has been an active research area during the last decade [21]. They were introduced in computer graphics to place hatches in non photo-realistic rendering [6]. However, the main application is quad meshing by global parameterization that was discovered a few years later [17], and improved in [2, 10].

Frame fields are usually produced by numerical optimization. The objective is to minimize the field smoothness, with some minor application dependent modifications like following the surface curvature [17], or better represent a 3D field projected in 2D [7]. Most works are contributing to better optimize the smoothness or to add geometric and topological constraints. The main challenge of frame field generation comes from the field topology (mostly the position and type of field singularities), which requires adding integer constraints in the optimization problem.

The first solution was to use periodic functions where the field topology was "encoded" in the modulus of the function. The idea comes from the observation that if the k^{th} direction of a frame have an angle of $\alpha + k\pi/2$ with a fixed vector, then $\cos(4(\alpha + k\pi/2))$ is the same for all directions. Therefore Hertzmann *et al.* [6] encode a frame by a cosine and minimizes the L^2-norm between adjacent samples. It is, however, more efficient to optimize the field with both the sine and the cosine [17], that is usually referred as representation vector or complex representation. The problem when working with these variables is to preserve the constraint that $\cos^2(4\alpha) + \sin^2(4\alpha) = 1$, making the problem highly non linear. Forcing the norm of the frame field [11, 1] improves the results but seriously impacts performances.

In some cases, control over the field topology is desirable. For instance, to reduce the number of singularities, change their types, or enforce the field behavior along non contractible loops. Complete control over the field topology can be obtained directly [20], or by modification of an existing field [15]. The problem is that the topology is defined by a large set of integers (a bit more than the

index of each point of the surface that can be a singularity), that are very hard to set. In [2, 7], these integers variables are automatically optimized, but could probably be constrained to force some of the topological degrees of freedom.

In practice, automatically generated field topology is not always the best, but it is difficult to choose a topology before generating the frame field. One can fix the topology of an existing frame field [15] as a post-processing. Another possibility is to modify the optimization problem in a way that will favor locally high field curvature against producing singularities on geometric details [19]. Our method is inspired by this work to control the impact of sharp corners.

Non orthogonal frame fields were introduced in [14] for generating planar quad meshes, where directions must be conjugate instead of orthogonal. Changing the metric of the surface [16, 9] naturally stretches the crosses, that loose their orthogonality. However, setting the metric prior to the cross constraints makes it hard to support alignment with feature edges, and the optimization becomes more challenging. Representing the field as root of a complex polynomial [3, 4] also supports non orthogonal frames, but introduces non-linear optimization problems and requires fine parameter tuning. In [7], the 2D frame field is the projection of a frame field defined on a surface, leading to non orthogonal frame fields, obtained by minimizing a different objective function. It should be possible to adapt it for our case, but it makes the solution dependent of a non trivial implementation of their mixed-integer solver. We instead solve the topology with orthogonal frames, then further optimize its geometry by relaxing the orthogonality. It should be possible to produce frame fields similar to ours using these methods, but it requires to adapt complex optimizations framework, whereas our method only requires to solve a couple of linear systems. Moreover, we support extreme distortions on sharp corners that would challenge the solvers.

2 Formalization and our approach overview

Our pipeline takes as input a triangle mesh with identified feature edges, and computes a non-orthogonal, per-triangle constant frame field aligned with the feature edges. The goal is to compute a frame field with the lowest total rotation; the field must not have index 1/2 singularities at sharp corners. In practice, we relax the orthogonality of the field at the very last step, therefore in this section we give formal definitions for the total rotation and the singularity indices both for orthogonal and non-orthogonal fields.

N.B. Our frame field will be represented on faces of a triangulated surface. It makes frames independent across feature edges, thus we can cut the mesh along feature edges without loss of generality. From now on, **feature edges will be considered as boundaries**; it considerably simplifies the notations.

Orthogonal frame fields: a frame on a triangle consists of a set of four tangent unit vectors $\left\{v, v^{\perp}, -v, -v^{\perp}\right\}$. It is common to endow each triangle i with a reference vector b_i and describe the frame as a rotation angle α_i of the reference

vector b_i around the normal. Thus, a frame field can be represented by the set of angles $\{\alpha_i, \alpha_i + \pi/2, \alpha_i + 2\pi/2, \alpha_i + 3\pi/2\}$, refer to Fig. 3–left.

