ESAIM: PROCEEDINGS, August 2008, Vol. 24, p. 60-76
C. Dobrzynski, P. Frey, Ph. Pebay, Editors

CO-REFINEMENT OF FAULT SURFACES : CONVEXIFICATION PROCESS

L. Agelas[1], A. Benali[1], S. Benteboula[2], A. Claisse[3], P. Havé[1] and A. Loseille[4]

Abstract. The aim of this work is to develop an algorithm that allows the common refinement of non-coincident meshes composed of arbitrary 3D surfaces elements (triangles, quad elements). This study is motivated by computations in geologic applications which involve complex geometries with heterogeneous components and geologic faults. The resulting meshes are linked through a continuous bijection in order to ensure the accuracy and the conservativity of the data transferring through the surfaces. The strategy adopted consists, first, of the simultaneous convexification of the two surfaces by the mean of points connections. A projection according to given normals is then achieved. The mesh quality improved by applying a surface smoothing should accelerate the procedure convergence.

Résumé. L'objectif de ce travail est de mettre au point un algorithme qui permet le co-raffinement de maillages non coïncidents composés d'éléments de surfaces 3D arbitraires (triangles, quadrangles). Cette étude est motivée par les simulations pour les applications géologiques qui impliquent des géométries complexes avec des composants hétérogènes et des failles géologiques. Les maillages fournis par le co-raffinement sont associés par une bijection bi-continue afin d'assurer la precision et la conservativité du transfert des données à travers les surfaces. La stratégie adoptée consiste, en premier, en la convexification simultanée des deux surfaces par connexions des sommets. Une projection suivant des normales données est ensuite appliquée. La qualité des maillages est améliorée en utilisant une technique de lissage de surface afin d'accélérer la convergence de la procédure.

Introduction

Modern geologic simulations on complex geometries require specific modeling tools for the accurate representation of heterogeneous components. The associated meshes evolve according to a displacement field provided by a geologist, which must preserve the initial heterogeneity distribution. The displacements are defined a priori at a few time steps, and then continuously interpolated between two given positions. Moreover, topological modifications may appear such as fractures, defining geologic faults. Computing transport of information through a fault is one of the goal of this project. The co-refinement (common refinement) of two non-coincident mesh surfaces aims at providing a mesh which links the refined surfaces by mean of a continuous bijection. This co-refinement must allow various numerical strategies through the faults, either relying on volume mesh adaptation or specifically designed numerical schemes in the context of extended stencil finite volume methods.

[1] IFP (Institut Français du Pétrole), 1-4 avenue de Bois Prau, 92500 Rueil Malmaison, France. e-mail: Leo.Agelas@ifp.fr, Abdallah.Benali@ifp.fr & Pascal.Have@ifp.fr

[2] UPMC Univ Paris 06, UMR 7598, Laboratoire J.-L. Lions, F-75005 Paris, France. e-mail: benteboula@ann.jussieu.fr

[3] UPMC Univ Paris 06, UMR 7598, Laboratoire J.-L. Lions, F-75005 Paris, France, supported by a grant from the Région Ile de France. e-mail: claisse@ann.jussieu.fr

[4] INRIA Rocquencourt, Le Chesnay, France. e-mail: Adrien.Loseille@inria.fr

Article published by EDP Sciences and available at http://www.edpsciences.org/proc or http://dx.doi.org/10.1051/proc:2008030

The first part of this project consists in defining a co-refinement algorithm which satisfies some properties ensuring the stability of the scheme. The second part consists in developing a prototype to test the algorithms on realistic non-coincident meshes (with large gaps, overlaps, non planar surfaces...).

A more accurate outline of the fault context may help to discuss on.

A fault is defined as the gap between two volumes. Then a fault is modeled by the two oriented surface meshes provided by the boundaries of the two unstructured volume mesh. However, due to the independent evolution of the volume meshes, some overlaps may appear (figures 1, 2). By construction, our fault surfaces are in general surface open mesh without holes, but we can only consider triangle elements. By the applications, the two fault surfaces are close but discretized by two independent ways. Building a relevant continuous bijection between their surfaces (where it is possible) is a compulsory step towards more complex simulations.

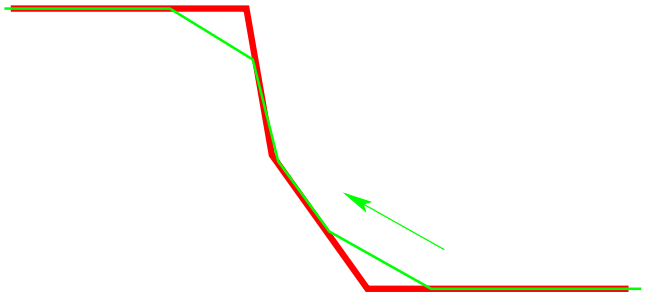

FIGURE 1. 2D sliding meshes. Green mesh slides on the red one. Gaps occur since meshes slide with their own mesh, without adaptation.

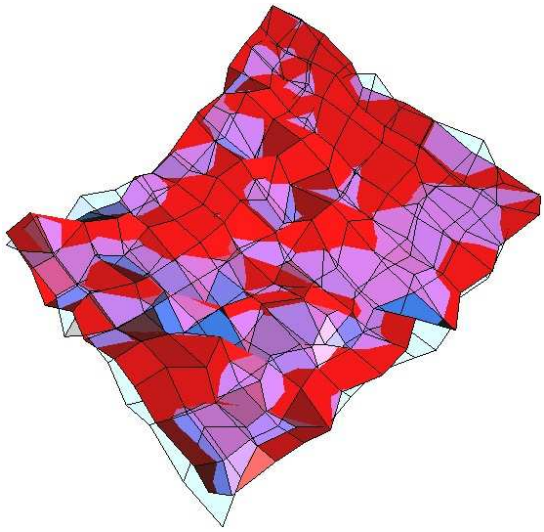

FIGURE 2. A 3D surface fault defined by two surface meshes. The transparent light blue mesh makes gaps and overlaps with the red one.

The remainder of this paper is organized as follows: first, the existing methods, not adopted for reasons of non-suitability for our case, are overviewed. Section 2 presents a reformulation of our problem and the algorithms developped to perform the convexification. Finally, in section 3, our different works during CEMRACS are described and some numerical results are presented.

1. Overview of available technics

At the beginning of the study, the co-refinement of surfaces seemed to be a classical and simple algorithm. Our constraints were to define a fast and self-consistent algorithm during the Cemracs summer school (4 weeks). Note that, the bijection we seek must be explicit and constructive, so that we will able to compute properties on faces in bijection.

The first step was to study available techniques and knowledge in the open litterature. In this way, we obtain some leads.

1.1. **Levelsets methods**

The goal of levelsets methods [1, 8] is to move fronts and interfaces (of codimension one bounding an open domain) by an evolution equation (PDE), for example to transport a surface Γ_1 to a second surface Γ_2.

One of the advantages of this method is to manage topological variations like the variation of the connected component variation. To adapt this method for the capture of the final surface, this technique uses some mesh tools and adaptative remeshing which are not freely available in IFP.

This method was judged to much costly (require to solve several linear system) and not accessible in a short time (require to develop complex mesh tools).

