Anisotropic and feature sensitive triangular remeshing using normal lifting

Vincent Nivoliers, Bruno Lévy, Christophe Geuzaine

Anisotropic and Feature Sensitive Triangular Remeshing using Normal Lifting

Vincent Nivoliers[a,b], Bruno Lévy[c], Christophe Geuzaine[b]

[a] *Université de Lyon, LIRIS, R3AM group*
[b] *Université de Liège, Montefiore institute, ACE group*
[c] *Loria / inria Nancy Grand Est, Alice project team*

Abstract

This work describes an automatic method to anisotropically remesh an input bad quality mesh while preserving sharp features. We extend the method of Lévy and Bonneel [17], based on the lifting of the input mesh in a 6D space (position and normal), and the optimization of a restricted Voronoï diagram in that space. The main advantage of this method is that it does not require any parameterization of the input geometry: the remeshing is performed globally, and triangles can overlap several input charts. We improve this work by modifying the objective function minimized in the optimization process, in order to take into account sharp features. This new formulation is a generalization of the work of Lévy and Liu [18], which does not require any explicit tagging of the sharp features. We provide efficient formulas to compute the gradient of our objective function, thus allowing us to use a quasi-Newton solver [19] to perform the minimization.

Keywords: geometry, mesh generation, restricted Voronoï diagram
PACS: 02.40.Sf, 02.60.Pn, 02.70.Dh
2000 MSC: 65L50

1. Introduction

Mesh generation is a crucial step in order to perform numerical simulations on CAD objects. Our work focuses on *surface* mesh generation, sometimes also denoted as *boundary recovery*. Many techniques have already been developed for this task. Our method takes as input a 3D object provided as a bad quality mesh (like many stereolithography, or STL, meshes), and generates a new mesh with nice triangles in terms of aspect ratio. For CAD objects defined in other formats, fully automatic algorithms exist to generate such a mesh, since these are internally used for the visualization and rendering of the object. In terms of quality, most STL meshes are not suitable for numerical simulations, since their triangles may have very bad shapes. In addition, gaps and T-junctions may occur, especially when adjacent spline patches in the CAD geometry were not discretized accordingly. Our method is an extension of the work of Lévy and Bonneel [17], based on the optimization of restricted Voronoï diagrams. It can be considered as a generalization of Lloyd relaxation [20], to generate a regular sampling of an input mesh. Lévy and Bonneel [17] designed a method capable of automatically generating anisotropic elements in curved regions of the mesh. Our contribution with respect to their algorithm adds several features :

- automatic detection and preservation of sharp features;
- automatic isotropic scaling of the elements in curved regions;
- handling of gaps and T-junctions in the input mesh with a robust normal computation;
- new efficient formulation of the gradient of the objective function.

Email addresses: vincent.nivoliers@ulg.ac.be (Vincent Nivoliers), bruno.levy@inria.fr (Bruno Lévy), cgeuzaine@ulg.ac.be (Christophe Geuzaine)

As compared to other state of the art methods, we emphasize that we do not use any preprocessing on the input mesh, and do not require any parameterization of the object. Our method can therefore be applied in cases when other fast and reliable meshing algorithms are not able to handle the input. We however do not prove guarantees on the quality of the elements upon termination.

A full summary of mesh generation techniques is outside the scope of this paper, and several overviews already exist on the field [4, 11]. We here review several techniques related to our approach.

1.1. Advancing front algorithms

Advancing front algorithms are very fast and reliable to generate surface and volume meshes. For surface remeshing, the method requires a parameterization of the surface in one or several user defined patches. The boundary of the patches is first sampled, and from this boundary, nicely shaped triangles are generated one by one towards the interior of the patch, given a prescribed edge length. When the meshed region starts to overlap itself, a final procedure is triggered to sew the front curve and close the mesh. Recent advances consider an additional metric field in the parametric space to define how the edge lengths of the generated triangles are computed [27, 16], and the *gradation* control, avoiding abrupt transitions between large and small elements [1].

These methods are very efficient, but require the input object to be split into parametric patches. This is sometimes not a limitation since CAD objects are often defined as a set of parametric spline patches. However the mesh is generated per patch, which can be problematic when many patches are use to describe the shape of the object in complicated regions. Marcum [22] propose a volumetric method to merge multiple patches with a unique parameterization while Attene et al. [3] define an advancing front method robust to the quality of the parameterization. Marchandise et al. [21] especially target the remeshing of STL meshes, and study various automatic parameterization techniques. While automatic parameterization methods exist, the topology of the input mesh needs to be properly defined, and gaps cannot be handled. In addition, the parameterization is stored and thus sampled on the vertices of a background mesh. While this is not a problem for dense STL meshes obtained from 3D scanning devices or extracted from voxel data, the STL triangulation automatically generated from CAD data is usually too bad for automatic parameterization.

1.2. Mesh adaptation

An other family of remeshing methods is based on the progressive modification of a bad input mesh. These methods are therefore directly applicable to STL meshes, but require a preprocessing of the input mesh to be able to handle gaps and connectivity problems. The input mesh is generally modified via edge splitting [5], collapsing or swapping, and vertex relocation [28]. Once all the triangles respect some desired quality criterion, a final smoothing step is applied, relocating the vertices. Rassineux et al. [25] study various configurations of edge splitting to simultaneously adapt the surface triangular mesh and the volume tetrahedral mesh. With respect to these methods, our approach replaces the introduction of new mesh vertices using edge splits by a random uniform sampling of the mesh, and our objective function minimization behaves similarly to the final vertex smoothing step.

1.3. Delaunay refinement and optimization

Delaunay refinement techniques are closer to our approach, since the connectivity of the final mesh is obtained via a Delaunay triangulation and is therefore automatically handled through the vertex insertion phase. Borouchaki et al. [7] combine the Delaunay triangulation with an advancing front method for vertex insertion given a certain metric. The connectivity of the mesh is therefore automatically updated during the algorithm. Dey and Ray [9] follow the approach of mesh adaptation by inserting vertices one by one in an existing mesh, updating the connectivity using a restricted Delaunay triangulation. The authors prove guarantees on the quality of the output mesh. Both methods however still require a correct input connectivity to be applied to be able to either parameterize the object, or recover its topology properly.

Chen et al. [8] use the concept of *optimal Delaunay triangulation* to generate volume meshes, which is somehow dual to our approach. An objective function is defined over the restricted Delaunay triangulation of a set of samples, measuring the quality of the output elements. The optimization of the objective function leads to nicely shaped elements. Although the input requirements are as weak as ours for this method, the continuity of the objective function is only C^0, and the optimization procedure requires the use of simulated annealing, whereas our objective function is almost C^2 and can be optimized with a quasi-Newton solver. The objective function however is directly defined on the shape of the output triangles, and can be directly related to the final element quality.

2. Background

We here detail the basic tools we rely on. Our method is based on the minimization of an objective function defined on a restricted Voronoï diagram. From a random initial set of samples, this minimization produces a regular sampling of the input mesh. Using the restricted Delaunay triangulation of this sampling, we then generate a triangulation automatically. Algorithm 1 outlines this pipeline : from an initial random sampling, the restricted Voronoï diagram of the samples in computed, to extract the objective function value and its gradient. These are provided to the solver, which generates a new set of samples. Each time the samples change, the restricted Voronoï diagram has to be recomputed. This process is then iterated until the solver reaches convergence, or the desired number of iterations is reached.

input : $\mathcal{S}$, a bad triangular mesh
output: $\mathcal{T}$, a mesh with good quality triangles
algorithm
  $\mathbf{V} \leftarrow$ random site initialization on $\mathcal{S}$
  while *a minimum is not reached* **do**
    Compute the restricted Voronoï diagram of $\mathbf{V}$ and $\mathcal{S}$
    Compute the objective function $\mathscr{F}$
    Compute the gradient $\frac{\mathrm{d}\mathscr{F}}{\mathrm{d}\mathbf{V}}$ of $\mathscr{F}$
    $\mathbf{V} \leftarrow$ lbfgs(V,$\mathscr{F}$, $\frac{\mathrm{d}\mathscr{F}}{\mathrm{d}V}$) // Liu and Nocedal [19]
  $\mathcal{T} \leftarrow$ restricted Delaunay triangulation of $\mathbf{V}$ and $\mathcal{S}$
  return $\mathcal{T}$

Algorithm 1: Outline of a centroïdal Voronoï tesselation based remeshing method.

This algorithm is a generalization of Lloyd's algorithm [20] and was introduced by Yan et al. [29]. Lévy and Liu [18] then modified it for applications in quadrangular and hexahedral remeshing and to introduce feature sensitiveness. Lévy and Bonneel [17] finally propose an other variation for anisotropic remeshing. We here summarize these methods on which our work is based.

2.1. Meshes and sites in d dimensional space

Throughout this article, we will deal with samples and meshes both in $\mathbb{R}^3$ and $\mathbb{R}^6$. Whatever the dimension, the meshes are always triangular meshes. A mesh $\mathcal{S}$ can therefore always be defined by a set of vertices $\mathbf{X} = \{\mathbf{x}_i\}_i \subset \mathbb{R}^d$ and a set of triangles $\mathbf{T} = \{T_p\}_p \subset \mathbf{X}^3$. Given a triangle $T_p = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, whatever the dimension, the surface of the triangle can always be defined as

$$\left\{\mathbf{x} \in \mathbb{R}^d, \mathbf{x} = u\mathbf{x}_1 + v\mathbf{x}_2 + w\mathbf{x}_3, \{u, v, w\} \subset \mathbb{R}^+, u + v + w = 1\right\}. \tag{1}$$

As an abuse of notation, we will use T_p both to refer to the triplet of vertices in $\mathbf{X}$ defining the triangle and the surface of that triangle defined by Equation 1.

2.2. Restricted Voronoï Diagrams

2.2.1. Definition

Given a set of sites $\mathbf{V} = \{\mathbf{v}_k\}_k \subset \mathbb{R}^d$ and a triangular mesh $\mathcal{S}$ embedded in $\mathbb{R}^d$, a restricted Voronoï diagram associates with each site $\mathbf{v}_k$ a (possibly empty) piece of $\mathcal{S}$. The Voronoï cell of $\mathbf{v}_k$ is defined as

$$\Omega_k = \left\{\mathbf{p} \in \mathbb{R}^d, \|\mathbf{p} - \mathbf{v}_k\| \leq \|\mathbf{p} - \mathbf{v}_\ell\|, \mathbf{v}_\ell \neq \mathbf{v}_k \in \mathbf{V}\right\}. \tag{2}$$

From this definition, the *restricted* Voronoï cell $\Omega_{k|\mathcal{S}}$ of $\mathbf{v}_k$ is the intersection between the classical Voronoï cell Ω_k and the surface $\mathcal{S}$, defined as

$$\Omega_{k|\mathcal{S}} = \{\mathbf{x} \in \mathcal{S}, \|\mathbf{x} - \mathbf{v}_k\| \leq \|\mathbf{x} - \mathbf{v}_\ell\|, \mathbf{v}_\ell \neq \mathbf{v}_k \in \mathbf{V}\}. \tag{3}$$

(a) restricted Voronoï diagram

(b) restricted Delaunay triangulation

Figure 1: Restricted Voronoï diagram and Delaunay triangulation, with $\mathcal{S}$ being a sphere in $\mathbb{R}^3$.