The rotation of an orthogonal frame field is usually measured by the rotation r_{ij} between frames of two adjacent triangles i, j:

$$r_{ij} = -\alpha_i + \alpha_j + c_{ij} + p_{ij}\pi/2, \tag{1}$$

where c_{ij} accounts for the change (once both triangles rotated into a common plane) between the reference vectors b_i and b_j, therefore $-\alpha_i + \alpha_j + c_{ij}$ is the rotation between two adjacent frames. Since the frame is invariant under $\pi/2$ rotation, the integer p_{ij} reflects the matching between the branches of the frames (see Fig. 4). In most cases, p_{ij} is chosen to minimize the rotation. The problem to generate the smoothest orthogonal frame field can be formulated as the minimization of the total field rotation:

$$\arg\min \sum_{ij} r_{ij}^2 \quad \text{subject to boundary constraints.} \tag{2}$$

The last thing we need to define for orthogonal frame fields is the frame field index. For a vertex v we define the index as follows:

$$\mathcal{I}(v) := \frac{1}{2\pi}\left(\sum_{ij \in E(v)} r_{ij} - \theta_v\right) + 1 - \frac{\mathbb{1}_{\partial\Omega}(v)}{2}, \tag{3}$$

where $\mathbb{1}_{\partial\Omega}(v) \in \{0, 1\}$ is the indicator variable telling us whether it is a boundary vertex or not, $E(v)$ denotes the set of dual edges over the 1-ring of the vertex v, and θ_v is the sum of all mesh angles at the vertex (all the triangle corners incident to the vertex). Most common indices are illustrated in Figure 5, they follow the convention used in [13].

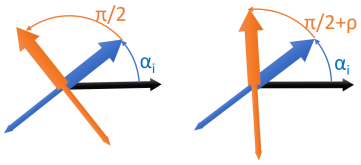


Fig. 3: A non-orthogonal frame (right) has an extra degree of freedom ρ_i pulling the second branch away from orthogonality.

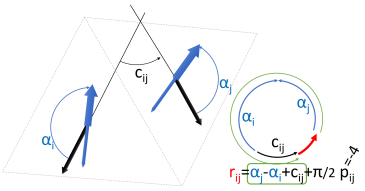


Fig. 4: (α_i) are frame angles, c_{ij} is the rotation between reference vectors of triangles i and j. Integer p_{ij} disambiguates the modulo $\pi/2$.

Non orthogonal frame fields: We want to relax the orthogonality constraint so non-orthogonal frames fit neatly sharp boundary corners. Thus, we need to add

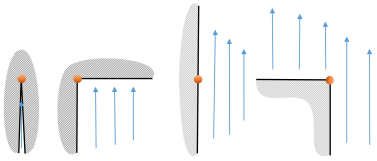


Fig. 5: Index for boundary vertices with a constant field. From left to right: indices $1/2, 1/4, 0, -1/4$.

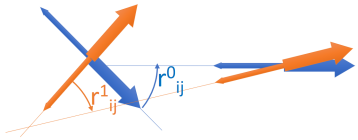
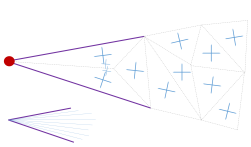
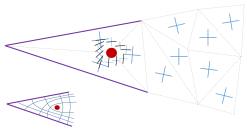


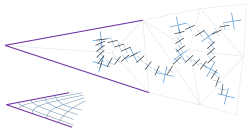
Fig. 6: Non orthogonal frame fields have two angular differences r^0_{ij} and r^1_{ij}, one for each pair of vectors.



(a) Smoothest field



(b) Constrain only the 1-ring of the sharp corner



(c) Propagate the field curvature

Fig. 7: Frame field on a sharp corner: the smoothest field's streamlines are converging to the corner, constraining the curvature of its one ring just moves the singularity to an adjacent vertex, so we need to further propagate the modification of the field curvature.

an extra parameter to the above representation to take into account the skewness of the frame. We introduce the angle ρ_i measuring the deviation from orthogonality; non-orthogonal frames are represented by a set of four unit vectors obtained by rotating b_i by angles $(\alpha_i, \alpha_i + \pi/2 + \rho_i, \alpha_i + 2\pi/2, \alpha_i + 3\pi/2 + \rho_i)$, refer to the right panel of Fig. 3 for an illustration.

