



HAL
open science

Anisotropic and feature sensitive triangular remeshing using normal lifting

Vincent Nivoliers, Bruno Lévy, Christophe Geuzaine

► **To cite this version:**

Vincent Nivoliers, Bruno Lévy, Christophe Geuzaine. Anisotropic and feature sensitive triangular remeshing using normal lifting. *Journal of Computational and Applied Mathematics*, 2015, 289, pp.225-240. 10.1016/j.cam.2015.01.041 . hal-01202738

HAL Id: hal-01202738

<https://hal.science/hal-01202738>

Submitted on 21 Sep 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

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)

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

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

24 1.1. Advancing front algorithms

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

32 These methods are very efficient, but require the input object to be split into parametric patches. This is sometimes
33 not a limitation since CAD objects are often defined as a set of parametric spline patches. However the mesh is gener-
34 ated per patch, which can be problematic when many patches are used to describe the shape of the object in complicated
35 regions. Marcum [22] propose a volumetric method to merge multiple patches with a unique parameterization while
36 Attene et al. [3] define an advancing front method robust to the quality of the parameterization. Marchandise et al.
37 [21] especially target the remeshing of STL meshes, and study various automatic parameterization techniques. While
38 automatic parameterization methods exist, the topology of the input mesh needs to be properly defined, and gaps
39 cannot be handled. In addition, the parameterization is stored and thus sampled on the vertices of a background mesh.
40 While this is not a problem for dense STL meshes obtained from 3D scanning devices or extracted from voxel data,
41 the STL triangulation automatically generated from CAD data is usually too bad for automatic parameterization.

42 1.2. Mesh adaptation

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

51 1.3. Delaunay refinement and optimization

52 Delaunay refinement techniques are closer to our approach, since the connectivity of the final mesh is obtained via
53 a Delaunay triangulation and is therefore automatically handled through the vertex insertion phase. Borouchaki et al.
54 [7] combine the Delaunay triangulation with an advancing front method for vertex insertion given a certain metric.
55 The connectivity of the mesh is therefore automatically updated during the algorithm. Dey and Ray [9] follow the
56 approach of mesh adaptation by inserting vertices one by one in an existing mesh, updating the connectivity using
57 a restricted Delaunay triangulation. The authors prove guarantees on the quality of the output mesh. Both methods
58 however still require a correct input connectivity to be applied to be able to either parameterize the object, or recover
59 its topology properly.