1.2. **Surfaces parametrization methods**

The surfaces parametrization techniques examined here come from the imagery methods such as morphing (e.g. faces). These techniques rely on the transformation of the one-to-one surfaces. Methods for parametrization of 3D meshes have been studied by a number of researchers [2, 9, 11]. Most of them map the surface of the mesh to a plane by solving linear equation systems while others build a mapping between the mesh surface and the surface of a sphere and have to solve systems of nonlinear equations. Nevertheless, these techniques need the prior knowledge of some common positions (e.g. locations of the eyes) in order to obtain a reasonable transformation. Moreover, some algorithms add constraints to reduce distortion.

1.3. **Temporary volume mesh**

More convenient techniques based on volume meshes have been considered. These techniques require compatible boundary surfaces. Then a volume mesh with these surfaces (either a covering volume mesh or a inter-surface volume mesh) is not obvious, moreover when the surfaces are overlapped (no guarantee of obtaining a valid mesh in terms of forcing front). First, these techniques rely on complex and non free tools not available in this form at the IFP. Second, we know that the automaticity and reliability are not guaranteed (even if they are high quality tools). Therefore, any technique employing this plea was not accepted.

2. Problem description and general algorithmic

Our objective is to define a continuous bijection between the two surface sections. Hence, we have to establish a bijection when this is possible and to locate the sections where there is no possibility to get a bijection.

The first natural choices are to consider the orthogonal projection or the projection using the closest point. Since the surfaces are not coincident, these projection appeared to be unable to provide a continuous bijection for different reasons illustrated by the examples in figure 3. As it can be seen from the figure 3.a, the orthogonal projection is not its own inverse (non involutary). The figure 3.b shows that the projection of the vertex p is not univocal (more than one solution). The projection by the closest point is also discarded because it is not injective as shown in figure 3.c.

Taking into account the previous cases, we opted for a projection using a continuous direction on the face [5] which may be a continuous bijection if there is no non visible points as shown in figure 4.

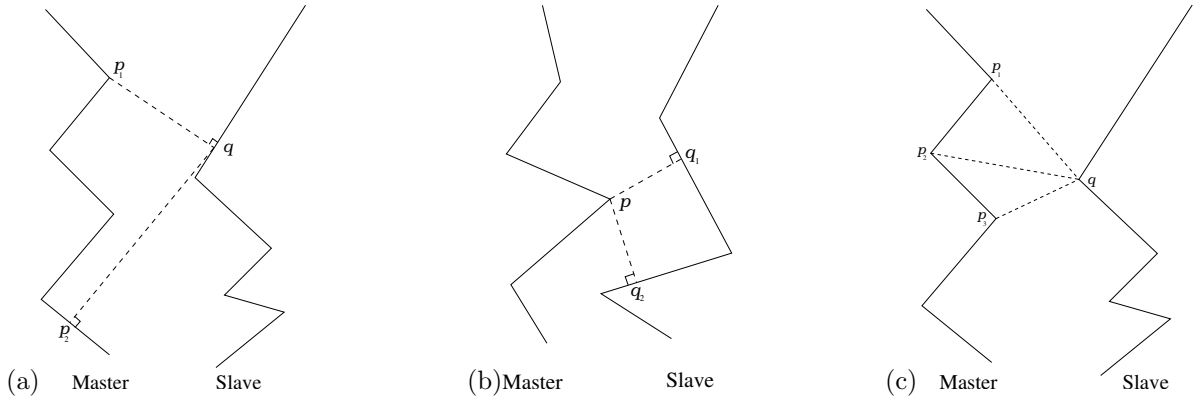

FIGURE 3. Examples of non bijective projections with $p \in \Gamma_0$ and $q \in \Gamma_1$. (a): orthogonal projection **non involutary**, (b): orthogonal projection **non univocal**, (c): projection by the closest point non **non injective**

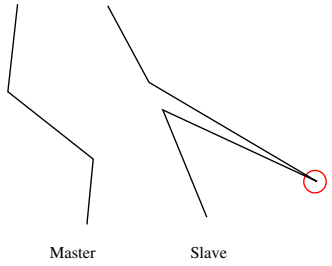

FIGURE 4. Example of surface with non visible point

2.1. **Bijective projections between smooth surfaces**

For simplicity, we wrote here only 2D calculations. There aren't no more difficulties in 3D.

Jiao and Heath [5, 6] defines a continuous projection ψ between two surfaces. This projection is based on a directional projection between Master and Slave surfaces. We summarize this process as follows.

Let $\Gamma_0 = \bigcup_{i=1}^{N} A_i$ and $\Gamma_1 = \bigcup_{i=1}^{M} B_i$ be two non-coincident surfaces. Γ_0 is considered as the Master surface, if it defines projection directions, and Γ_1 is considered as the Slave surface.

Let $d : p \in \Gamma_0 \to \mathbb{R}^3$ be a projection direction defined by the continuous normal at node $\vec{d}(p)$.

For all node $p \in \Gamma_0$, a projection direction $d(p) = \sum_i \beta_i \vec{d_i}$ is assigned, with $p = \sum_i \beta_i A_i$, where β_i are the barycentric coordinates and $\vec{d_i}$ are the normals at vertices A_i.

The projection is made in two steps.

2.1.1. *Direct "Master-Slave" projection:* $\Gamma_0 \rightarrow \Gamma_1$

This part is the most straightforward case.
q is the projection of p according to the direction d on the edge $B \subset \Gamma_1$ ($B = [B_0B_1]$). So for:

$$\left\{ \begin{array}{l} q = p + \gamma d(p) \\ q = \alpha B_1 + (1-\alpha)B_0 \end{array} \right. \quad \text{with} \quad 0 \leq \alpha \leq 1 \quad \text{and} \quad \gamma \in \mathbb{R}$$

where γ is the distance between the points p and q. We have just to solve the following linear system

$$\vec{B_0p} = \alpha \vec{B_0B_1} - \gamma d(p)$$

It should be noted that the projections on the Slave surface are not necessarily vertices of Γ_1. As a consequence, we have to create some new points on the Slave surface mesh and there exist also nodes on Γ_1 which have not inverse image on Γ_0. Therefore, we need to perform a second projection in the inverse direction to get the continuous bijection that we are looking for.

2.1.2. *Inverse "Slave-Master" projection:* $\Gamma_1 \rightarrow \Gamma_0$

This part is more interesting since it uses an implicit definition of the projection.
p is the projection of q according to the direction d on the edge $A \subset \Gamma_0$ ($A = [A_0A_1]$). So for

$$\left\{ \begin{array}{l} q = p + \gamma d(p) \\ p = \beta A_1 + (1-\beta)A_0 \end{array} \right. \quad \text{with} \quad 0 \leq \beta \leq 1 \quad \text{and} \quad \gamma \in \mathbb{R}$$

This leads to the resolution of a non linear system.