Non-orthogonal frames loose the invariance under $\pi/2$ rotation. Thus, it is important to note that the skewness parameter ρ_i is, by convention, always applied to the second and fourth branches. In order to measure how a frame rotates across the edge shared by two adjacent triangles i and j, we need to define two r^0_{ij} and r^1_{ij}. These angles account for the rotation of the first and third branches and the skew second and fourth branches respectively as illustrated in Fig. 6.

Imagine that the first branch of the frame on triangle i is matched with the first (or third) branch of the frame on triangle j (i.e. $p_{ij} = 0 \bmod 2$) then r^0_{ij} is given by Eq. (1). In this case the second branch matches on i the second (or fourth) branch on j and r^1_{ij} is computed by reapplying Eq. (1) with angles $\alpha_i + \rho_i$ and $\alpha_j + \rho_j$.

When $p_{ij} \bmod 2 = 1$, the first branch of the frame on triangle i is matched with the second (or fourth) branch of the frame on triangle j then r^0_{ij} is obtained by replacing α_j by $\alpha_j + \rho_j$ in Eq. (1).

The overall formula can be summarized by:

$$\begin{cases} r^0_{ij} = -\alpha_i + \alpha_j + c_{ij} + p_{ij}\frac{\pi}{2} + (p_{ij} \bmod 2)\rho_j \\ r^1_{ij} = -\alpha_i - \rho_i + \alpha_j + c_{ij} + p_{ij}\frac{\pi}{2} + (1 - (p_{ij} \bmod 2))\rho_j \end{cases} \tag{4}$$

Then the smoothest non orthogonal frame field is the one minimizing the total rotation:

$$\arg\min \sum_{ij} \left(r^0_{ij}\right)^2 + \left(r^1_{ij}\right)^2 \quad \text{subject to boundary constraints.} \tag{5}$$

Our approach overview: as explained in § 3.1, one daunting challenge for quad meshing a CAD model is the omnipresence of sharp corners. It forces index 1/2 boundary singularities to appear in the smoothest frame fields, making it impossible to extract a valid quad-mesh. To avoid this situation the main idea is to trade some smoothness against the appearance of acceptable singularities on sharp corners (Fig. 7). More precisely, we alter the notion of smoothness to ban unmeshable configurations.

So, given a triangle mesh with boundary (recall that all feature edges are considered as a boundary), our algorithm computes two tangent directions per triangle by following the steps described in the subsequent sections:

- § 3: compute an **orthogonal** frame field without index 1/2 singularities at sharp corners. The idea is to change the objective rotation ω (§ 3.2) and then to compute the smoothest frame field with respect to ω (§ 3.3)
- § 4: compute a **non-orthogonal** field that reduces the field rotation w.r.t the orthogonal field. In other words, we relax the orthogonality constraint.

3 Compute a meshable orthogonal frame field

A standard way to compute the smoothest (orthogonal) frame field is to minimize the total field rotation $\sum_{ij} r^2_{ij}$ under boundary constraints. The problem, however, is that this approach often leads to unmeshable frame fields (§ 3.1). To avoid this pitfall, we do not generate frame fields whose rotation is the smallest possible, but rather the closest to a prescribed rotation ω that prevents sharp corner collapses (§ 3.2). We represent ω by a rotation angle ω_{ij} between each pair of adjacent triangles i and j. Once this target rotation has been obtained, the frame field is simply computed by minimizing the distance to the target $\sum_{ij}(r_{ij} - \omega_{ij})^2$, § 3.3 provides all necessary details.

3.1 Problem characterization

A frame field becomes non quad meshable whenever there is a boundary vertex with index of 1/2 (see Figure 2).

Let v be a boundary vertex with a small total angle θ_v. The frame field minimizing the rotation is very likely to be such that $\sum_{ij \in E(v)} r_{ij} = \theta_v$ i.e.

placing an index of $1/2$ on vertex v. Note that Eq. (3) provides a way to detect potential problematic configurations. Indeed, if the total angle at a boundary vertex is $\theta_v < \pi/4$, minimizing the rotations of the frames around this vertex leads to the problematic index $1/2$ singularity, with a total field rotation of θ_v. This case can be avoided by forcing the frame rotation to be as close as possible to a target rotation field ω in this situation.