60 Chen et al. [8] use the concept of *optimal Delaunay triangulation* to generate volume meshes, which is somehow
61 dual to our approach. An objective function is defined over the restricted Delaunay triangulation of a set of samples,
62 measuring the quality of the output elements. The optimization of the objective function leads to nicely shaped
63 elements. Although the input requirements are as weak as ours for this method, the continuity of the objective function
64 is only C^0 , and the optimization procedure requires the use of simulated annealing, whereas our objective function is
65 almost C^2 and can be optimized with a quasi-Newton solver. The objective function however is directly defined on
66 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 is 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
1 algorithm
2    $\mathbf{V} \leftarrow$  random site initialization on  $\mathcal{S}$ 
3   while a minimum is not reached do
4     Compute the restricted Voronoï diagram of  $\mathbf{V}$  and  $\mathcal{S}$ 
5     Compute the objective function  $\mathcal{F}$ 
6     Compute the gradient  $\frac{d\mathcal{F}}{d\mathbf{V}}$  of  $\mathcal{F}$ 
7      $\mathbf{V} \leftarrow$  lbfgs(  $\mathbf{V}, \mathcal{F}, \frac{d\mathcal{F}}{d\mathbf{V}}$  ) // Liu and Nocedal [19]
8    $\mathcal{T} \leftarrow$  restricted Delaunay triangulation of  $\mathbf{V}$  and  $\mathcal{S}$ 
9   return  $\mathcal{T}$ 

```

Algorithm 1: Outline of a centroidal Voronoï tessellation 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

$$\{\mathbf{x} \in \mathbb{R}^d, \mathbf{x} = u\mathbf{x}_1 + v\mathbf{x}_2 + w\mathbf{x}_3, \{u, v, w\} \subset \mathbb{R}^+, u + v + w = 1\}. \quad (1)$$

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

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

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

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



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

2.2.2. Computation

Several algorithms were designed to compute a restricted Voronoi diagram. We use the approach of Lévy and Bonneel [17], which handles meshes embedded in \mathbb{R}^d . This method replaces the traditional Voronoi 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 Voronoi 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

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

In other words, each time three restricted Voronoi 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

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

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

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

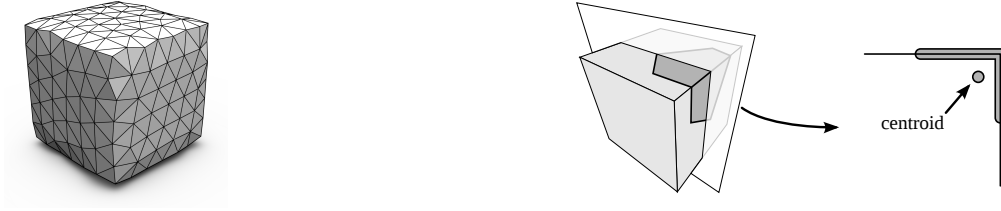


Figure 2: Remeshing a cube by the minimization of Equation 5. The sharp features are smoothed. This behavior is due to the fact that on sharp features the centroid of restricted Voronoi cells is not located on the surface.

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

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

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

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

130 2.3.2. Gradient

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

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

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

137 2.4. Feature sensitiveness

138 2.4.1. Objective function

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

142 When \mathcal{S} is in \mathbb{R}^3 , Lévy and Liu [18] propose a modification of the objective function to address this problem. Their
 143 idea is to bring back the sites on the surface, by increasing the importance of the normal component in the distance
 144 between a site and a point \mathbf{x} on the surface. Let \mathbf{n}_x be the normal of \mathcal{S} at $\mathbf{x} \in \mathcal{S}$ and $\mathbf{v}_k \in \mathbf{V}$, the normal component of
 145 $\mathbf{x} - \mathbf{v}_k$ at \mathbf{x} is given by

$$[(\mathbf{x} - \mathbf{v}_k) \cdot \mathbf{n}_x] \mathbf{n}_x = \mathbf{n}_x \mathbf{n}_x^t (\mathbf{x} - \mathbf{v}_k). \quad (8)$$

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

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

147 where σ is a parameter corresponding to the additional weighting of the normal component and I_3 is the identity