1– Two edges intersection

Let $A = [A_0A_1]$ be an edge of the Master surface Γ_0 and $B = [B_0B_1]$ an edge of the Slave surface Γ_1. In contrast with the 2D case, here we look for $q \in B$ and $p \in A$ such as :

$$q = p + \gamma d(p)$$

by writing

$$\left\{ \begin{array}{rcl} q & = & \alpha B_1 + (1-\alpha)B_0 \\ p & = & \beta A_1 + (1-\beta)A_0 \\ d(p) & = & \beta d_1 + (1-\beta)d_0 \end{array} \right. .$$

This leads to solve the following set of non linear equations

$$\beta \vec{A_0A_1} + \alpha \vec{B_1B_0} - \gamma d_0 = \vec{A_0B_0} + \alpha\gamma(d_1 - d_0). \tag{1}$$

2– Node-face intersection

As specified above, the node-face intersection is made of two steps:
– Projection Master node onto Slave face (straightforward)
– Projection Slave node onto Master face :
As in the previous step, we have

$$q = p + \gamma d(p)$$

Let T be a triangulation of the Master face obtained only from its vertices. We proceed by seeking p the image of the point q by the projection on a triangle (A_0, A_1, A_2) of the face. So the barycentric form can be written as follows

$$\left\{ \begin{array}{rcl} p & = & A_0 + \alpha \vec{A_0A_1} + \beta \vec{A_0A_2} \\ d(p) & = & d_0 + \alpha(d_1 - d_0) + \beta(d_2 - d_0) \end{array} \right.$$

Thus, we obtain the following non-linear set of equations:

$$\alpha A_0\vec{A}_1 + \beta A_0\vec{A}_2 + \gamma d_0 = A_0\vec{q} - \gamma\alpha(d_1 - d_0) + \beta(d_2 - d_0)$$

Works for improving this numerical algorithm are still on going.

2.2. **Algorithm for non-smooth surfaces**

The previous directional projection defines a surjection between parts of the Master and Slave surfaces.

In the following, we only consider Γ_0 and Γ_1 as their restricted parts where this surjection defined. By continuity, we can prove that surjective parts are connected. So, non bijective parts are deduced from the non surjective parts and are usually close to boundary of these surfaces.

Even if this surjection is only well defined on parts of these surfaces, this algorithm helps to define a bijection between surfaces as long as direction vectors defined on vertex of Master surface do not cross in front of the Slave surface (figure 5). This fact assumes that the surfaces are close, smooth and well discretized, but our fault

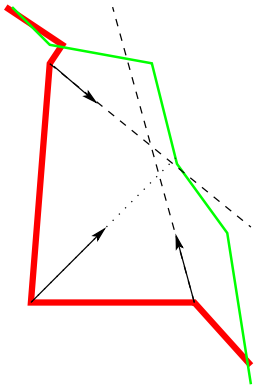

FIGURE 5. 2D example of crossing directions of projection in front of the Slave surface

surface meshes may not always satisfy these constraints.

However, we know that an orthogonal projection τ on a convex surface is well defined, so that, if there is no invisible point (figure 4), this projection defines a bijection.

Furthermore, a directional projection $\psi = \tau^{-1}$ from a convex Master surface to a Slave surface without non visible point is also well defined.

Hence, the main idea is to define a bijection ψ, more precisely a homeomorphism, between the smoothed surfaces $\tilde{\Gamma_0}$ and $\tilde{\Gamma_1}$ themselves in bijection with the original surfaces Γ_0 and Γ_1 through ϕ_0 and ϕ_1 respectively. The final transformation sought is then written as the composition $\phi_0 \circ \psi \circ \phi_1^{-1}$. It follows that we need to construct these functions ϕ_i.

We will build smooth surfaces $\tilde{\Gamma_0}$ and $\tilde{\Gamma_1}$ as the (oriented) convex hulls of the initial surfaces Γ_0 and Γ_1. It is not mandatory to build the convex hull of the Slave surface but the transformation acts also as a smoother which will remove non visible points.

Let remark that, when we consider two fault surfaces, a global convexification of each surface is not compulsory. To obtain a well defined projection we only need a local convex behavior. This *local* behavior is parametrized by the relative local distance between the surfaces. The part will not be exposed here.

As the main step of the bijection algorithm, we will consider the full convexification procedure of one surface Γ_0 which leads to consider the projection between Γ_0 and a plane at infinity. This is the goal that we want to achieve.

3. Convexification of a surface

We seek an explicit and constructive bijection. A way is to consider an iterative algorithm which *converges* to the convex hull of the surface (in a sense to define). Each step of the convexification algorithm contributes to the bijection ϕ which transforms Γ to $\tilde{\Gamma}$. Hence, each step of this iterative method must be a local bijective transform which improves the convexity and the transform ϕ will be the composition of all these steps.

3.1. 2D Convexifaction

The 2D convexification is a convenient way to introduce the convexifaction algorithm (Algorithm 3.1).

Algorithm 3.1. *Iterative 2D convexification algorithm*

repeat
  foreach *non-constrained node p (any order)* **do**
    if $\mathcal{V}_p$ *neighborhood is (already) convex* **then**
      Skip this node
    end
    Build edge e by connecting neighbor nodes of p.
    Project node p and all nodes associated to V_p *on edge e.*
    Remove $\mathcal{V}_p$ *from the mesh.*
  end
until *a convexification operation has not been done successfully ;*

This algorithm replaces concave neighborhood $\mathcal{V}_p$ of a node p by a convex approximation (a new edge e). However, this algorithm acts only on non constrained nodes, that are in our applications the interior nodes, which are non boundary nodes. As described before, the direction $d(p)$ defines first on each node p the direction of projection and moreover, the local orientation of the surface (as a concave or convex neighborhood). This continuous direction is used even for the associated nodes. An associated node p' is a node which has been removed in a local convexification step ($\mathcal{V}_{p'}$ was concave) and projected on the new edge e. At this level, p' becomes a non physical nodes of the mesh. That is why each edge of the mesh may support several associated nodes p'_i which may be projected using directions $d(p'_i)$ if their edge support is removed in a local convexification step (figure 6).

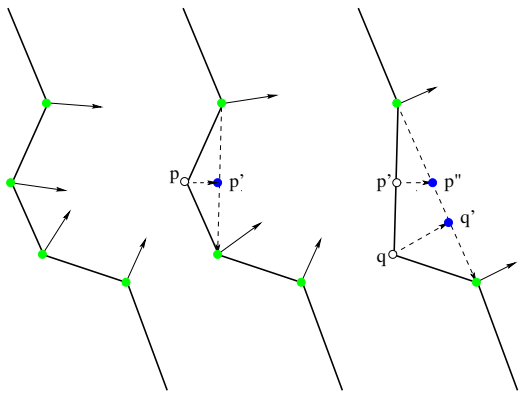

Figure 6. Example of simultaneous convexification procedure and points projection in 2D case. First, $\mathcal{V}_p$ is detected as concave. Then, one replaces it by a new edge and projects p following $d(p)$ as an associated node p'. At the next step, the same process is applied for q which implies to project p' as p'' following $d(p')$.

3.2. **3D convexification**

The 3D convexification algorithm is not as straightforward as the 2D one.

First, the local convexification step of the 2D algorithm may be seen in two different ways:

(1) the merge of a node to a neighbor node.
(2) the connection of the neighbor nodes by a new mesh. In 2D case, this is reduced to an edge line.

By these ways, the local convexification is more or less ill posed. First, how to move a node on a surface without folding its neighborhood ? Second, how to mesh a 3D polygonal curve without any interior point ?