From a differential geometry point of view, ω is a 1-form that we use to redefine the parallel transport. In contrast to redefinition of the metric [16, 9], it does not modify the orthogonality of the frames.

3.2 Prescribe the target field rotation ω

In this section, we start by reviewing a naive fix: a very local target rotation prescription, and then a global one that gives great quality results.

A naive local fix. The target rotation field ω should promote $1/4$ index singularity in places where the smoothest frame field (Eq. (2)) is likely to place an index $1/2$ singularity. More precisely, for each boundary vertex v with total angle $\theta_v < \pi/2$, setting its index to $1/4$ constrains the sum of field rotation in the 1-ring by Eq. (3): $\sum_{ij \in E(v)} r_{ij} = \theta_v - \pi/2$. We therefore add this constraint to the target rotation, that we write as $\mathrm{d}\omega(v) = \theta_v - \pi/2$, where $\mathrm{d}\omega(v) = \sum_{ij \in E(v)} \omega_{ij}$ denotes the exterior derivative of ω, extended to the boundary of the surface.

Why is it so local? If we simply evenly distribute $\theta_v - \pi/2$ on the ω_{ij} of each dual edge $ij \in E(v)$, it indeed moves the singularity away from the boundary, but only as little as possible. The common case of two adjacent boundary triangles i and j (Fig. 7-middle) is very representative of this behavior. As detailed in Fig. 8, imposing $\mathrm{d}\omega(v) = \omega_{ji} = \theta_v - \pi/2$ will lead to a matching rotation of $r_{ji} = \theta_v - \pi/2$. As we are looking for the smoothest field, the rotation on neighboring dual edges jk and ki will be $r_{jk} = r_{ki} = \theta_v/2$. By Eq. (3), the index of center vertex is then $-1/4$: the singularity has barely moved away from the boundary.

What happened here is that $\mathrm{d}\omega(v)$ is perturbing the energy Eq. (8) needed for choosing the index of v: as observed in [19], it acts as the angle defect. Moreover, if the triangles i, j share the edge v, v', increasing ω_{ij} mechanically increases $\mathrm{d}\omega(v)$ and decreases $\mathrm{d}\omega(v')$ by the same amount. As a consequence, the sum remains null, i.e. $\sum_v \mathrm{d}\omega(v) = 0$. When altering ω only in the 1-ring of v, we are simply moving $\mathrm{d}\omega(v)$ to its neighboring vertex... that naturally becomes the new singular vertex.

A global solution. Instead of forcing the nearest vertex to become a singularity, it is often better to place it a bit farther. To do so, the modification of $\mathrm{d}\omega(v)$ that prevents index $1/2$ on sharp corners should not just be transferred to it's nearest neighbor: it would be better to have $\mathrm{d}\omega(v) = 0$ for all other vertices. Unfortunately, the constraint $\sum_{ij} \mathrm{d}\omega(v) = 0$ makes it impossible, so we instead distribute the constrained $\mathrm{d}\omega(v)$ of sharp corner vertices onto all other vertices.

A two step optimization method can be used to determine ω:

- we compute a scalar $K(v)$ per vertex v such that $\sum_v K(v)^2$ is minimal, $K(v) = \pi/2 - \theta_v$ on sharp corners v, and $\sum_v K(v) = 0$. The solution just

distributes the sum of the $-K(v)$ of the sharp corners on other vertices. This optimization is done independently for each connected component of the surface (that are likely to be generated when cutting the surface along feature edges).
– we then minimize $\arg\min \sum_{ij} \|\omega_{ij}\|^2$ under the constraint $\mathrm{d}\omega(v) = K(v)$. The constraint $\sum_v K(v) = 0$ guarantees the existence of a solution.