 148 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

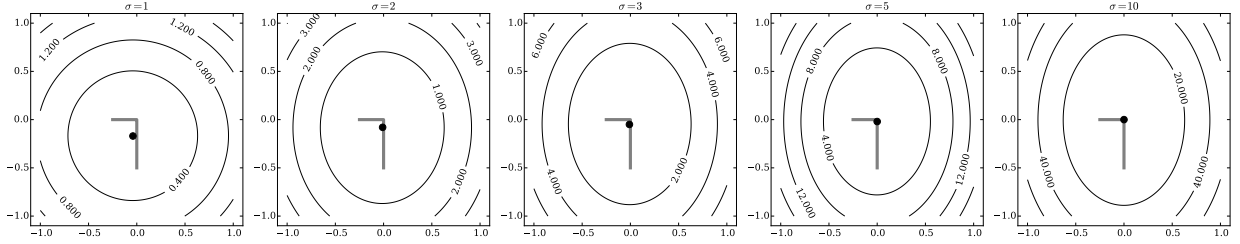


Figure 3: Isovalues of the feature sensitive objective function given by Equation 10 for different values of the feature sensitivity σ . The fat gray lines in the middle correspond to a simple 1D mesh embedded in 2D. The objective function is computed for one single site. The black dot corresponds to 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.

149 \mathcal{S} , we obtain

$$\mathcal{F}(\mathbf{V}, \sigma) = \sum_{\mathbf{v}_k \in \mathbf{V}} \sum_{T_p \in \mathbf{T}} \int_{\Omega_{k|T_p}} \|M_{T_p}(\sigma)(\mathbf{x} - \mathbf{v}_k)\|^2 dx, \quad (10)$$

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

152 2.4.2. Value

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

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

154 2.4.3. Gradient

155 With \mathcal{F} as defined in Equation 10 the expression of its gradient gets more complicated. The derivation used to
 156 obtained Equation 7 uses the fact that given two neighboring sites \mathbf{v}_k and \mathbf{v}_ℓ , for $\mathbf{x} \in \Omega_{k|\mathcal{S}} \cap \Omega_{\ell|\mathcal{S}}$, $\|\mathbf{x} - \mathbf{v}_k\| = \|\mathbf{x} - \mathbf{v}_\ell\|$
 157 since \mathbf{x} is on the bisector of \mathbf{v}_k and \mathbf{v}_ℓ . This is no longer the case using the modified distance of Equation 9, since
 158 the normal components of the two vectors are usually different. Lévy and Liu [18] therefore derive the gradient from
 159 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
 160 their final equation since we provide in Section 3.2 a simpler formulation.

161 2.5. Anisotropy through normal lifting

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

168 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
 169 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
 170 $\hat{\mathbf{X}}$ are defined as

$$\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_i \in \mathbb{R}^6, \hat{\mathbf{x}}_i = \mathbf{x}_i \oplus \nu \mathbf{n}_i, \mathbf{x}_i \in \mathbf{X}\}, \quad (12)$$

171 where \oplus concatenates the coordinates of the two provided vectors. The remeshing is then performed on the lifted
 172 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
 173 to obtain the final mesh \mathcal{T} .

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

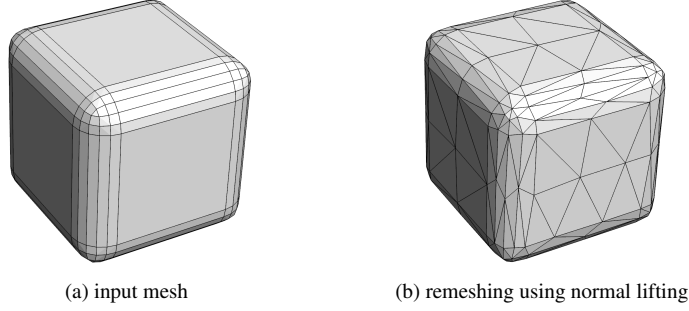


Figure 4: The input mesh is first lifted in \mathbb{R}^6 , remeshed. Projecting back in \mathbb{R}^3 yields the result, exhibiting thin triangles in curved regions.

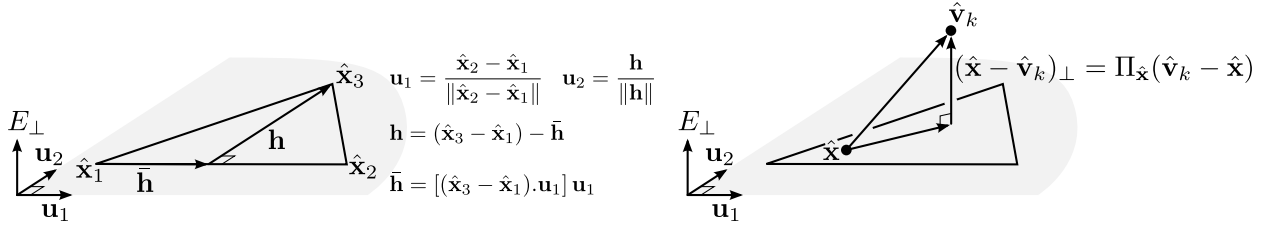


Figure 5: Computing the normal component of a vector $(\hat{\mathbf{v}}_k - \hat{\mathbf{x}})$. First an orthogonal basis $(\mathbf{u}_1, \mathbf{u}_2)$ of the triangle containing \mathbf{x} is computed. The projection of $(\hat{\mathbf{v}}_k - \hat{\mathbf{x}})$ in the normal space E_\perp is computed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

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

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

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

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

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

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

200 3.2. Objective function gradient

201 We compute the gradient of \mathcal{F} using the technique of Nivoliens and Lévy [24], with Reynolds' transport theorem.
 202 This theorem states that

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

203 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
 204 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
 205 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
 206 $\hat{\mathbf{v}}_k$. We will now study the inner term and the boundary term separately.

207 3.2.1. Inner term

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

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

210 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
 211 Equation 7.

212 3.2.2. Boundary term

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

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

217 **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
 218 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