At this point, we have chosen to try the first approach. Indeed, when we study how to build a surface mesh from its boundary, we observe that this mesh is not only constrained by the neighborhood but also by the entire mesh (figure 7): a local reconstruction may intersect a non local part of the mesh. The only way we find to solve efficiently this is to include the surface mesh into a volume mesh. This new volume mesh helps to avoid auto-intersection of the surface. However, as described in the first part, at this level of this work, we avoid any dependency to a complex volume mesh generator.

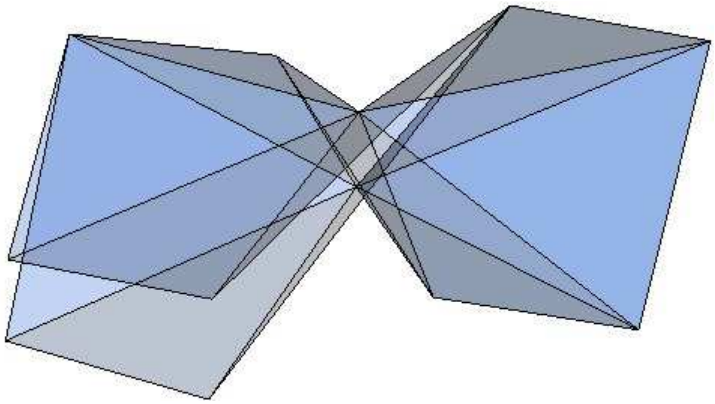

FIGURE 7. Example of complex surface which consists in the superposition of two saddle points. If one of these saddle points is merged, the surface becomes self-intersected. This is not realistic fault surface but may appear during the convexification algorithm.

This problem may also appear with the merge of two nodes, but we have estimated that this approach is simpler (a posteriori, at this level, this choice is not so clear).

Moreover, to ensure an iterative algorithm which *converges* towards the convex hull, we will only allow steps which improve the convexification. In this way, we will define two operations:

(1) The node merge: removing a node which is not on the convex hull. Hence, this step reduces the complexity of the surface.
(2) The edge swap: swapping an edge which is not on the convex hull.

We cannot prove a formal convergence of the algorithm, but this improve the convexity either by reducing the number of free nodes with a local node convexification (which implies a convergence to something since the number of nodes is finite) or by a local edge convexification (which is stable if the orientation of the surface is well defined).

3.2.1. *Node merge*

A node is not on the convex hull, if its neighborhood is not convex (according to the orientation of the surface).

A node p may be merged, only if it is not on the convex hull. The convex hull is implicitly defined when all nodes and edges local neighborhood are convex. Once the target node q selected, the merge operation is

performed as removing p from the mesh and replacing p by q in all local faces connected to p. This operation may produce degenerated faces which are then removed. Moreover, we want to ensure that after each merge the surface mesh is still consistent.

3.2.2. *Edge swap*

This step requires that the surface is only composed by triangles.

An edge may be swapped only if the associated fold is not convex according to the orientation of the surface.

The convexity of a fold is easily estimated by computing the sign of the associated tetrahedron volume. Following figure 8, for swapping edge n_3n_4, we compute sign of the volume of tetrahedron $n_1n_3n_4n_2$. Since this one is positive, this fold is concave. Then, we check if this swap will imply new concave folds (what we want to avoid). For this, we check if the projection signs $\overrightarrow{n_3n_4} \cdot \overrightarrow{n_3n_1}$, $\overrightarrow{n_3n_4} \cdot \overrightarrow{n_3n_2}$, $\overrightarrow{n_4n_3} \cdot \overrightarrow{n_4n_1}$ and $\overrightarrow{n_4n_3} \cdot \overrightarrow{n_4n_2}$ are all positive what ensure that n_3n_4 and n_1n_2 edge projections are intersected (this may also be interpreted as triangle orientations).

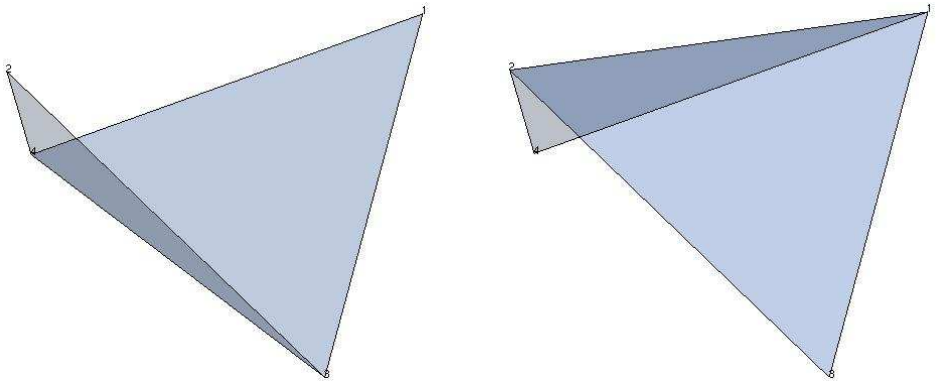

FIGURE 8. Swapping edge 3-4 (left figure) gives a new edge 1-2 (right figure).

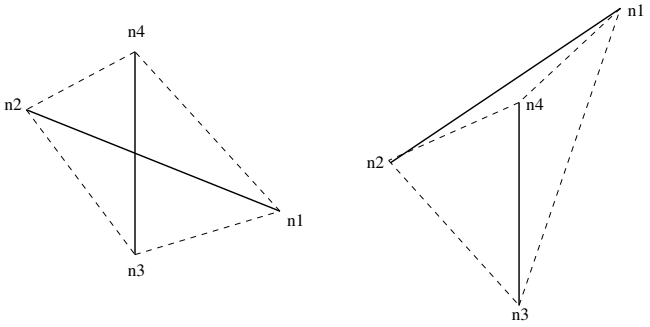

FIGURE 9. According to triangle orientations, a good candidate (left figure) and a bad candidate (right figure) for the swap of edge n_3n_4.

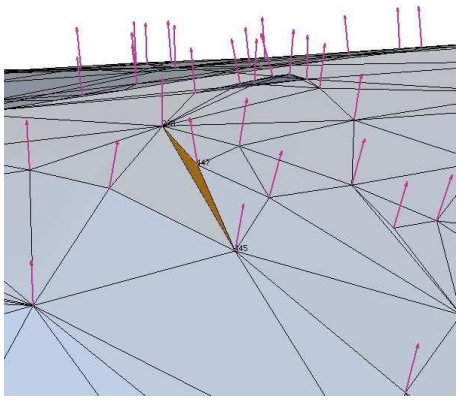

FIGURE 10. Real example of a edge swap. Edge 241-145 will be swapped and connect node 147 to a new neighbor.