This global solution is not always optimal, we don't want the field distorsion to spread too far from the sharp corner and produce global distorsions. In pratice, we initialize $K(v) \leftarrow \pi/2 - \theta_v$ for sharp corners and $K(v) \leftarrow 0$ for other vertices. Then, for each sharp corner v_s, we compute the set of vertices that can be reached by following less than 5 edges. This is a tradeoff that permit to avoid both local or global high distorsion. Let n be the size of this set of vertices, we update K for all vertices of the set : $K(v) \leftarrow K(v) - K(v_s)/n$. We then optimize omega values $\arg\min \sum_{ij} \|\omega_{ij}\|^2$ under the constraint $\mathrm{d}\omega(v) = K(v)$.

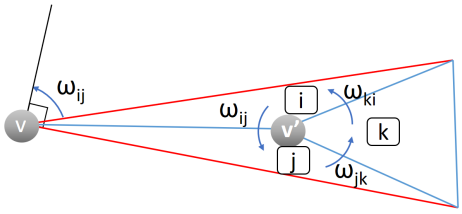


Fig. 8: Local fix of a sharp corner: We set $\omega_{ij} \leftarrow \theta_v - \pi/2$ to force $I(v) = 1/4$. If ω_{jk} and ω_{ki} are null, then $\mathrm{d}\omega(v') = \theta_v - \pi/2$. Therefore, the final frame field will have $r_{ij} = \omega_{ij}$, $r_{jk} = r_{ki} = \theta_v/2$, leading to an index $1/4$ at v'.

3.3 Computing an orthogonal frame field

Let us start with a brief primer on smoothest orthogonal frame field generation (Eq. (2)). The state of the art is to lock boundary frames to have one direction tangent to the boundary, and to minimize the total field rotation. Note that if a triangle has two boundary edges, one can split the triangle in three by adding a vertex at the center to avoid overconstraining the problem .

The smoothest field is the field with minimum rotation r_{ij}. To avoid having to deal with integer variable p_{ij}, a well-known trick is to use a "representation vector" as it is referred to in [18]. The smoothest orthogonal frame field is the one having zero rotation across every dual edge (i.e. $r_{ij} = 0$). Thus in Eq. (1) one can multiply by 4i and take the complex exponential, leading to:

$$e^{4\mathrm{i}\alpha_i} = e^{4\mathrm{i}\alpha_j} e^{4\mathrm{i}c_{ij}} e^{2\mathrm{i}p_{ij}\pi}. \tag{6}$$

Since p_{ij} is integer, the last term is equal to one. We can perform a change of variable $z_i = e^{4\mathrm{i}\alpha_i}$, representing a frame by a unit complex number per triangle.

Then the smoothest frame field is obtained by solving the following optimization problem:

$$\arg\min_{\|z\|=1} \sum_{ij} \|z_i - z_j e^{4\mathrm{i}c_{ij}}\|^2 \quad \text{subject to boundary constraints.} \tag{7}$$

As CAD models have many feature edges that force $\|z_i\| = 1$ on neighboring triangles, a common practice to solve this problem is to use an ordinary least square method and ignore the unit norm constraint. At the end of the process, the variables p_{ij} are set in order to minimize r_{ij} in Eq. (1).

So long for the smoothest frame fields, but recall that we want to compute a frame field whose variation is as close to the prescribed rotation ω as possible. It turns out that this problem can be solved exactly as the smoothest frame field by a simple update of the optimization problem (7):

$$\arg\min_{\|z\|=1} \sum_{ij} \|z_i - z_j e^{4\mathrm{i}(c_{ij}-\omega_{ij})}\|^2 \quad \text{subject to boundary constraints.} \tag{8}$$

As for the variables p_{ij}, they are now set to minimize $r_{ij} - \omega_{ij}$.

4 Relax the orthogonality

At this point, we generate quad-meshable orthogonal frame fields by minimizing $\sum_{ij}(r_{ij}-\omega_{ij})^2$. The geometry can be further optimized by relaxing the orthogonality constraint. The objective here is to minimize $\sum_{ij} \left(r_{ij}^0\right)^2 + \left(r_{ij}^1\right)^2$ as defined in Eq. (4).

Instead of working with variables ρ_i, α_i, we will optimize the rotation angles to apply to each vector of the frame field: γ_i^0 for even vectors and γ_i^1 for odd vectors. The final frame field will be obtained by updating $\alpha_i \leftarrow \alpha_i + \gamma_i^0$, and setting $\rho_i \leftarrow \gamma_i^1 - \gamma_i^0$.