 219 derivative of $\hat{\mathbf{x}}$ is given by

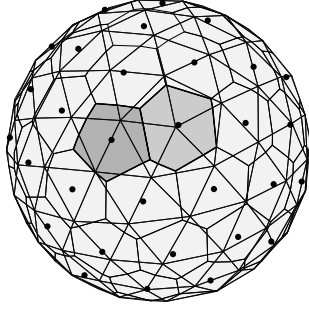
$$\frac{d\hat{\mathbf{x}}}{d\hat{\mathbf{v}}_k} \cdot \mathbf{n}_{k,\ell,p} = \frac{(\hat{\mathbf{v}}_k - \hat{\mathbf{x}})^t}{\mathbf{n}_{k,\ell,p} \cdot (\hat{\mathbf{v}}_k - \hat{\mathbf{v}}_\ell)}. \quad (18)$$

220 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 segment is a bisector edge and yields two terms in the gradient
 221 of \mathcal{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
 222 of such a segment is illustrated on Figure 6. Denoting $\frac{d\mathcal{F}_{k,\ell,p}}{d\hat{\mathbf{v}}_k}$ the sum of these two terms, we have

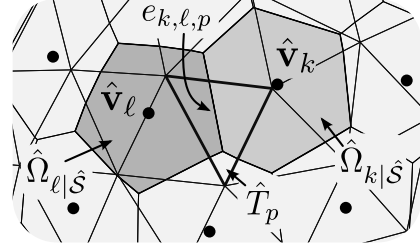
$$\frac{d\mathcal{F}_{k,\ell,p}}{d\hat{\mathbf{v}}_k} = \int_{e_{k,\ell,p}} \left(\left\| M_{\hat{T}_p}(\sigma)(\hat{\mathbf{x}} - \hat{\mathbf{v}}_k) \right\|^2 - \left\| M_{\hat{T}_p}(\sigma)(\hat{\mathbf{x}} - \hat{\mathbf{v}}_\ell) \right\|^2 \right) \frac{(\hat{\mathbf{v}}_k - \hat{\mathbf{x}})^t}{\mathbf{n}_{k,\ell,p}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

223 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 Voronoi cells



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

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

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

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

225 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 $e_{k,l,p}$, $\hat{\mathbf{x}}$ can
 226 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
 227 $\Pi_{\hat{T}_p}(\hat{\mathbf{x}} - \hat{\mathbf{v}}_k) = \Pi_{\hat{T}_p}(\hat{\mathbf{c}}_1 - \hat{\mathbf{v}}_k)$. We finally obtain

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

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

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

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

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

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

232 3.2.3. Continuity

233 From the study of Nivoliers and Lévy [24], our objective function \mathcal{F} is C^2 almost everywhere. The matrix
 234 $M_{\hat{T}_p}(\sigma)$ changes on every triangle, and is therefore a piecewise constant function, with discontinuities on every edge
 235 of \hat{S} . The gradient of \mathcal{F} therefore has discontinuities when edges of \hat{S} intersect non generically bisectors of the
 236 Voronoi diagram. In Equation 20, this corresponds to the case when $e_{k,l,p}$ coincides with an edge of the triangle \hat{T}_p .

237 3.3. Objective function computation

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

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 \mathcal{F} of the objective function and its gradient array $\Delta\mathcal{F}$.

1 algorithm

2 $\hat{\mathbf{g}}_{k,p} \leftarrow \mathbf{0}, \quad \left| \hat{\Omega}_{k|\hat{T}_p} \right| \leftarrow 0$

3 **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 $\hat{\mathbf{g}}_{k,p} \leftarrow \hat{\mathbf{g}}_{k,p} + |\hat{T}| \frac{\hat{\mathbf{c}}_1 + \hat{\mathbf{c}}_2 + \hat{\mathbf{c}}_3}{3}, \quad \left| \hat{\Omega}_{k|\hat{T}_p} \right| \leftarrow \left| \hat{\Omega}_{k|\hat{T}_p} \right| + |\hat{T}|$

5 $\hat{\mathbf{w}}_1 \leftarrow \hat{\mathbf{c}}_1 - \hat{\mathbf{v}}_k, \quad \hat{\mathbf{w}}_2 \leftarrow \hat{\mathbf{c}}_2 - \hat{\mathbf{v}}_k, \quad \hat{\mathbf{w}}_3 \leftarrow \hat{\mathbf{c}}_3 - \hat{\mathbf{v}}_k$

6 $\mathcal{F} \leftarrow \mathcal{F} + \frac{|\hat{T}|}{6} \left(\sum_{i=1}^3 \sum_{j=1}^3 \hat{\mathbf{w}}_i \cdot \hat{\mathbf{w}}_j \right)$ // Equation 6

7 $\Delta\mathcal{F}[k] \leftarrow \Delta\mathcal{F}[k] + 2 \left(\left| \hat{\Omega}_{k|\hat{T}_p} \right| \hat{\mathbf{v}}_k - \hat{\mathbf{g}}_{k,p} \right)$ // Equation 7

8 **if** $\sigma > 1$ **then** // Feature sensitiveness.

9 $\mathbf{u}_{1,p}, \mathbf{u}_{2,p} \leftarrow$ orthogonal basis of \hat{T}_p // Equation 15

10 $\hat{\mathbf{c}} \leftarrow$ any point of $\hat{\Omega}_{k|\hat{T}_p}$

11 $\hat{\mathbf{w}}_k^\perp \leftarrow \hat{\mathbf{c}} - \hat{\mathbf{v}}_k - \mathbf{u}_{1,p} \mathbf{u}_{1,p}^t (\hat{\mathbf{c}} - \hat{\mathbf{v}}_k) - \mathbf{u}_{2,p} \mathbf{u}_{2,p}^t (\hat{\mathbf{c}} - \hat{\mathbf{v}}_k)$