3.2.3. *3D Algorithm*

Algorithm 3.2 follows the 2D algorithm including the node merge and edge swap as defined above. Most of this algorithm used the same ideas. First, it processes only non constrained nodes and edges. The main difference is the strict definition of a convex area. We use an heuristic one which defines saddle points as not concave and not convex. That is why we exclude them from the process. However, the exact definition of a saddle point is not easy to use at the numerical level: a saddle point comes from two local curvatures with different signs [3,12]. The main problem comes from the estimation of the local behavior (saddle point, concave, convex) around a node. The projection direction as defined before helps but may also introduce some confusing: the weights used on the summation of neighbor normals must represent the local behavior, independently of the number or the size of the neighbor elements. Some experiences suggest that a weight giving the angle around the node is more robust. Indeed, this local behavior implies the definition of a *best* neighbor node, in the sense that it will not break the mesh geometry (self-intersection, non manifold surface...). But, here, we see it through mesh topology point of view. That is why a geometric improvement will follow in a next section.

Algorithm 3.2. *Iterative convexification algorithm*

repeat
  foreach *non-constrained node i (any order)* **do**
    if $\mathcal{V}_i$ *is saddle point* **then**
      Skip this node
    end
    if $\mathcal{V}_i$ *is (already) convex* **then**
      Skip this node
    end
    Save local mesh context
    Merge node i with its best neighbor
    if *Mesh topology is broken* **then**
      Restore local mesh context
    end
  end
  foreach *non-constrained edge e (any order)* **do**
    Save local mesh context
    if *fold around e is concave* **then**
      Swap edge e
    end
    if *Mesh topology is broken* **then**
      Restore local mesh context
    end
  end
until *a convexification operation has not been done successfully ;*

The local behavior around a node consists in defining if it is a saddle point (with a cone criterion) and its orientation. Many methods may help to compute this and works are still to do for a better definition.

Some of them use a local characterization by the curvatures (elliptic, hyperbolic, parabolic) with quadric fittings. However, these thecniques are limited to the curvatures of the first orders. In that sense, a high order curvature may produce a not detected saddle point.

Instead of this curvature characterization, we have defined as a *cone* like area, each area where the oriented sum of the angles around a node is below than 2π. This is not an equivalence with a non saddle point area but we prefer to reduce the convergence speed rather than damaging the mesh coherence.

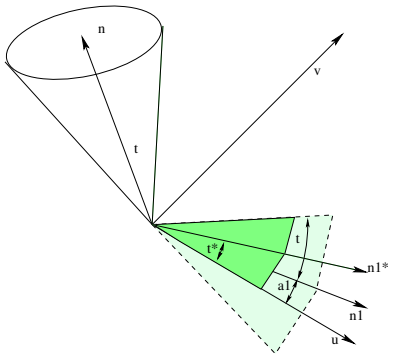

FIGURE 11. Inscribed cone $\mathcal{N}(\vec{n}, \theta)$ process.

Another approach was to compute the inscribed cone (or visibility cone) around a node (figure 17). In this approach, we use an iterative process over the neighbor triangles where one constraint parametrized cone projection. Hence, the cone $\mathscr{N}(\vec{n},\theta)$ is parametrized by an angle θ and a direction $\vec{n}$. Starting with $\mathscr{N}_0 = \mathscr{N}(\vec{n_0}, 2\pi)$ (any $\vec{n_0}$), the constraint is that the cone $\mathscr{N}_i$ must have a projection in the basis defined by 2 successive triangles (coplanar triangles are eliminated). Then, we define $\mathscr{N}_{i+1}$ as the constraint given by the intersection between $\mathscr{N}_i$ and its projection. This algorithm is described in an annex section.

3.3. **Convexification improvements**

3.3.1. *Geometry preservation*

Even if the last algorithm works for many realistic cases, sometimes, the procedure creates self-intersections for complex surfaces close to be self-intersected. This behavior cannot be solved by the standard algorithm (figure 12).

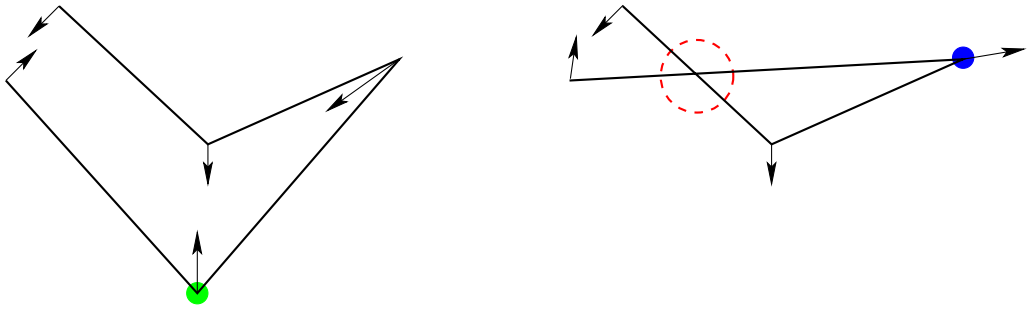

FIGURE 12. Self-intersection of the surface (right figure) due to the local convexification around a point (left figure)

Instead of a complex volume mesh as we discussed at the beginning of this part, we can use an approximation based on a regular grid. By this grid, we tag all cells containing the surface (not only the vertices). Then, when the algorithm suggests a local convexification, we can check that the new faces created in this way do not intersect the mesh by checking only cells of the grid which contains a tag from the original mesh and a tag from the local convexification. If a intersection between the original mesh (without the neighborhood of the removed point) and the local convexification mesh is found, the local convexification will not be allowed.

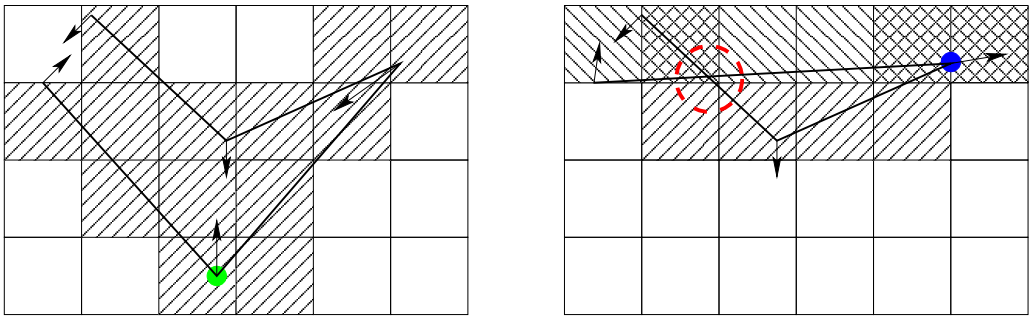

FIGURE 13. Detection of a self-intersection of the surface (right figure). Original tag is denotes by $\diagup$ hatching and local convexification tag by $\diagdown$ hatching. Candidate cells to check are double hatched.

3.3.2. *Laplacian/anti-laplacian smoothing*

An additional tool may help the convergence of the algorithm. The laplacian/anti-laplacian smoothing provides an efficient way to solve singular area.

Consider p^* the barycentric of neighbors points of p:

$$p^* = \frac{1}{|V(p)|} \sum_{q \in V(p)} .$$

Consider $\alpha > 0$, we define the *laplacian* α relaxed as:

$$p^{n+1} = p^n + \alpha \overrightarrow{p^n p^*}$$

The smoothing procedure using Laplacien/anti-laplacian is performed by applying a laplacien α relaxed followed by an other laplacian β relaxed with $\beta <_\sim \alpha$. Good results were obtained with $\beta = \alpha$. It should be noted that this step is clearly enhancing the mesh quality, but not necessary to the strict convexification (even nasty to a strict convexification due to the moving points induced). An additional projection step has to be done at each iteration for preserving the coherence of the sought bijection.