In other words, each point $\mathbf{x} \in \Omega_{k|\mathcal{S}}$ belongs to $\mathcal{S}$ and the nearest site from $\mathbf{x}$ is $\mathbf{v}_k$.

The set $\{\Omega_{k|\mathcal{S}}\}_k$ of restricted Voronoï cells forms a partition of $\mathcal{S}$: each point of $\mathcal{S}$ belongs to a restricted Voronoï cell. The converse is however not true: a site located too far from $\mathcal{S}$ may have an empty restricted Voronoï cell. Figure 1 shows an example of a restricted Voronoï diagram.

2.2.2. Computation

Several algorithms were designed to compute a restricted Voronoï diagram. We use the approach of Lévy and Bonneel [17], which handles meshes embedded in $\mathbb{R}^d$. This method replaces the traditional Voronoï diagram computation by nearest neighbor queries [23], which can be performed efficiently in any dimension. In terms of complexity, although worst case scenarios exist, the experienced complexity we encountered in our work for this computation is $O(n + m)$, where n is the number of sites and m the number of triangles in $\mathcal{S}$. This is mainly due to the fact that the sets of sites we consider are located on $\mathcal{S}$ or very close to it, and optimized to form a regular sampling of $\mathcal{S}$.

2.2.3. Restricted Delaunay triangulation

From the restricted Voronoï diagram, a triangulation of the surface can be derived. This triangulation is our output mesh $\mathcal{T}$ and is the *dual* of the diagram: each site of the diagram is a vertex of $\mathcal{T}$. The triangles $\mathbf{D} \subset \mathbf{V}^3$ of $\mathcal{T}$ are defined as

$$\mathbf{D} = \left\{(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) \in \mathbf{V}^3, \Omega_{1|\mathcal{S}} \cap \Omega_{2|\mathcal{S}} \cap \Omega_{3|\mathcal{S}} \neq \emptyset\right\}. \tag{4}$$

In other words, each time three restricted Voronoï cells meet at one point, a triangle connects the three corresponding sites. These triangles are the restricted Delaunay triangles of $\mathbf{V}$ with respect to $\mathcal{S}$, and the corresponding triangulation is the restricted Delaunay triangulation, used in Algorithm 1. Figure 1 provides an example of a restricted Delaunay triangulation.

Edelsbrunner and Shah [10] provide conditions on the set of sites $\mathbf{V}$ with respect to $\mathcal{S}$ to ensure that $\mathcal{T}$ is a valid remeshing of $\mathcal{S}$, or more formally that $\mathcal{T}$ is homeomorphic to $\mathcal{S}$. Problems may happen if the number of samples is insufficient, in regions where $\mathcal{S}$ is thin for instance.

2.3. Objective function

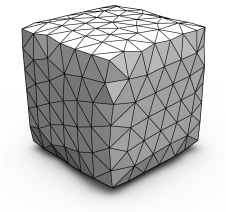
2.3.1. Definition

The objective function measures the quality of the sampling of $\mathcal{S}$ by the set of samples $\mathbf{V}$. The underlying idea is that the sampling is good if every point $\mathbf{x} \in \mathcal{S}$ is close to a site $\mathbf{v}_k \in \mathbf{V}$. Formulating this idea in a least-squares fashion, we obtain

$$\mathscr{F}(\mathbf{V}) = \int_{\mathcal{S}} \min_{\mathbf{v}_k \in \mathbf{V}} \|\mathbf{x} - \mathbf{v}_k\|^2 \mathrm{d}\mathbf{x}.$$

Using the restricted Voronoï diagram of $\mathbf{V}$ and $\mathcal{S}$, this integral can be split into

$$\mathscr{F}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \int_{\Omega_{k|\mathcal{S}}} \|\mathbf{x} - \mathbf{v}_k\|^2 \mathrm{d}\mathbf{x}, \tag{5}$$



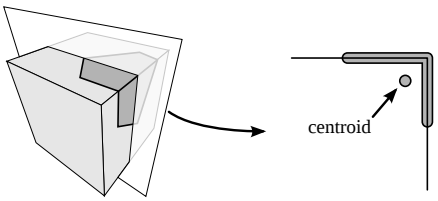


Figure 2: Remeshing a cube by the minimization of Equation 5. The sharp features are smoothed. This behavior is due to the fact that on sharp features the centroid of restricted Voronoï cells is not located on the surface.

which is the basic formulation for all the objective functions variations we will define. This objective function was proved to be C^2 almost everywhere in the case of restricted Voronoï diagrams [24], and is therefore suitable for a quasi-Newton minimization procedure like LBFGS [19]. As a comparison, Lloyd's algorithm can be seen as a steepest descent algorithm for the minimization of the same objective function. The result is a *centroïdal* restricted Voronoï diagram, where every site $\mathbf{v}_k$ is located at the centroïd of its restricted Voronoï cell $\Omega_{k|\mathcal{S}}$. The diagram depicted on Figure 1 is centroïdal.

Triangulating each restricted Voronoï cell, the objective function sums polynomial integrals over triangles. Its value is computed using the formulas of Lasserre and Avrachenkov [15] as

$$\mathscr{F}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \sum_{T \in \Omega k|\mathcal{S}} \frac{|T|}{6} \sum_{(\mathbf{c}_i, \mathbf{c}_j) \in T^2} (\mathbf{v}_k - \mathbf{c}_i).(\mathbf{v}_k - \mathbf{c}_j), \tag{6}$$

where $\mathbf{c}_i$ and $\mathbf{c}_j$ are (potentially identical) vertices of the triangle T in a triangulation of $\Omega_{k|\mathcal{S}}$.

2.3.2. Gradient

The derivatives of this objective function with respect to the set of sites are provided by Iri et al. [14], and a more detailed explanation of their formulas is given by Asami [2]. The derivative of $\mathscr{F}$ with respect to a site $\mathbf{v}_k \in \mathbf{V}$ is given by

$$\frac{\mathrm{d}\mathscr{F}}{\mathrm{d}\mathbf{v}_k}(\mathbf{V}) = 2|\Omega_{k|\mathcal{S}}|(\mathbf{v}_k - \mathbf{g}_{k|\mathcal{S}})^t, \tag{7}$$

where $|\Omega_{k|\mathcal{S}}|$ is the area of the restricted Voronoï cell and $\mathbf{g}_{k|\mathcal{S}}$ is its centroid. The gradient is the transpose of this derivative. This equation shows that Lloyd's algorithm, by moving sites towards the centroid of their restricted cell, is a steepest descent algorithm.

2.4. Feature sensitiveness

2.4.1. Objective function

Minimizing the basic objective function as defined by Equation 5 works well for the remeshing of smooth surfaces. For surfaces with sharp features however, the result is smoothed. This behavior is due to the fact that the sites end up at the centroid of their restricted cell, which is not located on the surfaces at sharp features (Figure 2).

When $\mathcal{S}$ is in $\mathbb{R}^3$, Lévy and Liu [18] propose a modification of the objective function to address this problem. Their idea is to bring back the sites on the surface, by increasing the importance of the normal component in the distance between a site and a point $\mathbf{x}$ on the surface. Let $\mathbf{n}_\mathbf{x}$ be the normal of $\mathcal{S}$ at $\mathbf{x} \in \mathcal{S}$ and $\mathbf{v}_k \in \mathbf{V}$, the normal component of $\mathbf{x} - \mathbf{v}_k$ at $\mathbf{x}$ is given by

$$[(\mathbf{x} - \mathbf{v}_k).\mathbf{n}_\mathbf{x}]\mathbf{n}_\mathbf{x} = \mathbf{n}_\mathbf{x}\mathbf{n}_\mathbf{x}^t(\mathbf{x} - \mathbf{v}_k). \tag{8}$$

From this, Lévy and Liu [18] modify the distance used in Equation 5 as

$$\|\mathbf{x} - \mathbf{v}_k\|_\mathbf{x} = \|M_\mathbf{x}(\sigma)(\mathbf{x} - \mathbf{v}_k)\|, \text{ with } M_\mathbf{x}(\sigma) = (\sigma - 1)\mathbf{n}_\mathbf{x}\mathbf{n}_\mathbf{x}^t + I_3, \tag{9}$$

where σ is a parameter corresponding to the additional weighting of the normal component and I_3 is the identity matrix in $\mathbb{R}^3$. Inserting this into Equation 5, and noticing that the normal $\mathbf{n}_x$ is constant over each triangle $T_p \in \mathbf{T}$ of

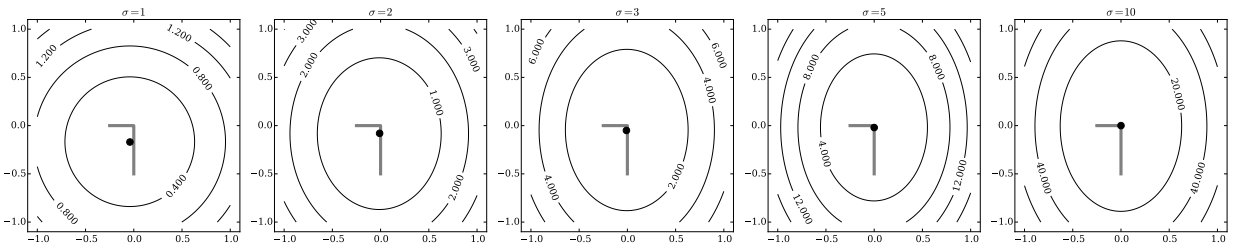


Figure 3: Isovalues of the feature sensitive objective function given by Equation 10 for different values of the feature sensitivity σ. The fat gray lines in the middle correspond to a simple 1D mesh embedded in 2D. The objective function is computed for one single site. The black dot corresponds to the optimal position of the site. With $\sigma = 1$, no feature sensitivity is used, the optimal site is far from the corner, which is the feature we are willing to preserve. Increasing σ moves the optimal site position towards the corner.