12 $\mathcal{F} = \mathcal{F} + 2 \left| \hat{\Omega}_{k|\hat{T}_p} \right| (\sigma^2 - 1) \hat{\mathbf{w}}_k^\perp \cdot \hat{\mathbf{w}}_k^\perp$ // missing term for Equation 11

13 $\Delta\mathcal{F}[k] \leftarrow \Delta\mathcal{F}[k] + 2 \left| \hat{\Omega}_{k|\hat{T}_p} \right| (\sigma^2 - 1) \hat{\mathbf{w}}_k^\perp$ // missing term for Equation 17

14 **foreach** bisector edge $e_{k,\ell,p}$ of $\hat{\Omega}_{k|\hat{T}_p}$ **do** // Boundary term of the gradient

15 $\hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2 \leftarrow$ vertices of $e_{k,\ell,p}$

16 $\mathbf{n}_{k,\ell,p} \leftarrow$ outward normal of $e_{k,\ell,p}$ in the plane of \hat{T}_p

17 $\hat{\mathbf{w}}_\ell^\perp \leftarrow \hat{\mathbf{c}} - \hat{\mathbf{v}}_\ell - \mathbf{u}_{1,p} \mathbf{u}_{1,p}^t (\hat{\mathbf{c}} - \hat{\mathbf{v}}_\ell) - \mathbf{u}_{2,p} \mathbf{u}_{2,p}^t (\hat{\mathbf{c}} - \hat{\mathbf{v}}_\ell)$

18 $\Delta\mathcal{F}[k] \leftarrow \Delta\mathcal{F}[k] + \frac{(\sigma^2 - 1) |e_{k,\ell,p}|}{\mathbf{n}_{k,\ell,p} \cdot (\hat{\mathbf{v}}_k - \hat{\mathbf{v}}_\ell)} \left(\hat{\mathbf{w}}_k^\perp \cdot \hat{\mathbf{w}}_k^\perp - \hat{\mathbf{w}}_\ell^\perp \cdot \hat{\mathbf{w}}_\ell^\perp \right) \left(\hat{\mathbf{v}}_k - \frac{\hat{\mathbf{c}}_1 + \hat{\mathbf{c}}_2}{2} \right)$ // Equation 20

Algorithm 2: Computing a piece of the value and gradient of \mathcal{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.

242 4. Local scaling

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

248 4.1. Setup

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

$$\mathcal{F}(\mathbf{V}, \sigma, \delta) = \sum_{\mathbf{v}_k \in \mathbf{V}} \sum_{T_p \in \mathbf{T}} \int_{\Omega_{kl} T_p} \|M_{T_p}(\sigma)(\mathbf{x} - \mathbf{v}_k)\|^2 \rho_p^\delta d\mathbf{x}, \quad (21)$$

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

252 4.2. Gradient of the lifting

253 Our goal is to transform the anisotropic scaling obtained through the lifting into an isotropic scaling. Using the
 254 lifting, the anisotropic scaling is due to the difference in length between triangle edges on the lifted remeshing and
 255 their projection pack in \mathbb{R}^3 . This difference comes from the *variations* of the lifting. Given a triangle $T_p = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$
 256 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
 257 $\mathbf{x} \in T_p$. 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
 258 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 $\frac{d\mathbf{n}}{d\mathbf{x}}$ is expressed as

$$\frac{d\mathbf{n}}{d\mathbf{x}} = \mathbf{n}_1 \frac{du}{d\mathbf{x}} + \mathbf{n}_2 \frac{dv}{d\mathbf{x}} + \mathbf{n}_3 \frac{dw}{d\mathbf{x}}. \quad (22)$$

259 The derivative of the barycentric coordinate u is given by

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

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

$$\rho_p = 1 + \max_{i \in \{1, 2, 3\}} \|dn_i\|. \quad (24)$$

263 5. Implementation

264 5.1. Parameters

265 Our final objective function is defined as

$$\mathcal{F}(\mathbf{V}, \sigma, \delta) = \sum_{\mathbf{v}_k \in \mathbf{V}} \sum_{T_p \in \mathbf{T}} \int_{\hat{\Omega}_{kl} T_p} \|M_{T_p}(\sigma)(\hat{\mathbf{x}} - \hat{\mathbf{v}}_k)\|^2 \rho_p^\delta d\hat{\mathbf{x}}. \quad (25)$$

266 This objective function has two parameters, σ controlling the feature sensitivity and δ controlling the local scaling. In
 267 addition, the lifting applied to the input surface \mathcal{S} uses an additional parameter ν , controlling the scaling of the normal
 268 in the lifting, and therefore the amount of anisotropy. To make these parameters independent to the mesh initial scale,
 269 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
 270 be undone once the remeshing is performed.

271 5.2. Handling gaps in the input mesh

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

274 5.2.1. Computing the normal for the lifting

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

281 5.2.2. Filling gaps in the restricted Delaunay triangulation

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

288 5.3. Final algorithm

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

input : \mathcal{S} , a bad triangular mesh, parameters $\sigma, \delta, \nu, \eta$
output: \mathcal{T} , a mesh with regular triangles