4. 3D CALCULATIONS

This section shows some 3D calculations of convexification. Figure 14 shows the first step of the convexification algorithm. Complex nodes due to the double saddle are removed, in first, since they are not constrained (these nodes are locally strictly concave).

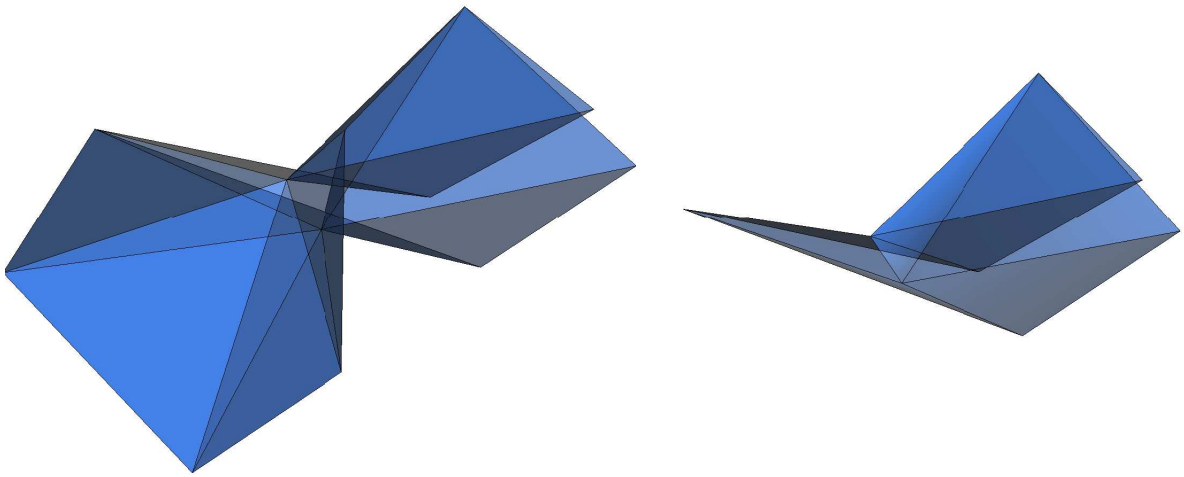

FIGURE 14. Steps of convexification of the two-saddle points mesh (*cf.* figure 7)

Figures 15 and 16 show some steps of the convexification process. These surfaces are not representative of surface faults but allow to test many features of the algorithms. These results are given by the 3D algorithm (section 3.2.3) without any geometry preservation algorithm (section 3.3.1) or smoothing (section 3.3.2). These

convexifications require 40 and 49 (resp.) iterations of the 3D algorithm. As specified above, the boundary of the meshes is preserved.