We can rewrite Eq. (4) for the new variables γ_i^0 and γ_i^1:

$$\begin{cases} r_{ij}^0 = -(\alpha_i + \gamma_i^0) + \alpha_j + \gamma_j^{p_{ij} \bmod 2} + c_{ij} + p_{ij}\frac{\pi}{2} \\ r_{ij}^1 = -(\alpha_i + \gamma_i^1) + \alpha_j + \gamma_j^{(p_{ij}+1) \bmod 2} + c_{ij} + p_{ij}\frac{\pi}{2} \end{cases}$$

The new field is obtained by minimizing:

$$\arg\min \left\{ \sum_{ij} \left[\left(r_{ij}^0\right)^2 + \left(r_{ij}^1\right)^2 \right] + \epsilon \sum_i \left(\gamma_i^0 - \gamma_i^1\right)^2 \right\},$$

where $\epsilon = 10^{-2}$. Note that the term $\sum_i \left(\gamma_i^0 - \gamma_i^1\right)^2$ favors the field to be orthogonal, which is necessary for under-constrained configurations such as ring or torus.

To prevent degenerate frames, once the energy is minimized, we check whether $|\gamma_i^0 - \gamma_i^1| > \alpha$, where $0 < \alpha < \pi/2$ is the maximum deviation from the orthogonality. We use $\alpha = 0.45\pi$ in our experiments. If the condition is not satisfied,

we fix the frame locally by applying the smallest possible modification to γ_i^0 and γ_i^1: if $\gamma_i^0 > \gamma_i^1$ then $\gamma_i^0 \leftarrow (\gamma_i^0 + \gamma_i^1 + \alpha)/2$ and $\gamma_i^1 \leftarrow (\gamma_i^0 + \gamma_i^1 - \alpha)/2$, otherwise $\gamma_i^0 \leftarrow (\gamma_i^0 + \gamma_i^1 - \alpha)/2$ and $\gamma_i^1 \leftarrow (\gamma_i^0 + \gamma_i^1 + \alpha)/2$.

5 Results and applications

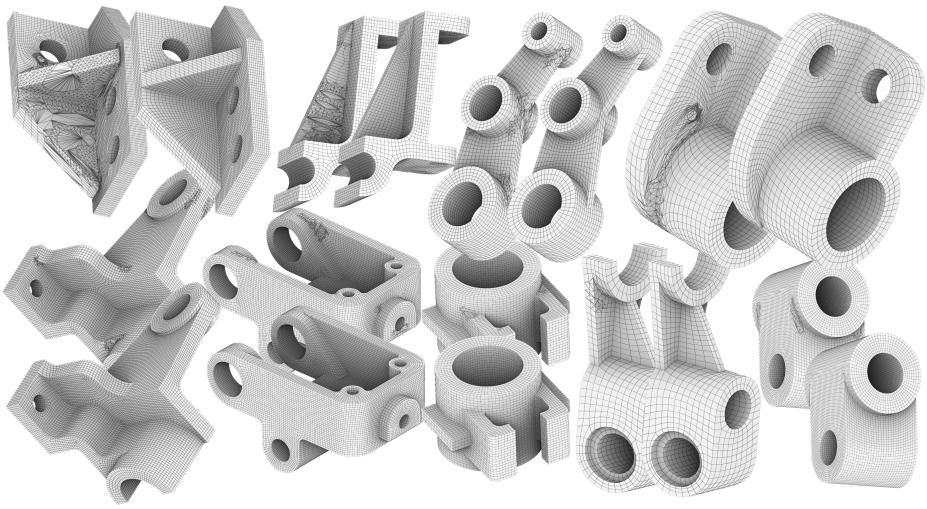


Fig. 9: Each model is remeshed with (foreground) and without (background) fixing sharp corners by our method.

The **running time** was not major consideration in this work because, as long as we stick to solving a couple of linear systems, generating the frame field is not the weakest part of the quad meshing process. Our algorithm is 3 to 5 times slower than the fastest way to produce orthogonal frame fields (without taking the representation vector norm into account). The extra time is consumed by computing ω and relaxing the orthogonality. For example, a model of 35K triangles is optimized in 3 seconds: 0.4 seconds to compute ω, 0.5 seconds to optimize the orthogonal field, and 1.3 to relax the orthogonality. As a comparison, computing the global parameterization take 155 seconds.