$\mathcal{S}$, we obtain

$$\mathscr{F}(\mathbf{V}, \sigma) = \sum_{\mathbf{v}_k \in \mathbf{V}} \sum_{T_p \in \mathbf{T}} \int_{\Omega_{k|T_p}} \left\| M_{T_p}(\sigma)(\mathbf{x} - \mathbf{v}_k) \right\|^2 \mathrm{d}\mathbf{x}, \tag{10}$$

where $\Omega_{k|T_p} = \Omega_k \cap T_p$. Algorithmically, these intersections are obtained during the restricted Voronoï diagram computation. The effect of this modification of the objective function is shown on Figure 3.

2.4.2. Value

Using a similar development as for Equation 6, the value of this objective function is obtained as

$$\mathscr{F}(\mathbf{V}) = \sum_{\mathbf{v}_k \in \mathbf{V}} \sum_{T \in \Omega_{k|\mathcal{S}}} \frac{|T|}{6} \sum_{(\mathbf{c}_i, \mathbf{c}_j) \in T^2} (\mathbf{v}_k - \mathbf{c}_i)^t M_T^t(\sigma) M_T(\sigma)(\mathbf{v}_k - \mathbf{c}_j). \tag{11}$$

2.4.3. Gradient

With $\mathscr{F}$ as defined in Equation 10 the expression of its gradient gets more complicated. The derivation used to obtained Equation 7 uses the fact that given two neighboring sites $\mathbf{v}_k$ and $\mathbf{v}_\ell$, for $\mathbf{x} \in \Omega_{k|\mathcal{S}} \cap \Omega_{\ell|\mathcal{S}}$, $\|\mathbf{x} - \mathbf{v}_k\| = \|\mathbf{x} - \mathbf{v}_\ell\|$ since $\mathbf{x}$ is on the bisector of $\mathbf{v}_k$ and $\mathbf{v}_\ell$. This is no longer the case using the modified distance of Equation 9, since the normal components of the two vectors are usually different. Lévy and Liu [18] therefore derive the gradient from Equation 6, which requires in particular to compute the Jacobians of the vertices $\mathbf{c}_i$ and $\mathbf{c}_j$. We will not detail here their final equation since we provide in Section 3.2 a simpler formulation.

2.5. Anisotropy through normal lifting

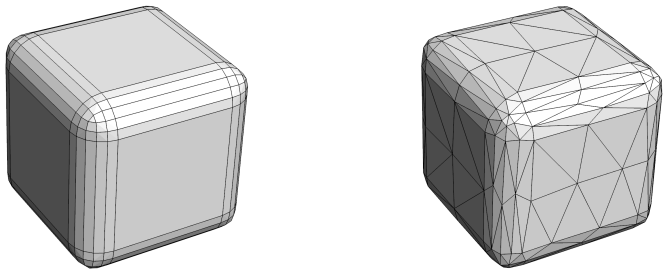
The restricted Delaunay triangulation of a centroïdal restricted Voronoï diagram is made of triangles nearly equilateral. While this is a requirement for many finite element problems, it requires that many elements be used to mesh the geometry in highly curved regions. When the problem allows it, it might therefore be desirable to trade a bit of the element quality to be able mesh curved regions with fewer elements. This is anisotropic remeshing. The goal is therefore to mesh curved regions of the mesh with elements that are thin in the main curvature direction. Lévy and Bonneel [17] achieve this goal using *normal lifting*.

Normal lifting consists in embedding the input mesh $\mathcal{S}$ in $\mathbb{R}^6$ by appending to each vertex $\mathbf{x}_i \in \mathbf{X}$ three new coordinates based on the average normal vector $\mathbf{n}_i$ at $\mathbf{x}_i$. Introducing a parameter ν to scale the lifting, the new vertices $\hat{\mathbf{X}}$ are defined as

$$\hat{\mathbf{X}} = \left\{ \hat{\mathbf{x}}_i \in \mathbb{R}^6,\ \hat{\mathbf{x}}_i = \mathbf{x}_i \oplus \nu \mathbf{n}_i,\ \mathbf{x}_i \in \mathbf{X} \right\}, \tag{12}$$

where $\oplus$ concatenates the coordinates of the two provided vectors. The remeshing is then performed on the lifted mesh $\hat{\mathcal{S}}$, producing a mesh $\hat{\mathcal{T}}$ in $\mathbb{R}^6$. This mesh is then projected orthogonally in $\mathbb{R}^3$ using the first three coordinates to obtain the final mesh $\mathcal{T}$.

By definition of the curvature, along the main curvature direction, the normal varies. This means that input triangles located in curved regions of $\mathcal{S}$ will get wider along the main curvature direction when they are lifted. Conversely, equilateral triangles on the lifted mesh, located in curved regions will get thin along the main curvature direction once projected. Hence the anisotropic triangles in the remeshing in curved regions shown on Figure 4.



(a) input mesh (b) remeshing using normal lifting

Figure 4: The input mesh is first lifted in $\mathbb{R}^6$, remeshed. Projecting back in $\mathbb{R}^3$ yields the result, exhibiting thin triangles in curved regions.

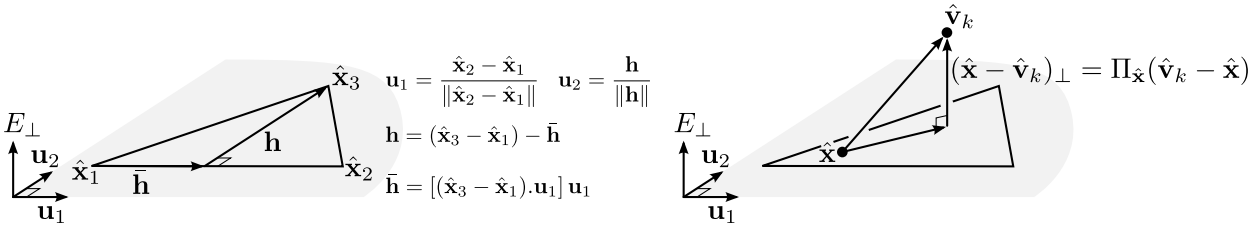


Figure 5: Computing the normal component of a vector $(\hat{\mathbf{v}}_k - \hat{\mathbf{x}})$. First an orthogonal basis $(\mathbf{u}_1, \mathbf{u}_2)$ of the triangle containing $\mathbf{x}$ is computed. The projection of $(\hat{\mathbf{v}}_k - \hat{\mathbf{x}})$ in the normal space $E_\perp$ is computed by removing from $(\hat{\mathbf{v}}_k - \hat{\mathbf{x}})$ its projection in the tangent space (Equation 13).

3. Generalizing feature sensitiveness

We now detail our first contribution, which allows users to benefit from both feature sensitivity and anisotropic remeshing. In its current definition in Section 2.4, feature sensitiveness is not compatible with normal lifting, since it uses the normal of a point $\mathbf{x} \in \mathcal{S}$. In this section, we reformulate Equation 10 to make it compatible with normal lifting. We also provide new formulas to compute efficiently the gradient of $\mathscr{F}$ with our formulation.

3.1. Using the tangent space

The problem of the formulation of feature sensitivity by Lévy and Liu [18] is that is relies on the normal of the surface. Using normal lifting, the normal space is no longer 1D, and the computation of the normal component of a vector has te be reformulated. To provide feature sensitivity when using normal lifting, our key idea is to use the *tangent* space at a point $\hat{\mathbf{x}} \in \hat{\mathcal{S}}$ rather than the normal space. Whatever the dimension, the tangent space remains 2D, which allows us to use the same formulation, whatever the lifting used. Given a surface $\mathcal{S}$ lifted as $\hat{\mathcal{S}} \subset \mathbb{R}^d$, and a point $\hat{\mathbf{x}} \in \hat{\mathcal{S}}$, we define a basis $(\mathbf{u}_{1,\hat{\mathbf{x}}}, \mathbf{u}_{2,\hat{\mathbf{x}}}) \in (\mathbb{R}^d)^2$ of the tangent space of $\hat{\mathcal{S}}$ at $\hat{\mathbf{x}}$. Given a site $\hat{\mathbf{v}}_k \in \hat{\mathbf{V}}$, the tangent component of $\hat{\mathbf{x}} - \hat{\mathbf{v}}_k$ is given by

$$(\hat{\mathbf{x}} - \hat{\mathbf{v}}_k).\mathbf{u}_{1,\hat{\mathbf{x}}}\mathbf{u}_{1,\hat{\mathbf{x}}} + (\hat{\mathbf{x}} - \hat{\mathbf{v}}_k).\mathbf{u}_{2,\hat{\mathbf{x}}}\mathbf{u}_{2,\hat{\mathbf{x}}} = (\mathbf{u}_{1,\hat{\mathbf{x}}}\mathbf{u}_{1,\hat{\mathbf{x}}}^t + \mathbf{u}_{2,\hat{\mathbf{x}}}\mathbf{u}_{2,\hat{\mathbf{x}}}^t)(\hat{\mathbf{x}} - \hat{\mathbf{v}}_k). \tag{13}$$

The normal component can now be expressed as $(I_d - \mathbf{u}_{1,\hat{\mathbf{x}}}\mathbf{u}_{1,\hat{\mathbf{x}}}^t - \mathbf{u}_{2,\hat{\mathbf{x}}}\mathbf{u}_{2,\hat{\mathbf{x}}}^t)(\hat{\mathbf{x}} - \hat{\mathbf{v}}_k)$, where I_d is the identity matrix in $\mathbb{R}^d$. Let $\Pi_{\hat{\mathbf{x}}} = (I_d - \mathbf{u}_{1,\hat{\mathbf{x}}}\mathbf{u}_{1,\hat{\mathbf{x}}}^t - \mathbf{u}_{2,\hat{\mathbf{x}}}\mathbf{u}_{2,\hat{\mathbf{x}}}^t)$. This matrix encodes the orthogonal projection of any vector onto the normal space of $\hat{\mathcal{S}}$ at $\hat{\mathbf{x}}$ (see Figure 5). We can now use this expression to provide a new formulation of the matrix $M_{\hat{\mathbf{x}}}(\sigma)$ defined in Equation 9, as

$$M_{\hat{\mathbf{x}}}(\sigma) = I_d + (\sigma - 1)\Pi_{\hat{\mathbf{x}}}. \tag{14}$$

This formulation applies in any dimension d, and can therefore be used in conjunction with normal lifting.