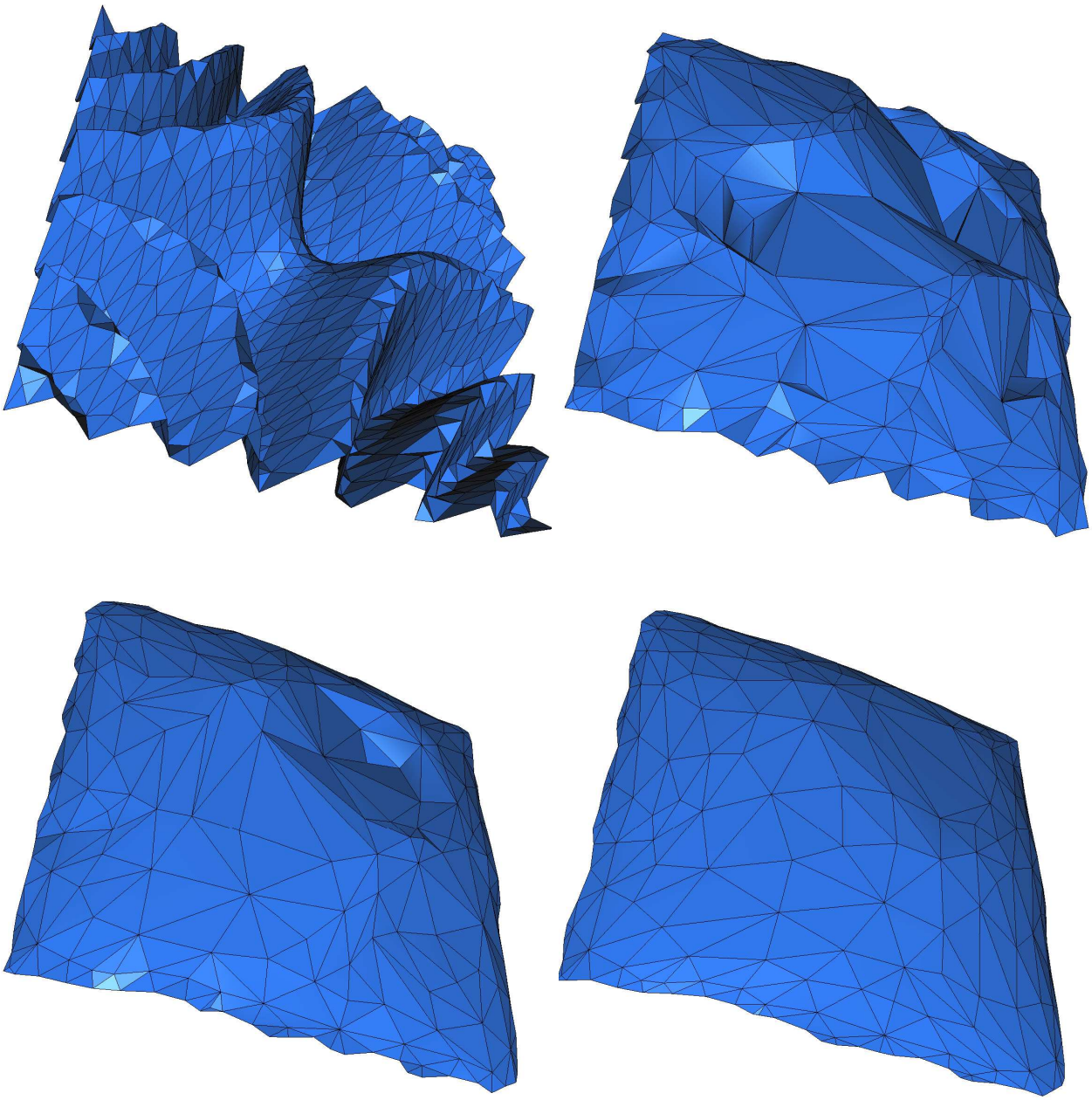

FIGURE 15. Steps of convexification of wave mesh

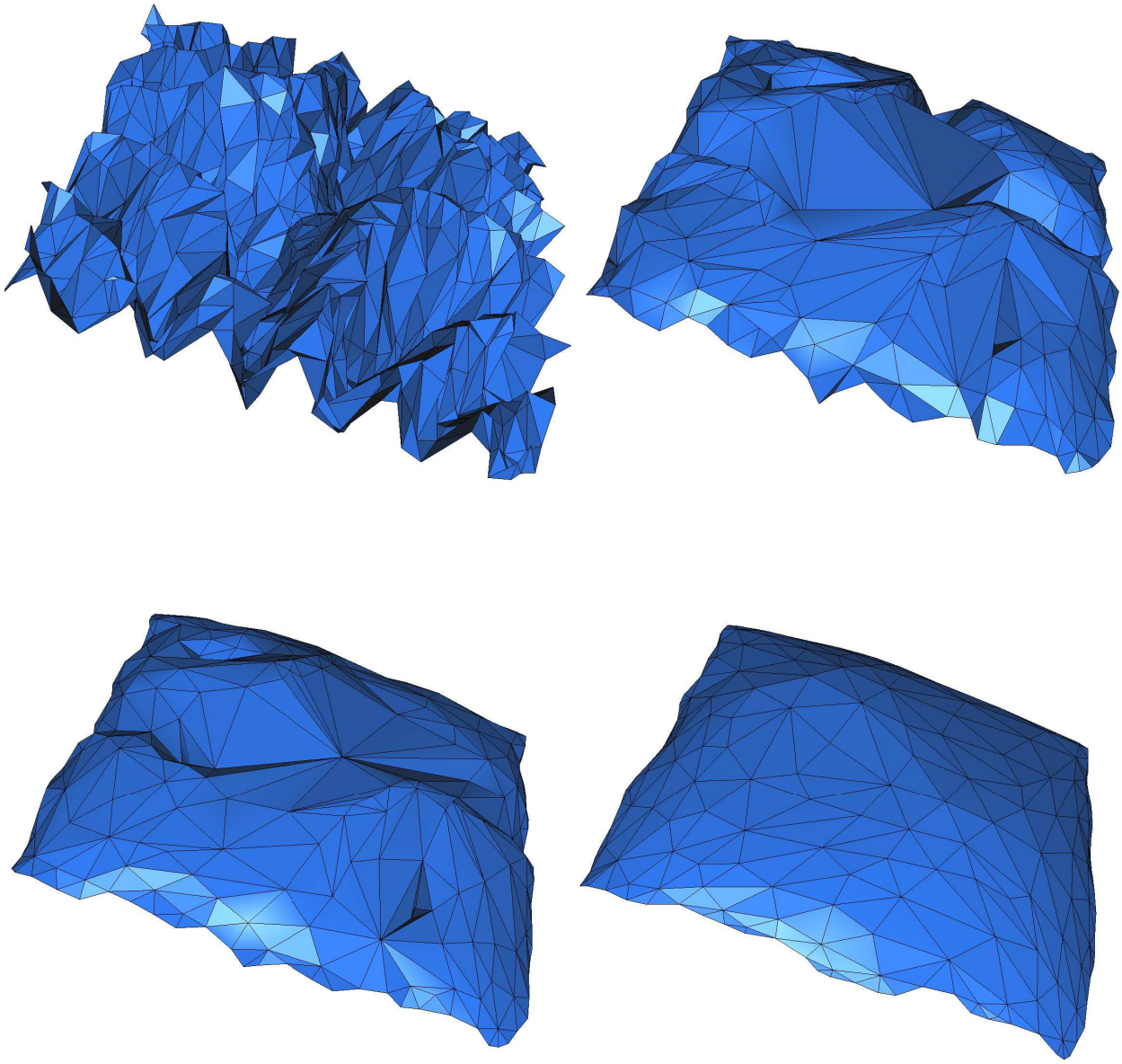

FIGURE 16. Steps of convexification of noisy wave mesh

ANNEX: VISIBILITY CONE

Let $\mathscr{N}(\vec{n}, \theta)$ be the visibility cone inscribed into the polyedral sector from the vertex p, of axis $\vec{n}$ and angle θ.

Consider a pair adjacent non-coplanar faces denoted by their base formed by the edges from p : $(\vec{u}, \vec{v})$ and $(\vec{u}, \vec{w})$ with $\vec{u}$, $\vec{v}$, $\vec{w}$(In the rare case of two coplanar adjacent faces, these latter are treated as a simple face).

Let P be the change of basis $P_{(\vec{u},\vec{v},\vec{w})\to\mathbb{R}^3}(\vec{n}) = \{\vec{u}, \vec{v}, \vec{w}\}$.

Let $\tilde{n} = P^{-1}(\vec{n})$ be the image (projection) of $\vec{n}$ in the basis $\mathcal{B} = (\vec{u}, \vec{v}, \vec{w})$. In what follows, $\vec{.}$ will denote the vector expressed in the canonical basis, while $\tilde{.}$ will denote the vector expressed in the base $\mathcal{B}$.

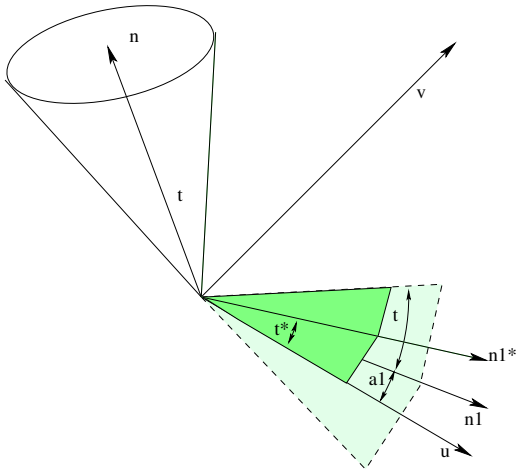

FIGURE 17. Projection process of $\mathscr{N}(\vec{n}, \theta)$

Let $\tilde{n}_1 = P_{(\vec{u},\vec{v},\vec{w})\to(\vec{u},\vec{v})}(\tilde{n})$ and $\tilde{n}_2 = P_{(\vec{u},\vec{v},\vec{w})\to(\vec{u},\vec{w})}(\tilde{n})$ be the projections of $\tilde{n}$ on the planes formed by $(\vec{u}, \vec{v})$ and $(\vec{u}, \vec{w})$ represented in the basis $\mathcal{B}$.

The scalar product in $\mathcal{B}$ is defined by the matrix $P^t.P$, it means that $\langle \tilde{x}, \tilde{y} \rangle_{\mathcal{B}} = \tilde{x}^t P^t P \tilde{y}$. In the same way, the norm in $\mathcal{B}$ is then written $||\tilde{x}||^2_{\mathcal{B}} = \langle \tilde{x}, \tilde{x} \rangle_{\mathcal{B}}$.

We denote $\tilde{u}$ the description of $\vec{u}$ on $\mathcal{B}$, which means that $P\tilde{u} = \vec{u}$. $\tilde{v}$ and $\tilde{w}$ are defined similarly.

Under these notations, we must verify that $\vec{n}$ is projected on the oriented faces $(\vec{u}, \vec{v})$ and $(\vec{u}, \vec{w})$. In other words: $\langle \tilde{n}_1, \tilde{u} \rangle_{\mathcal{B}} 0$, $\langle \tilde{n}_1, \tilde{v} \rangle_{\mathcal{B}} > 0$, $\langle \tilde{n}_2, \tilde{u} \rangle_{\mathcal{B}} > 0$ and $\langle \tilde{n}_2, \tilde{w} \rangle_{\mathcal{B}} > 0$.