5.1 Quad meshing

A family of quad meshing algorithms takes a frame field as input, computes a global parameterization and extract quads from it. To compute the presented results, we used the quadcover algorithm [10]. From a quadcover parameterization, the extracted quad mesh have extraordinary vertices (valence $\neq 4$) only on the singularities of the input frame field.

To work on CAD models with sharp corners, we choosed to act on the input frame field to make it compatible with high quality quad mesh algorithm like [10].

Previous works like [17, 8, 5] handled those CAD models cases from traditional frame fields, but at the cost of apparition of more extraordinary vertices in the output quad mesh.(see Fig 10)

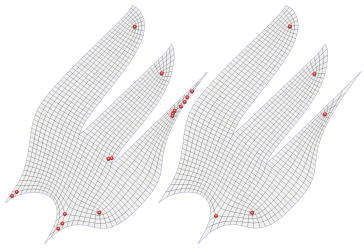


Fig. 10: The quad mesh generated by [5] (left) needs a lot more singularities to deal with feature edges than ours (right).

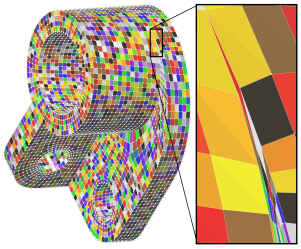


Fig. 11: Our method produces a quad mesh aligned with feature curves even in the presence of extremely sharp corners.

Conclusion

Typical meshing algorithm of the computer graphics domain aren't adapted to handle sharp corners configurations that are omnipresent in CAD models. However, previous work showed that releasing the orthogonality constraint on frame fields is a sufficient fix in many cases.

A geometrical approach of generating non-orthogonal frame field induces non-linear systems that often cost a lot of computation time when compared to the linear systems of an orthogonal frame field.

Our solution solves it in a more combinatorial way, from an orthogonal field. Instead of having to optimize non-orthogonal frames, we redefine the parallel transport near every sharp corners to make a classical orthogonal optimization solves non-orthogonal issues. The relaxation of the orthogonality then permits to obtain a geometrically smoothed non-orthogonal frame fields.

The advantages of the method are that the computation time is as unsignificant as the implementation time compared to classical orthogonal surface frame fields, and the results fields are guaranteed to be meshable with a small amount of extraordinary vertices on sharp corners.

The drawbacks comes from the fact that it is based on combinatory : If the input triangle mesh has inconsistent size, the parameter of the field rotation propagation could be too local or too global. Moreover, if the feature edge detection is of bad quality, the algorithm could try to solve problems that doesn't exist, and so provide a suboptimal solution.