For each triangle $\hat{T}_p = (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \hat{\mathbf{x}}_3)$ of $\hat{\mathcal{S}}$, we compute an orthogonal basis (Figure 5) as

$$\mathbf{u}_{1,\hat{T}} = \frac{\hat{\mathbf{x}}_2 - \hat{\mathbf{x}}_1}{\|\hat{\mathbf{x}}_2 - \hat{\mathbf{x}}_1\|}, \quad \mathbf{u}_{2,\hat{T}} = \frac{(\hat{\mathbf{x}}_3 - \hat{\mathbf{x}}_1) - \mathbf{u}_{1,\hat{T}}\mathbf{u}_{1,\hat{T}}^t(\hat{\mathbf{x}}_3 - \hat{\mathbf{x}}_1)}{\left\|(\hat{\mathbf{x}}_3 - \hat{\mathbf{x}}_1) - \mathbf{u}_{1,\hat{T}}\mathbf{u}_{1,\hat{T}}^t(\hat{\mathbf{x}}_3 - \hat{\mathbf{x}}_1)\right\|}. \tag{15}$$

In the case when one of the denominators is null, the considered triangle is flat. Any integral on this triangle yields a null result, and this triangle can be ignored in the minimization process. The value of the objective function is computed using Equation 11, using the above orthogonal basis to define $M_{\hat{T}_p}(\sigma)$ on each lifted triangle $\hat{T}_p$.

3.2. Objective function gradient

We compute the gradient of $\mathscr{F}$ using the technique of Nivoliers and Lévy [24], with Reynolds' transport theorem. This theorem states that

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\int_{\Omega(t)} f(\mathbf{x},t)\mathrm{d}\mathbf{x}\right] = \underbrace{\int_{\Omega(t)} \frac{\partial f}{\partial t}(\mathbf{x},t)\mathrm{d}\mathbf{x}}_{\text{inner term}} + \underbrace{\int_{\partial\Omega(t)} f(\mathbf{x},t)\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}.\mathbf{n}_{\partial\Omega(t)}(\mathbf{x})\mathrm{d}\mathbf{x}}_{\text{boundary term}}, \tag{16}$$

where $\partial\Omega(t)$ is the boundary of $\Omega(t)$ and $\mathbf{n}_{\partial\Omega(t)}(\mathbf{x})$ is the outward normal of $\partial\Omega(t)$ at $\mathbf{x}$. In our case the variable t is a site $\hat{\mathbf{v}}_k$. At the simplest level, the domain Ω considered is the intersection $\hat{\Omega}_{k|\hat{T}_p}$ between the Voronoï cell of $\hat{\mathbf{v}}_k$ in the lifting space and a triangle $\hat{T}_p$ of the lifted input mesh $\hat{\mathcal{S}}$. This domain is a convex polygon, and varies with respect to $\hat{\mathbf{v}}_k$. We will now study the inner term and the boundary term separately.

3.2.1. Inner term

The inner term in Equation 16 considers that the domain is static. It simplifies nicely to provide an expression similar to Equation 7 :

$$\begin{aligned}\int_{\hat{\Omega}_{k|\hat{T}_p}} \frac{\partial}{\partial\hat{\mathbf{v}}_k}\left[\left\|M_{\hat{T}_p}(\sigma)(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k)\right\|^2\right]\mathrm{d}\hat{\mathbf{x}} &= \int_{\hat{\Omega}_{k|\hat{T}_p}} -2M^t_{\hat{T}_p}(\sigma)M_{\hat{T}_p}(\sigma)(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k)\mathrm{d}\hat{\mathbf{x}} \\ &= 2\left|\hat{\Omega}_{k|\hat{T}_p}\right|(\hat{\mathbf{v}}_k-\hat{\mathbf{g}}_{k,p})^t M^t_{\hat{T}_p}(\sigma)M_{\hat{T}_p}(\sigma),\end{aligned} \tag{17}$$

where $\hat{\mathbf{g}}_{k,p}$ is the centroid of $\hat{\Omega}_{k|\hat{T}_p}$ and $\left|\hat{\Omega}_{k|\hat{T}_p}\right|$ its area. In the case when $\sigma = 1$, this equation falls back to the result of Equation 7.

3.2.2. Boundary term

From the study of the boundary term by Nivoliers and Lévy [24], it turns out that the boundary of $\hat{\Omega}_{k|\hat{T}_p}$ is made out of two kinds of edges :

internal edges are pieces of the edges of $\hat{T}_p$. On these edges, the integral of Equation 16 is null, since the normal derivative of $\hat{\mathbf{x}}$ is null.

bisector edges are the intersection between $\hat{T}_p$ and the bisector of $\hat{\mathbf{v}}_k$ with some other site $\hat{\mathbf{v}}_\ell$. Let $\mathbf{n}_{k,\ell,p}$ be the normal of the edge in the plane of $\hat{T}_p$, oriented outside of $\hat{\Omega}_{k|\hat{T}_p}$. On these edges, the integral is not null, and the normal derivative of $\hat{\mathbf{x}}$ is given by

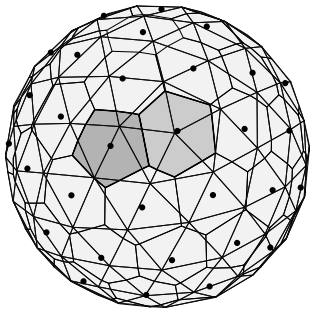
$$\frac{\mathrm{d}\hat{\mathbf{x}}}{\mathrm{d}\hat{\mathbf{v}}_k}.\mathbf{n}_{k,\ell,p} = \frac{(\hat{\mathbf{v}}_k-\hat{\mathbf{x}})^t}{\mathbf{n}_{k,\ell,p}.(\hat{\mathbf{v}}_k-\hat{\mathbf{v}}_\ell)}. \tag{18}$$

Given a segment $e_{k,\ell,p} = \partial\hat{\Omega}_{k|\hat{\mathcal{S}}} \cap \partial\hat{\Omega}_{\ell|\hat{\mathcal{S}}} \cap \hat{T}_p$, this segments is a bisector edge and yields two terms in the gradient of $\mathscr{F}$ with respect to $\hat{\mathbf{v}}_k$: one for each of the Voronoï cells containing $e_{k,\ell,p}$, since both cells vary with $\hat{\mathbf{v}}_k$. An example of such a segment is illustrated on Figure 6. Denoting $\frac{\mathrm{d}\mathscr{F}_{k,\ell,p}}{\mathrm{d}\hat{\mathbf{v}}_k}$ the sum of these two terms, we have

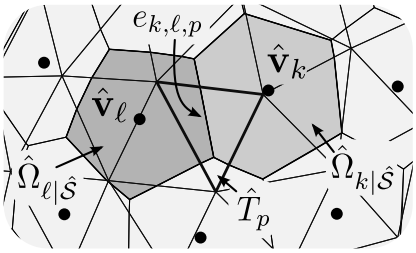
$$\frac{\mathrm{d}\mathscr{F}_{k,\ell,p}}{\mathrm{d}\hat{\mathbf{v}}_k} = \int_{e_{k,\ell,p}} \left(\left\|M_{\hat{T}_p}(\sigma)(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k)\right\|^2 - \left\|M_{\hat{T}_p}(\sigma)(\hat{\mathbf{x}}-\hat{\mathbf{v}}_\ell)\right\|^2\right)\frac{(\hat{\mathbf{v}}_k-\hat{\mathbf{x}})^t}{\mathbf{n}_{k,\ell,p}.(\hat{\mathbf{v}}_k-\hat{\mathbf{v}}_\ell)}\mathrm{d}\hat{\mathbf{x}}. \tag{19}$$

From Equation 14, we have

$$\left\|M_{\hat{T}_p}(\sigma)(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k)\right\|^2 = \left\|(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k) + (\sigma-1)\Pi_{\hat{T}_p}(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k)\right\|^2.$$



(a) two restricted Voronoï cells



(b) closeup on a segment $e_{k,\ell,p}$

Figure 6: A segment $e_{k,\ell,p} = \partial\hat{\Omega}_{k|\hat{\mathcal{S}}} \cap \partial\hat{\Omega}_{\ell|\hat{\mathcal{S}}} \cap \hat{T}_p$.

Since $\Pi_{\hat{T}_p}$ is an orthogonal projection matrix, we have $\Pi^t_{\hat{T}_p}\Pi_{\hat{T}_p} = \Pi_{\hat{T}_p}$. Using this relation we can develop further into

$$\left\|M_{\hat{T}_p}(\sigma)(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k)\right\|^2 = \|\hat{\mathbf{x}}-\hat{\mathbf{v}}_k\|^2 + (\sigma^2-1)(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k)^t\Pi^t_{\hat{T}_p}\Pi_{\hat{T}_p}(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k) = \|\hat{\mathbf{x}}-\hat{\mathbf{v}}_k\|^2 + (\sigma^2-1)\left\|\Pi_{\hat{T}_p}(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k)\right\|^2.$$

The vector $\Pi_{\hat{T}_p}(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k)$ is actually constant, whatever the point $\hat{\mathbf{x}}$. Indeed, if $\hat{\mathbf{c}}_1$ and $\hat{\mathbf{c}}_2$ are the endpoints of $e_{k,\ell,p}$, $\hat{\mathbf{x}}$ can be rewritten as $\hat{\mathbf{c}}_1 + \alpha_{\hat{\mathbf{x}}}(\hat{\mathbf{c}}_2-\hat{\mathbf{c}}_1)$. Since the vector $(\hat{\mathbf{c}}_2-\hat{\mathbf{c}}_1)$ is in the plane of $\hat{T}_p$, we have $\Pi_{\hat{T}_p}(\hat{\mathbf{c}}_2-\hat{\mathbf{c}}_1) = \mathbf{0}$, and thus $\Pi_{\hat{T}_p}(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k) = \Pi_{\hat{T}_p}(\hat{\mathbf{c}}_1-\hat{\mathbf{v}}_k)$. We finally obtain

$$\left\|M_{\hat{T}_p}(\sigma)(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k)\right\|^2 = \|\hat{\mathbf{x}}-\hat{\mathbf{v}}_k\|^2 + (\sigma^2-1)\left\|\Pi_{\hat{T}_p}(\hat{\mathbf{c}}_1-\hat{\mathbf{v}}_k)\right\|^2,$$

where $\mathbf{c}_1$ is an endpoint of $e_{k,\ell,p}$, and can be replaced by any point of $e_{k,\ell,p}$. Similarly, we have

$$\left\|M_{\hat{T}_p}(\sigma)(\hat{\mathbf{x}}-\hat{\mathbf{v}}_\ell)\right\|^2 = \|\hat{\mathbf{x}}-\hat{\mathbf{v}}_\ell\|^2 + (\sigma^2-1)\left\|\Pi_{\hat{T}_p}(\hat{\mathbf{c}}_1-\hat{\mathbf{v}}_\ell)\right\|^2.$$