By considering an iterative process of $\vec{n}$ construction where this one is projected on $\mathcal{B}$, then corrected to give $\vec{n}^* = P(\tilde{n}^*)$ with new $\tilde{n}^*_1 = P_{(\vec{u},\vec{v},\vec{w})\to(\vec{u},\vec{v})}(\tilde{n}^*)$ and $\tilde{n}^*_2 = P_{(\vec{u},\vec{v},\vec{w})\to(\vec{u},\vec{w})}(\tilde{n}^*)$. If $\tilde{n}^*_1$ and $\tilde{n}^*_2$ are known, the estimation of $\tilde{n}^*$ will be immediate.

First, in $(\vec{u}, \vec{v})$, the permissible angular opening is defined by the interval $I_1 = [\max(0, \alpha_{un_1} - \theta), \min(\alpha_{uv}, \alpha_{un_1} + \theta)]$. In $(\vec{u}, \vec{w})$, this one is described by $I_2 = [\max(0, \alpha_{un_2} - \theta), \min(\alpha_{uw}, \alpha_{un_2} + \theta)]$. To synthesize, the permissible angular opening in terms of these two directions is $I = I_1 \cap I_2$. Thus, we deduce the new opening angle of the cone $\mathscr{N}(\vec{n}^*, \theta^*)$ verify $2\theta^* \leq |I|$.

$\tilde{n}^*_1$ and $\tilde{n}^*_2$ are determined as the bisecting line of I according to planes $(\vec{u}, \vec{v})$ and $(\vec{u}, \vec{w})$ respectively. In other words, the angle α between $\tilde{n}^*_1$ and $\tilde{u}$ (and similarly between $\tilde{n}^*_2$ and $\tilde{u}$) is equal to the average of the interval I.

To determine $\tilde{n}^*_1$ we can use the relations $\tilde{n}^*_1 \in Span(\tilde{u}, \tilde{v})$, $\tilde{u} \cdot \tilde{n}^*_1 = ||\tilde{u}|| \cdot ||\tilde{n}^*_1|| \cos(\alpha)$, $\tilde{v} \cdot \tilde{n}^*_1 = ||\tilde{v}|| \cdot ||\tilde{n}^*_1|| \cos(\alpha_{uv} - \alpha)$ once more $\tilde{n}^*_1 = P(\tilde{n}^*_1)$. Similarly $\tilde{n}^*_2$ in the basis $(\tilde{u}, \tilde{w})$ (these relations can also be written in $\mathcal{B}$ only by using the appropriate norms and scalar products).

Thus, by seeking $\tilde{n}^*_1$ under the constraint $||\tilde{n}^*_1||_{\mathcal{B}} = 1$ and with (n^u_1, n^v_1) the coefficients of $\tilde{n}^*_1$ in the base $(\tilde{u}, \tilde{v})$, *i.e.*$\tilde{n}^*_1 = n^u_1 \tilde{u} + n^v_1 \tilde{v}$, we have

$$\begin{pmatrix} ||\tilde{u}||^2_{\mathcal{B}} & \langle \tilde{u}, \tilde{v} \rangle_{\mathcal{B}} \\ \langle \tilde{u}, \tilde{v} \rangle_{\mathcal{B}} & ||\tilde{v}||^2_{\mathcal{B}} \end{pmatrix} \begin{pmatrix} n^u_1 \\ n^v_1 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) \\ \cos(\alpha_{uv} - \alpha) \end{pmatrix}$$

and likewise for $\tilde{n}^*_2 = n^u_2 \tilde{u} + n^w_2 \tilde{w}$ we have

$$\begin{pmatrix} ||\tilde{u}||^2_{\mathcal{B}} & \langle \tilde{u}, \tilde{w} \rangle_{\mathcal{B}} \\ \langle \tilde{u}, \tilde{w} \rangle_{\mathcal{B}} & ||\tilde{w}||^2_{\mathcal{B}} \end{pmatrix} \begin{pmatrix} n^u_2 \\ n^w_2 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) \\ \cos(\alpha_{uw} - \alpha) \end{pmatrix}$$

Remark . *All the coefficients of the previous matrices are obtained by calculating $P^t \cdot P$.*

Therefore, $\tilde{n}^* = n^u\tilde{u} + n^v\tilde{v} + n^w\tilde{w}$ is simply determined by defining $\tilde{n}_1^*$ and $\tilde{n}_2^*$ in a common scale (this is due to the constraints on the projection of $\vec{n}$, $n_1^u > 0$ and $n_2^u > 0$). It means that, $\tilde{n} = (1, n_1^v/n_1^u, n_2^w/n_2^u)$. $\vec{n}^*$ is immediately obtained by $\vec{n}^* = P(\tilde{n}^*)$.

References

[1] A. Dervieux and F. Thomasset, Multifluid incompressible flows by a finite element method, Lecture Notes in Physics, 11 (1981), 158-163.

[2] M. S. Floater, Parametrization and smooth approximation of surface triangulations, Comp. Aided Geom. Design 14 (1997), 231–250.

[3] P. Frey, About Surface Remeshing, Proceedings, 9th International Meshing Roundtable, Sandia National Laboratories, pp.123-136, October 2000.

[4] S. Goossens, X.-C. Cai, Mortar Projection in Overlapping Composite Mesh Difference Methods, 2001.

[5] X. Jiao and M. T. Heath, Overlaying surface meshes, part 1: Algorithms, Int. Jour. Comp. Geom. and Appl., Vol. 14, 6 (2004), 379-402.

[6] X. Jiao and M. T. Heath, Overlaying surface meshes, part 2: Topology preservation and feature matching, Int. Jour. Comp. Geom. and Appl., Vol. 14, 6 (2004), 403-419.

[7] M. Meyer, M. Desbrun, P. Schroder and A. H. Barr, Discrete Differential Geometry Operators for Triangulated 2-Manifolds, 2002. VisMath, Vol. '02 (2002), pp. 35-57.

[8] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed : Algorithms based on Hamilton-Jacobi formulations, J. Comp. Phys., 79 (1988), 12-49.

[9] E. Praun, W. Sweldens, P. Schröder, Consistent Mesh Parameterizations, SIGGRAPH 2001, Computer Graphics Proceedings, ACM Press / ACM SIGGRAPH (2001), 179–184

[10] X. Roca, J. Sarrate and A. Huerta, Surface Mesh Projection for Hexahedral Mesh Generation by Sweeping, Proceedings, 13th International Meshing Roundtable, Williamsburg, VA, Sandia National Laboratories, SAND #2004-3765C, pp.169-180, September 19-22 2004.

[11] A. Sheffer, E. Praun, K. Rose, Mesh Parameterization Methods and Their Applications, Now Publishers, 2006, ISBN 978-1-933019-43-7, (also Foundations and Trends in Computer Graphics and Vision, volume 2(2),2006)

[12] G.-l. Xu, C. L. Bajaj, Curvature Computations of 2-manifolds in IR$\hat{\text{k}}$, Journal Of Computational Mathematics -International Edition- 2003, VOL 21; PART 5, pages 681-688