References

1. Beaufort, P., Lambrechts, J., Henrotte, F., Geuzaine, C., Remacle, J.: Computing two dimensional cross fields - A PDE approach based on the ginzburg-landau theory. CoRR **abs/1706.01344** (2017), http://arxiv.org/abs/1706.01344
2. Bommes, D., Zimmer, H., Kobbelt, L.: Mixed-integer quadrangulation. ACM Trans. Graph. **28**(3), 77:1–77:10 (Jul 2009). https://doi.org/10.1145/1531326.1531383, http://doi.acm.org/10.1145/1531326.1531383
3. Diamanti, O., Vaxman, A., Panozzo, D., Sorkine-Hornung, O.: Designing n-polyvector fields with complex polynomials. Computer Graphics Forum **33**(5), 1–11 (Aug 2014). https://doi.org/10.1111/cgf.12426
4. Diamanti, O., Vaxman, A., Panozzo, D., Sorkine-Hornung, O.: Integrable polyvector fields. ACM Trans. Graph. **34**(4) (Jul 2015). https://doi.org/10.1145/2766906, https://doi.org/10.1145/2766906
5. Fang, X., Bao, H., Tong, Y., Desbrun, M., Huang, J.: Quadrangulation through morse-parameterization hybridization. ACM Trans. Graph. **37**(4) (Jul 2018). https://doi.org/10.1145/3197517.3201354, https://doi.org/10.1145/3197517.3201354
6. Hertzmann, A., Zorin, D.: Illustrating smooth surfaces. In: PROCEEDINGS OF SIGGRAPH 2000. pp. 517–526 (2000)
7. Iarussi, E., Bommes, D., Bousseau, A.: BendFields: Regularized Curvature Fields from Rough Concept Sketches. ACM Transactions on Graphics **34**(3) (Apr 2015). https://doi.org/10.1145/2710026, https://hal.inria.fr/hal-01261456
8. Jakob, W., Tarini, M., Panozzo, D., Sorkine-Hornung, O.: Instant field-aligned meshes. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) **34**(6), 189:1–189:15 (Nov 2015). https://doi.org/10.1145/2816795.2818078
9. Jiang, T., Fang, X., Huang, J., Bao, H., Tong, Y., Desbrun, M.: Frame field generation through metric customization. ACM Trans. Graph. **34**(4) (Jul 2015). https://doi.org/10.1145/2766927, https://doi.org/10.1145/2766927
10. Kaelberer, F., Nieser, M., Polthier, K.: QuadCover - Surface Parameterization using Branched Coverings. Computer Graphics Forum (2007). https://doi.org/10.1111/j.1467-8659.2007.01060.x
11. Knöppel, F., Crane, K., Pinkall, U., Schröder, P.: Globally optimal direction fields. ACM Trans. Graph. **32**(4) (2013)
12. Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., Panozzo, D.: Abc: A big cad model dataset for geometric deep learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
13. Liu, H., Zhang, P., Chien, E., Solomon, J., Bommes, D.: Singularity-constrained octahedral fields for hexahedral meshing **37**(4) (Jul 2018). https://doi.org/10.1145/3197517.3201344, https://doi.org/10.1145/3197517.3201344
14. Liu, Y., Xu, W., Wang, J., Zhu, L., Guo, B., Chen, F., Wang, G.: General planar quadrilateral mesh design using conjugate direction field. ACM Trans. Graph. **30**(6), 1?10 (Dec 2011). https://doi.org/10.1145/2070781.2024174, https://doi.org/10.1145/2070781.2024174
15. Palacios, J., Zhang, E.: Rotational symmetry field design on surfaces. ACM Trans. Graph. **26**(3) (Jul 2007). https://doi.org/10.1145/1276377.1276446, http://doi.acm.org/10.1145/1276377.1276446

16. Panozzo, D., Puppo, E., Tarini, M., Sorkine-Hornung, O.: Frame fields: Anisotropic and non-orthogonal cross fields. ACM Trans. Graph. **33**(4) (Jul 2014). https://doi.org/10.1145/2601097.2601179, https://doi.org/10.1145/2601097.2601179
17. Ray, N., Li, W.C., Lévy, B., Sheffer, A., Alliez, P.: Periodic global parameterization. ACM Trans. Graph. **25**(4), 1460–1485 (Oct 2006). https://doi.org/10.1145/1183287.1183297, http://doi.acm.org/10.1145/1183287.1183297
18. Ray, N., Sokolov, D., Lévy, B.: Practical 3d frame field generation. ACM Trans. Graph. **35**(6) (Nov 2016). https://doi.org/10.1145/2980179.2982408, https://doi.org/10.1145/2980179.2982408
19. Ray, N., Vallet, B., Alonso, L., Levy, B.: Geometry-aware direction field processing. ACM Trans. Graph. **29**(1), 1:1–1:11 (Dec 2009). https://doi.org/10.1145/1640443.1640444, http://doi.acm.org/10.1145/1640443.1640444
20. Ray, N., Vallet, B., Li, W.C., Lévy, B.: N-symmetry direction field design. ACM Trans. Graph. **27**(2), 10:1–10:13 (May 2008). https://doi.org/10.1145/1356682.1356683, http://doi.acm.org/10.1145/1356682.1356683
21. Vaxman, A., Campen, M., Diamanti, O., Bommes, D., Hildebrandt, K., Technion, M.B.C., Panozzo, D.: Directional field synthesis, design, and processing. In: ACM SIGGRAPH 2017 Courses. SIGGRAPH '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3084873.3084921, https://doi.org/10.1145/3084873.3084921
22. Zhou, Q., Jacobson, A.: Thingi10k: A dataset of 10,000 3d-printing models. arXiv preprint arXiv:1605.04797 (2016)