The segment $e_{k,\ell,p}$ is included in the bisector of $\hat{\mathbf{v}}_k$ and $\hat{\mathbf{v}}_\ell$. Therefore, for any $\hat{\mathbf{x}} \in e_{k,\ell,p}$, we have $\|\hat{\mathbf{x}}-\hat{\mathbf{v}}_k\| = \|\hat{\mathbf{x}}-\hat{\mathbf{v}}_\ell\|$. Replacing in Equation 19, we finally obtain

$$\begin{aligned}
\frac{\mathrm{d}\mathscr{F}_{k,\ell,p}}{\mathrm{d}\hat{\mathbf{v}}_k} &= \int_{e_{k,\ell,p}} (\sigma^2-1)\left(\left\|\Pi_{\hat{T}_p}(\hat{\mathbf{c}}_1-\hat{\mathbf{v}}_k)\right\|^2 - \left\|\Pi_{\hat{T}_p}(\hat{\mathbf{c}}_1-\hat{\mathbf{v}}_\ell)\right\|^2\right)\frac{(\hat{\mathbf{v}}_k-\hat{\mathbf{x}})^t}{\mathbf{n}_{k,\ell,p}.(\hat{\mathbf{v}}_k-\hat{\mathbf{v}}_\ell)}\mathrm{d}\hat{\mathbf{x}} \\
&= \frac{(\sigma^2-1)}{\mathbf{n}_{k,\ell,p}.(\hat{\mathbf{v}}_k-\hat{\mathbf{v}}_\ell)}\left(\left\|\Pi_{\hat{T}_p}(\hat{\mathbf{c}}_1-\hat{\mathbf{v}}_k)\right\|^2 - \left\|\Pi_{\hat{T}_p}(\hat{\mathbf{c}}_1-\hat{\mathbf{v}}_\ell)\right\|^2\right)\int_{e_{k,\ell,p}} (\hat{\mathbf{v}}_k-\hat{\mathbf{x}})^t\,\mathrm{d}\hat{\mathbf{x}} \\
&= \frac{(1-\sigma^2)\left|e_{k,\ell,p}\right|}{\mathbf{n}_{k,\ell,p}.(\hat{\mathbf{v}}_k-\hat{\mathbf{v}}_\ell)}\left(\left\|\Pi_{\hat{T}_p}(\hat{\mathbf{c}}_1-\hat{\mathbf{v}}_k)\right\|^2 - \left\|\Pi_{\hat{T}_p}(\hat{\mathbf{c}}_1-\hat{\mathbf{v}}_\ell)\right\|^2\right)\left(\hat{\mathbf{g}}_{k,\ell,p}-\hat{\mathbf{v}}_k\right)^t, \qquad (20)
\end{aligned}$$

where $\left|e_{k,\ell,p}\right|$ is the length of $e_{k,\ell,p}$, and $\hat{\mathbf{g}}_{k,\ell,p}$ its center point.

3.2.3. Continuity

From the study of Nivoliers and Lévy [24], our objective function $\mathscr{F}$ is C^2 almost everywhere. The matrix $M_{\hat{T}_p}(\sigma)$ changes on every triangle, and is therefore a piecewise constant function, with discontinuities on every edge of $\hat{\mathcal{S}}$. The gradient of $\mathscr{F}$ therefore has discontinuities when edges of $\hat{\mathcal{S}}$ intersect non generically bisectors of the Voronoï diagram. In Equation 20, this corresponds to the case when $e_{k,\ell,p}$ coincides with an edge of the triangle $\hat{T}_p$.

3.3. Objective function computation

The computation of the restricted Voronoï diagram produces all the polygons $\hat{\Omega}_{k|\hat{T}_p}$ to be considered. On each polygon, a piece of the value and gradient of the objective function is computed. This computation is formalized in Algorithm 2. This algorithm uses simplifications similar to those of Section 3.2, in particular the fact that $\Pi_{\hat{T}_p}(\hat{\mathbf{x}}-\hat{\mathbf{v}}_k)$ is constant over the whole triangle $\hat{T}_p$, and that $\Pi^t_{\hat{T}_p}\Pi_{\hat{T}_p} = \Pi_{\hat{T}_p}$.

input : a site $\hat{\mathbf{v}}_k$, a triangle $\hat{T}_p$, the polygon $\hat{\Omega}_{k|\hat{T}_p}$, and the parameter σ.
data : the value $\mathscr{F}$ of the objective function and its gradient array $\Delta\mathscr{F}$.

1 **algorithm**
2 $\quad \hat{\mathbf{g}}_{k,p} \leftarrow \mathbf{0}, \quad \left|\hat{\Omega}_{k|\hat{T}_p}\right| \leftarrow 0$
3 $\quad$ **foreach** *triangle* $\hat{T} = (\hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2, \hat{\mathbf{c}}_3)$ *in a triangulation of* $\hat{\Omega}_{k|\hat{T}_p}$ **do** // Gradient as if no feature sensitiveness
4 $\quad\quad \hat{\mathbf{g}}_{k,p} \leftarrow \hat{\mathbf{g}}_{k,p} + |\hat{T}|\frac{\hat{\mathbf{c}}_1+\hat{\mathbf{c}}_2+\hat{\mathbf{c}}_3}{3}, \quad \left|\hat{\Omega}_{k|\hat{T}_p}\right| \leftarrow \left|\hat{\Omega}_{k|\hat{T}_p}\right| + \left|\hat{T}\right|$
5 $\quad\quad \hat{\mathbf{w}}_1 \leftarrow \hat{\mathbf{c}}_1 - \hat{\mathbf{v}}_k, \quad \hat{\mathbf{w}}_2 \leftarrow \hat{\mathbf{c}}_2 - \hat{\mathbf{v}}_k, \quad \hat{\mathbf{w}}_3 \leftarrow \hat{\mathbf{c}}_3 - \hat{\mathbf{v}}_k$
6 $\quad\quad \mathscr{F} \leftarrow \mathscr{F} + \frac{|\hat{T}|}{6}\left(\sum_{i=1}^{3}\sum_{j=1}^{3} \hat{\mathbf{w}}_i.\hat{\mathbf{w}}_j\right)$ // Equation 6
7 $\quad \Delta\mathscr{F}[k] \leftarrow \Delta\mathscr{F}[k] + 2\left(\left|\hat{\Omega}_{k|\hat{T}_p}\right| \hat{\mathbf{v}}_k - \hat{\mathbf{g}}_{k,p}\right)$ // Equation 7
8 $\quad$ **if** $\sigma > 1$ **then** // Feature sensitiveness.
9 $\quad\quad \mathbf{u}_{1,p}, \mathbf{u}_{2,p} \leftarrow$ orthogonal basis of $\hat{T}_p$ // Equation 15
10 $\quad\quad \hat{\mathbf{c}} \leftarrow$ any point of $\hat{\Omega}_{k|\hat{T}_p}$
11 $\quad\quad \hat{\mathbf{w}}_k^{\perp} \leftarrow \hat{\mathbf{c}} - \hat{\mathbf{v}}_k - \mathbf{u}_{1,p}\mathbf{u}_{1,p}^t(\hat{\mathbf{c}} - \hat{\mathbf{v}}_k) - \mathbf{u}_{2,p}\mathbf{u}_{2,p}^t(\hat{\mathbf{c}} - \hat{\mathbf{v}}_k)$
12 $\quad\quad \mathscr{F} = \mathscr{F} + 2\left|\hat{\Omega}_{k|\hat{T}_p}\right|(\sigma^2 - 1)\hat{\mathbf{w}}_k^{\perp}.\hat{\mathbf{w}}_k^{\perp}$ // missing term for Equation 11
13 $\quad\quad \Delta\mathscr{F}[k] \leftarrow \Delta\mathscr{F}[k] + 2\left|\hat{\Omega}_{k|\hat{T}_p}\right|(\sigma^2 - 1)\hat{\mathbf{w}}_k^{\perp}$ // missing term for Equation 17
14 $\quad\quad$ **foreach** *bisector edge* $e_{k,\ell,p}$ *of* $\hat{\Omega}_{k|\hat{T}_p}$ **do** // Boundary term of the gradient
15 $\quad\quad\quad \hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2 \leftarrow$ vertices of $e_{k,\ell,p}$
16 $\quad\quad\quad \mathbf{n}_{k,\ell,p} \leftarrow$ outward normal of $e_{k,\ell,p}$ in the plane of $\hat{T}_p$
17 $\quad\quad\quad \hat{\mathbf{w}}_{\ell}^{\perp} \leftarrow \hat{\mathbf{c}} - \hat{\mathbf{v}}_{\ell} - \mathbf{u}_{1,p}\mathbf{u}_{1,p}^t(\hat{\mathbf{c}} - \hat{\mathbf{v}}_{\ell}) - \mathbf{u}_{2,p}\mathbf{u}_{2,p}^t(\hat{\mathbf{c}} - \hat{\mathbf{v}}_{\ell})$
18 $\quad\quad\quad \Delta\mathscr{F}[k] \leftarrow \Delta\mathscr{F}[k] + \frac{(\sigma^2-1)\left|e_{k,\ell,p}\right|}{\mathbf{n}_{k,\ell,p}.(\hat{\mathbf{v}}_k - \hat{\mathbf{v}}_{\ell})}\left(\hat{\mathbf{w}}_k^{\perp}.\hat{\mathbf{w}}_k^{\perp} - \hat{\mathbf{w}}_{\ell}^{\perp}.\hat{\mathbf{w}}_{\ell}^{\perp}\right)\left(\hat{\mathbf{v}}_k - \frac{\hat{\mathbf{c}}_1+\hat{\mathbf{c}}_2}{2}\right)$ // Equation 20

Algorithm 2: Computing a piece of the value and gradient of $\mathscr{F}$ on $\hat{\Omega}_{k|\hat{T}_p}$. Once the restricted Voronoï diagram is computed and triangulated, the first portion (lines 4 to 7) computes the inner part of the gradient, as if no feature sensitivity was used : the area and centroids of the cells are computed. In case of feature sensitivity (line 8), we first compute an orthonormal basis for the triangle plane (line 9) to be able to compute the tangent and normal components of vectors for this triangle. The normal component of the vector between any point of the triangle and the site is computed using this basis (line 10 and 11). The value and the inner term of the gradient are then updated to take it into account the feature sensitivity (line 12 and 13). Finally, on each edge of the triangle, the boundary terms of the gradient are computed and accumulated in the final gradient (lines 14 to 18). This term only exists on edges at the interface between two cells, and to compute it, the normal component for the site of the neighboring cell is computed.

4. Local scaling

Depending on the problem to be solved and the operators to be discretized on the mesh, the definition of a good element may differ [26]. One may therefore still desire isotropic elements. We derive a local density of elements from the lifting, such that smaller elements can be generated in curved regions. This density is solely based on the lifting, and does not require any parameterization as well. As described in Section 5.1, this user is then provided with a new parameter to control the desired amount of local scaling.

4.1. Setup

To obtain isotropic small elements in curved regions, we compute a density factor ρ_p for every triangle T_p of the input surface. This factor ρ_p is then introduced in Equation 10 to weight the integrals on every triangle :

$$\mathscr{F}(\mathbf{V}, \sigma, \delta) = \sum_{\mathbf{v}_k \in \mathbf{V}} \sum_{T_p \in \mathbf{T}} \int_{\Omega_{k|T_p}} \left\| M_{T_p}(\sigma)(\mathbf{x} - \mathbf{v}_k) \right\|^2 \rho_p^{\delta} \mathrm{d}\mathbf{x}, \tag{21}$$

where $\delta \geq 0$ is a parameter controlling the amount of local scaling desired.

4.2. Gradient of the lifting

Our goal is to transform the anisotropic scaling obtained through the lifting into an isotropic scaling. Using the lifting, the anisotropic scaling is due to the difference in length between triangle edges on the lifted remeshing and their projection pack in $\mathbb{R}^3$. This difference comes from the *variations* of the lifting. Given a triangle $T_p = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ of $\mathcal{S}$ and the smooth vertex normals $\mathbf{n}_1$, $\mathbf{n}_2$, and $\mathbf{n}_3$ of its vertices, we compute the Jacobian of the lifting $\mathbf{n}$ at any point $\mathbf{x} \in T_p$. Expressing $\mathbf{x}$ in barycentric coordinates as $\mathbf{x} = u\mathbf{x}_1 + v\mathbf{x}_2 + w\mathbf{x}_3$ with $(u, v, w) \in (\mathbb{R}^+)^3$ and $u + v + w = 1$, the lifting $\mathbf{n}$ at $\mathbf{x}$ is obtained as $\mathbf{n} = u\mathbf{n}_1 + v\mathbf{n}_2 + w\mathbf{n}_3$. Its Jacobian $\frac{\mathrm{d}\mathbf{n}}{\mathrm{d}\mathbf{x}}$ is expressed as

$$\frac{\mathrm{d}\mathbf{n}}{\mathrm{d}\mathbf{x}} = \mathbf{n}_1 \frac{\mathrm{d}u}{\mathrm{d}\mathbf{x}} + \mathbf{n}_2 \frac{\mathrm{d}v}{\mathrm{d}\mathbf{x}} + \mathbf{n}_3 \frac{\mathrm{d}w}{\mathrm{d}\mathbf{x}}. \tag{22}$$

The derivative of the barycentric coordinate u is given by

$$\frac{\mathrm{d}u}{\mathrm{d}\mathbf{x}} = \frac{\mathbf{h}_1^t}{\|\mathbf{h}_1\|^2}, \text{ with } \mathbf{h}_1 = (\mathbf{x}_1 - \mathbf{x}_2) - \frac{(\mathbf{x}_1 - \mathbf{x}_2).(\mathbf{x}_3 - \mathbf{x}_2)(\mathbf{x}_3 - \mathbf{x}_2)}{\|\mathbf{x}_3 - \mathbf{x}_2\|^2}. \tag{23}$$

Here $\mathbf{h}_1$ is the height of T_p through $\mathbf{x}_1$. The matrix $\frac{\mathrm{d}\mathbf{n}}{\mathrm{d}\mathbf{x}}$ is constant over T_p, and each of its rows is the derivative of one of the coordinates appended in the lifting. Denoting $\mathrm{d}n_i$ with $i \in \{1, 2, 3\}$ these rows, we finally define the local density ρ_p as

$$\rho_p = 1 + \max_{i \in \{1,2,3\}} \|\mathrm{d}n_i\| . \tag{24}$$

5. Implementation

5.1. Parameters

Our final objective function is defined as

$$\mathscr{F}(\mathbf{V}, \sigma, \delta) = \sum_{\mathbf{v}_k \in \mathbf{V}} \sum_{T_p \in \mathbf{T}} \int_{\hat{\Omega}_{k|\hat{T}_p}} \left\| M_{\hat{T}_p}(\sigma)(\hat{\mathbf{x}} - \hat{\mathbf{v}}_k) \right\|^2 \rho_p^{\delta} \mathrm{d}\hat{\mathbf{x}}. \tag{25}$$

This objective function has two parameters, σ controlling the feature sensitivity and δ controlling the local scaling. In addition, the lifting applied to the input surface $\mathcal{S}$ uses an additional parameter ν, controlling the scaling of the normal in the lifting, and therefore the amount of anisotropy. To make these parameters independent to the mesh initial scale, we start by centering the mesh to its centroid, and scaling it so that it fits in a ball of radius 1. These modifications can be undone once the remeshing is performed.

5.2. Handling gaps in the input mesh

To make our method more robust, we implemented a few classical techniques in order to handle gaps in the input mesh.

5.2.1. Computing the normal for the lifting

We cannot rely on the sole connectivity to compute the average normals at the mesh vertices. We therefore compute these normals on the geometrical neighborhood of the vertices. Given a radius η, for each vertex of the mesh we compute the set of triangles intersecting an axis-aligned cube of side 2η centered at the vertex. We then compute the area of the intersection between these triangles and a sphere of radius η centered at the vertex, and use this area to weight the normal of the triangle in the average normal of the vertex. To implement efficiently the construction of these geometric neighborhoods, we used a classical kd-tree space decomposition technique [13, chapter 39].

5.2.2. Filling gaps in the restricted Delaunay triangulation

Gaps may also create holes in the restricted Delaunay triangulation. Dy definition, a triangle of this triangulation is generated only if the Voronoï cells of the corresponding sites intersect as a segment, and this segment intersects the surface. In practice, this intersection may fall into a gap in the input surface, and the corresponding triangle will not be generated. To fix these artifacts, we use a final hole filling procedure. Whenever a hole is detected, we compute the best plane to project its boundary vertices using principal component analysis, and fill the hole using a constrained Delaunay triangulation [6].

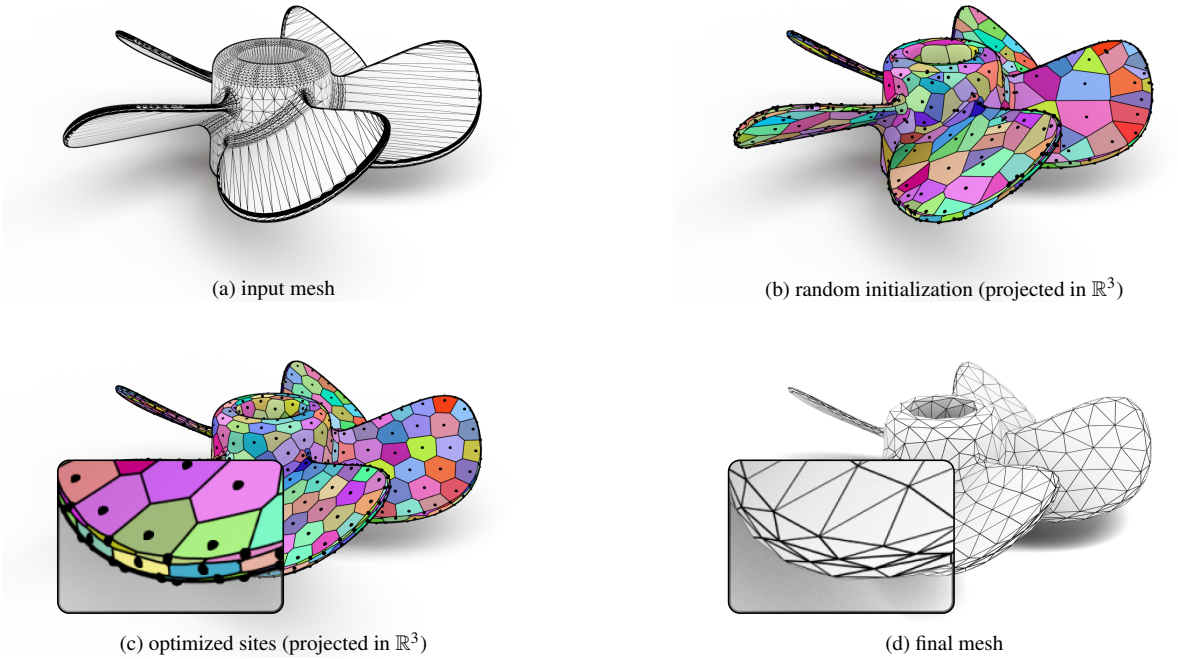
5.3. Final algorithm

Our final algorithm is described in Algorithm 3. After robustly computing the average normals at the vertices, the mesh is lifted in 6D and randomly sampled. Our objective function is then computed and the sampling optimized to minimize it. The final mesh is obtained by projecting the 6D restricted Voronoï diagram of the samples back in 3D space.

input : $\mathcal{S}$, a bad triangular mesh, parameters σ, δ, ν, η
output: $\mathcal{T}$, a mesh with regular triangles

algorithm
- $\{\mathbf{n}_i\}_i \leftarrow$ average vertex normals based on geometric neighborhoods of size η
- $\{\rho_p\}_p \leftarrow$ triangle local scaling factors // Equation 24
- $\hat{\mathcal{S}} \leftarrow$ lifting of $\mathcal{S}$ using $\{\mathbf{x}_i \oplus \nu\mathbf{n}_i\}_i$ as vertices
- $\hat{\mathbf{V}} \leftarrow$ random site initialization on $\hat{\mathcal{S}}$
- **while** *a minimum is not reached* **do**
  - Compute the restricted Voronoï diagram of $\hat{\mathbf{V}}$ and $\hat{\mathcal{S}}$
  - Compute the objective function $\mathscr{F}$ and its gradient $\frac{\mathrm{d}\mathscr{F}}{\mathrm{d}\hat{\mathbf{V}}}$ at $(\hat{\mathbf{V}}, \sigma, \delta)$ // Algorithm 2
  - $\hat{\mathbf{V}} \leftarrow \mathsf{lbfgs}(\hat{V}, \mathscr{F}, \frac{\mathrm{d}\mathscr{F}}{\mathrm{d}\hat{\mathbf{V}}})$ // Liu and Nocedal [19]
- $\hat{\mathcal{T}} \leftarrow$ restricted Delaunay triangulation of $\hat{\mathbf{V}}$ and $\hat{\mathcal{S}}$
- $\mathcal{T} \leftarrow$ orthogonal projection of $\hat{\mathcal{T}}$ in $\mathbb{R}^3$ using the first three coordinates
- **return** $\mathcal{T}$

Algorithm 3: Our remeshing technique. One normal per vertex is first computed, in order to lift the input mesh. On this lifted mesh, our initial set of sites is generated randomly with a uniform probability. Then, a loop computes the restricted Voronoï diagram of the samples, and our objective function value and gradient. Finally, once a stop criterion is reached (minimum reached or maximum iterations), the final triangulation is computed in 6D using the restricted Delaunay triangulation of the sites. This triangulation is projected back in 3D orthogonally, to provide our result.



(a) input mesh

(b) random initialization (projected in $\mathbb{R}^3$)

(c) optimized sites (projected in $\mathbb{R}^3$)

(d) final mesh

Figure 7: Our algorithm : the input surface is lifted, and randomly sampled. The sampling is optimized by minimizing our objective function, and the result is obtained using the restricted Delaunay triangulation. The anisotropy shown on the result comes from the lifting. The input model has 49k elements and is remeshed with 1000 sites, leading to 2000 elements. The optimization was limited to 40 iterations and the whole process (including normal computation and restricted Delaunay triangulation extraction) took 8.5 seconds.

6. Discussion and future work

6.1. Results

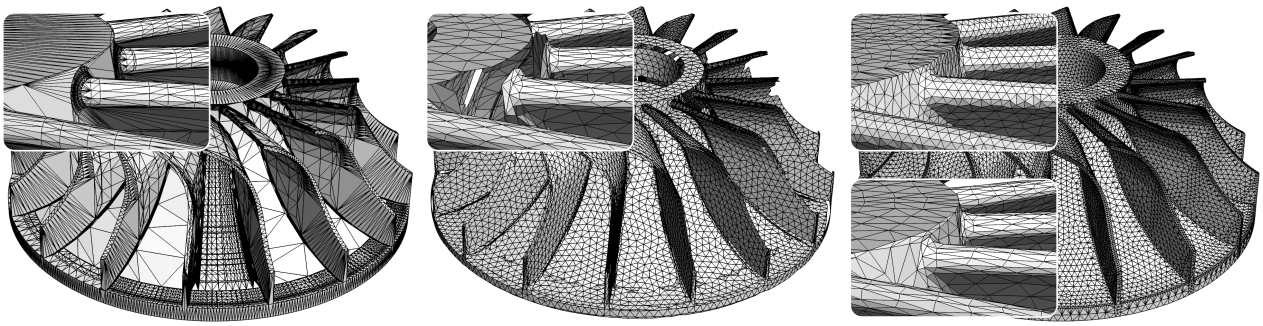
This section illustrates our results on various test cases. After showing the basic steps of the algorithm, we demonstrate the robustness of our method, and its feature sensitivity. All the meshes were generated with a single thread implementation, running on an Intel Core i7-M2620 (2.GHz to 3.4GHz) processor, and 8GB RAM. Anisotropic meshes were generated using a parameter $\nu = 0.05$. Locally scalled meshes were generated using $\nu = 0.01$ and $\delta = 0.4$. The feature sensitivity in all the results was set to $\sigma = 5$.

6.1.1. Algorithm

We start by illustrating our method on a real model, with Figure 7. Using the approach of Lévy and Bonneel [17], we obtain a new mesh with anisotropic elements in the curved regions of the input mesh. We emphasize here that using our method, no parameterization of the input mesh is used, our input is obtained from the CAD geometry by exporting the object as a mesh for rendering. The type of mesh obtained is a very coarse triangulation, solely meant to respect the shape of the objects, and approximate the spline patches with a given precision.

6.1.2. Robustness

We show on Figure 8 the results obtained by our algorithm on a mesh with bad connectivity information. This bad connectivity is due to file conversions and discretization of spline patches and patch boundaries: neighboring patches were discretized at different resolutions, causing a mismatch in the input mesh. Due to the lack of connectivity, classical parametric meshing methods like frontal methods, or methods using naive sharp feature detection to sample the sharp edges fail. The algorithm of Lévy and Bonneel [17] also fails in this case, since the normals used for the lifting are computed using the mesh connectivity. In contrast, using geometric neighborhoods to compute the normal, the connectivity missing in the input mesh is restored in the final mesh, and nice elements are produced.



(a) input mesh (b) Lévy and Bonneel [17] (c) ours, local scaling (top) or anisotropy (bottom)

Figure 8: Robustness of our approach. The connectivity of the input surface is missing in some portions of the mesh. By computing normals using the geometric neighborhoods, we can handle such degeneracies. The input model has 66k elements, and is remeshed with 30k sites, yielding 60k elements. Using 100 iterations took 102 seconds for anisotropy and 130 seconds for local scaling. Model courtesy of Pedro Marins (GrabCAD).

6.1.3. Feature sensitivity and local density

Figure 9 shows the results of our two major improvements, namely feature sensitivity and local density. Without using any heuristic algorithm to tag the sharp features, the minimization of our objective function captures these features. Using the normal variation to control the density of the elements, the curved regions of the mesh are well sampled, and fine details are correctly remeshed. The triangle aspect ratio is computed as [12]

$$\kappa = \alpha \frac{\text{inscribed radius}}{\text{circumscribed radius}} = 4 \frac{\sin a \sin b \sin c}{\sin a + \sin b + \sin c}, \tag{26}$$

where a, b and c are the angles of the measured triangle. In contrast, a frontal remeshing approach meshes each spline patch individually, and is constrained by the patch boundaries. When the boundary has very acute angles, bad triangles are generated, although the surface itself is almost flat. When the patches are not properly connected, holes also appear in the result.

6.2. Limitations

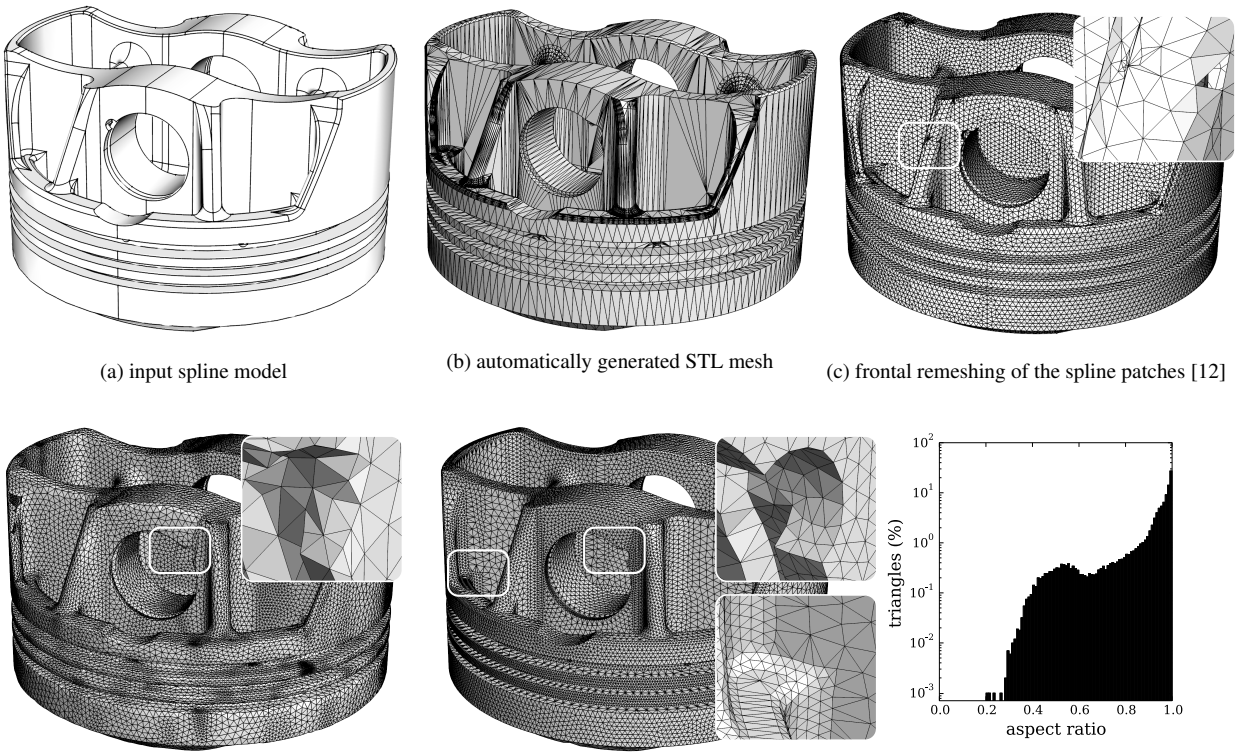
6.2.1. Efficiency

Compared to state of the art method, our approach is usually slower. With respect to the approach of Lévy and Bonneel [17], our implementation is not parallel. However, our objective function calculation algorithm is very similar to theirs, and could be integrated in their framework to gain a speedup depending on the number of cores. Frontal meshing methods are usually much faster to handle heavy meshes, but have stronger requirements. We emphasize here that our method uses no parameterization of the input geometry. Our output mesh does not depend on surface patch boundaries, but we have to recompute a restricted Voronoï diagram at every iteration of the solver.

6.2.2. Edge crunching

An artifact can appear on sharp edges, due to a local minimum of the objective function. The Equation 10 favors sites in the tangent space of the triangles in their restricted Voronoï cells. On sharp features, this can be achieved in two ways : either move the site to the sharp feature, which is the usual behavior, or align restricted cell boundaries along sharp features, such that the two neighboring cells contain at most one single tangent space. The first case usually corresponds to the behavior of the algorithm, and the second case leads to artifacts as shown on Figure 10.

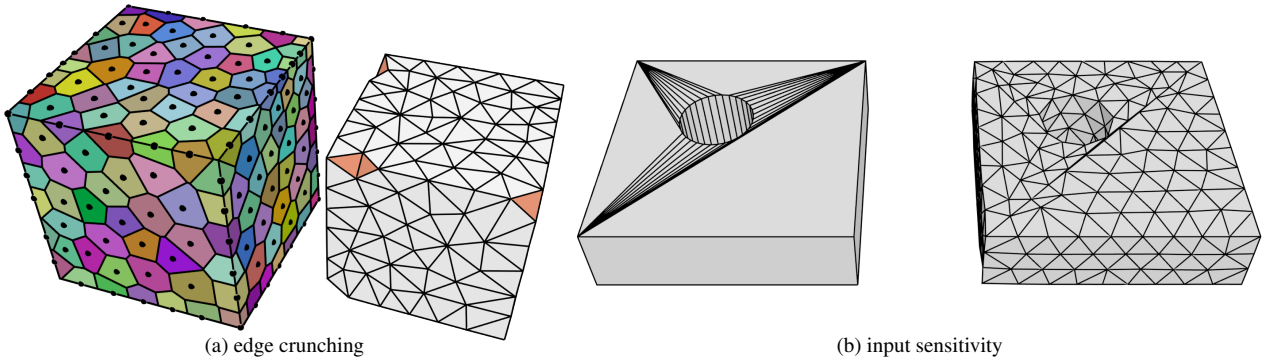
To fix this issue, our current solution is to optimize the optimization in two phases. We start the optimization without feature sensitivity for the first 90% of the required iterations. We then switch the objective function introducing feature sensitivity for the last 10% of the iterations. We use at least 10 feature sensitive iterations. Increasing the number of iterations usually decreases the number of artifacts, at the cost of a higher remeshing time. To obtain Figure 10 we solely used several feature sensitive iterations, and the diagram is clearly not centroïdal.



(a) input spline model (b) automatically generated STL mesh (c) frontal remeshing of the spline patches [12]

(d) Lévy and Bonneel [17] (e) Our method

Figure 9: Feature sensitivity and local density on a CAD model (550 spline patches, 40k elements in the STL), remeshed with 100k elements. A frontal remeshing of the spline patches [12] is constrained by the patch boundaries, and leads to bad triangles when these boundaries have acute angles, or holes when the patches are not properly connected. The algorithm of Lévy and Bonneel [17] was applied using their automatic local density, based on the local feature size. As compared to their result, our method captures well the sharp features and the density of the remeshing increases in regions where the normal varies. As shown on the closeups, we capture small details with sharp features. The final mesh has desirable angles, mostly around 60 degrees. The average triangle aspect ratio is 0.92 while the worst aspect ratio is 0.2. A peek appears around 0.5, because of triangles in the transition areas between different densities, corresponding to an angle peek around 20 degrees. Our mesh was generated in 100 seconds using 100 solver iterations, while the method of Lévy and Bonneel [17] took 48 seconds for the same amount of iterations. The frontal remeshing was done using Gmsh [12] in 51 seconds. Model courtesy of Peter Murárik (GrabCAD).



(a) edge crunching (b) input sensitivity

Figure 10: Limitations of our algorithm. Left : edge crunching appears when the boundary of Voronoï cells aligns with sharp features. Right : when the input mesh contains flat triangles with different normals on its vertices, flat triangles can be generated in the output.

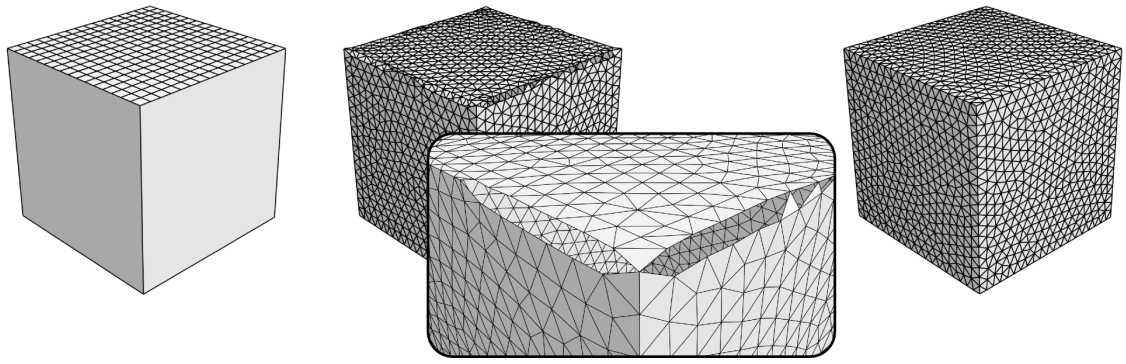


Figure 11: Remeshing problem due to the use of normal lifting on gaps with different resolutions on both sides. On the left, the input mesh is a cube with its top face disconnected and meshed with a finer resolution. In the middle, the remeshing using normal lifting has gaps, due to the fact that the normal is interpolated differently on both sides of the gap. At the middle of the edge, the gap disappears, since the interpolations match. To show the gap, the hole filling procedure described in Section 5.2.2 was disabled. Otherwise the gap is filled, and the result is manifold. On the right, the remeshing without using normal lifting provides a fine mesh.

An other solution would be to post-process the solution, however the detection of the artifacts can be difficult due to the quality of the input mesh, considering cases when the surface patches along the sharp features are not properly connected. We will explore this approach in future work.

6.2.3. Input triangulation sensitivity

Using normal lifting, the normal field of the surface is sampled on the vertices of the mesh. When the input mesh has triangles with very big angles, the gradient of the normal can be very large, and triangles that are nearly flat in 3D can become large in 6D. After remeshing, this may lead to flat triangles in the output, as shown on Figure 10.

An idea to solve this issue would be to preprocess the input mesh, using an edge splitting strategy to remove big angles, without altering the shape of the object. We also plan on testing such a method in future work.

6.2.4. Worsening gaps in the lifting

When the input mesh exhibits a gap, and the resolution of the mesh on both sides of the gap is different, The lifting may worsen the gap. This is again due to the interpolation of the normal along the edges of the mesh. If a big edge has different normals on both ends, the normal along the edge is linearly interpolated by the lifting. When this edge corresponds to a gap, and the resolution on the other side is finer, the interpolation of the normal will be different, worsening the gap because of the normal dimensions. This is illustrated on Figure 11.

7. Perspectives

In its current state, our method can already prove useful in the remeshing of bad quality CAD models : we do not require any parameterization and most gaps in the input mesh are automatically handled. Sharp features are taken into account without any feature detection algorithm and thresholding.

Apart from the future work related to the limitations, this work could be extended in several manners. In its current state, we do not control the gradation of the output mesh, which can lead to thiner triangles in transition areas as shown on Figure 9. In addition, the anisotropy or density is currently guided by the curvature of the input mesh, and cannot be easily user provided. We therefore plan on exploring the automatic generation of a lifting of the input mesh, given a user-provided metric for the mesh. Gradation control also requires the metric field to be sampled and interpolated on the input mesh, which could have insufficient quality for such a task. Existing mesh adaptation methods could prove useful as a preprocessing step to ensure a proper sampling of a user provided metric in the input mesh.

References

[1] Alauzet, F., 2010. Size gradation control of anisotropic meshes. Finite Elements in Analysis and Design 46 (1), 181–202.

[2] Asami, Y., 1991. A note on the derivation of the first and second derivative of objective functions in geographical optimization problems. Journal of the Faculty of Engineering, The University of Tokyo (B) 41 (1), 1–13.
[3] Attene, M., Falcidieno, B., Spagnuolo, M., Wyvill, G., 2003. A mapping-independent primitive for the triangulation of parametric surfaces. Graphical models 65 (5), 260–273.
[4] Baker, T. J., 2005. Mesh generation: Art or science? Progress in Aerospace Sciences 41 (1), 29–63.
[5] Béchet, E., Cuilliere, J.-C., Trochu, F., 2002. Generation of a finite element mesh from stereolithography (stl) files. Computer-Aided Design 34 (1), 1–17.
[6] Boissonnat, J.-D., Devillers, O., Teillaud, M., Yvinec, M., 2000. Triangulations in cgal. In: Proceedings of the sixteenth annual symposium on Computational geometry. ACM, pp. 11–18.
[7] Borouchaki, H., Laug, P., George, P.-L., 2000. Parametric surface meshing using a combined advancing-front generalized delaunay approach. International Journal for Numerical Methods in Engineering 49 (1-2), 233–259.
[8] Chen, Z., Wang, W., Lévy, B., Liu, L., Sun, F., 2014. Revisiting optimal delaunay triangulation for 3d graded mesh generation. SIAM Journal on Scientific Computing 36 (3), A930–A954.
[9] Dey, T. K., Ray, T., 2010. Polygonal surface remeshing with delaunay refinement. Engineering with Computers 26 (3), 289–301.
[10] Edelsbrunner, H., Shah, N., 1997. Triangulating topological spaces. International Journal of Computational Geometry and Applications 7 (4), 365–378.
[11] Frey, P., George, P.-L., 2010. Mesh generation. Vol. 32. John Wiley & Sons.
[12] Geuzaine, C., Remacle, J.-F., 2009. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. International Journal for Numerical Methods in Engineering 79 (11), 1309–1331.
[13] Goodman, J. E., O'Rourke, J., 2010. Handbook of discrete and computational geometry. CRC press.
[14] Iri, M., Murota, K., Ohya, T., 1984. A fast voronoi-diagram algorithm with applications to geographical optimization problems. System Modelling and Optimization, 273–288.
[15] Lasserre, J., Avrachenkov, K., 2001. The multi-dimensional version of $\int_a^b x^p dx$. The American Mathematical Monthly 108 (2), 151–154.
[16] Laug, P., Borouchaki, H., 2003. Interpolating and meshing 3d surface grids. International Journal for Numerical Methods in Engineering 58 (2), 209–225.
[17] Lévy, B., Bonneel, N., 2012. Variational anisotropic surface meshing with voronoi parallel linear enumeration. In: IMR-21st International Meshing Roundtable.
[18] Lévy, B., Liu, Y., 2010. Lp centroidal voronoi tesselation and its applications. ACM Transactions on Graphics 29, 4.
[19] Liu, D., Nocedal, J., 1989. On the limited memory bfgs method for large scale optimization. Mathematical programming 45 (1), 503–528.
[20] Lloyd, S., 1982. Least squares quantization in pcm. IEEE Transactions on Information Theory 28 (2), 129–137.
[21] Marchandise, E., Remacle, J.-F., Geuzaine, C., 2012. Optimal parametrizations for surface remeshing. Engineering with Computers, 1–20.
[22] Marcum, D. L., 2001. Efficient generation of high-quality unstructured surface and volume grids. Engineering with Computers 17 (3), 211–233.
[23] Muja, M., Lowe, D. G., 2014. Scalable nearest neighbor algorithms for high dimensional data. Pattern Analysis and Machine Intelligence, IEEE Transactions on 36.
[24] Nivoliers, V., Lévy, B., 2013. Approximating functions on a mesh with restricted voronoï diagrams. In: Computer Graphics Forum. Vol. 32. Wiley Online Library, pp. 83–92.
[25] Rassineux, A., Breitkopf, P., Villon, P., 2003. Simultaneous surface and tetrahedron mesh adaptation using mesh-free techniques. International journal for numerical methods in engineering 57 (3), 371–389.
[26] Shewchuk, J., 2002. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). University of California at Berkeley 73.
[27] Tristano, J. R., Owen, S. J., Canann, S. A., 1998. Advancing front surface mesh generation in parametric space using a riemannian surface definition. In: IMR. pp. 429–445.
[28] Wang, D., Hassan, O., Morgan, K., Weatherill, N., 2007. Enhanced remeshing from stl files with applications to surface grid generation. Communications in numerical methods in engineering 23 (3), 227–239.
[29] Yan, D.-M., Lévy, B., Liu, Y., Sun, F., Wang, W., 2009. Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In: Computer graphics forum. Vol. 28. Wiley Online Library, pp. 1445–1454.