```

1 algorithm
2    $\{\mathbf{n}_i\}_i \leftarrow$  average vertex normals based on geometric neighborhoods of size  $\eta$ 
3    $\{\rho_p\}_p \leftarrow$  triangle local scaling factors // Equation 24
4    $\hat{\mathcal{S}} \leftarrow$  lifting of  $\mathcal{S}$  using  $\{\mathbf{x}_i \oplus \nu \mathbf{n}_i\}_i$  as vertices
5    $\hat{\mathbf{V}} \leftarrow$  random site initialization on  $\hat{\mathcal{S}}$ 
6   while a minimum is not reached do
7     Compute the restricted Voronoï diagram of  $\hat{\mathbf{V}}$  and  $\hat{\mathcal{S}}$ 
8     Compute the objective function  $\mathcal{F}$  and its gradient  $\frac{d\mathcal{F}}{d\hat{\mathbf{V}}}$  at  $(\hat{\mathbf{V}}, \sigma, \delta)$  // Algorithm 2
9      $\hat{\mathbf{V}} \leftarrow$  lbfgs( $\hat{\mathbf{V}}, \mathcal{F}, \frac{d\mathcal{F}}{d\hat{\mathbf{V}}}$ ) // Liu and Nocedal [19]
10   $\hat{\mathcal{T}} \leftarrow$  restricted Delaunay triangulation of  $\hat{\mathbf{V}}$  and  $\hat{\mathcal{S}}$ 
11   $\mathcal{T} \leftarrow$  orthogonal projection of  $\hat{\mathcal{T}}$  in  $\mathbb{R}^3$  using the first three coordinates
12  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.

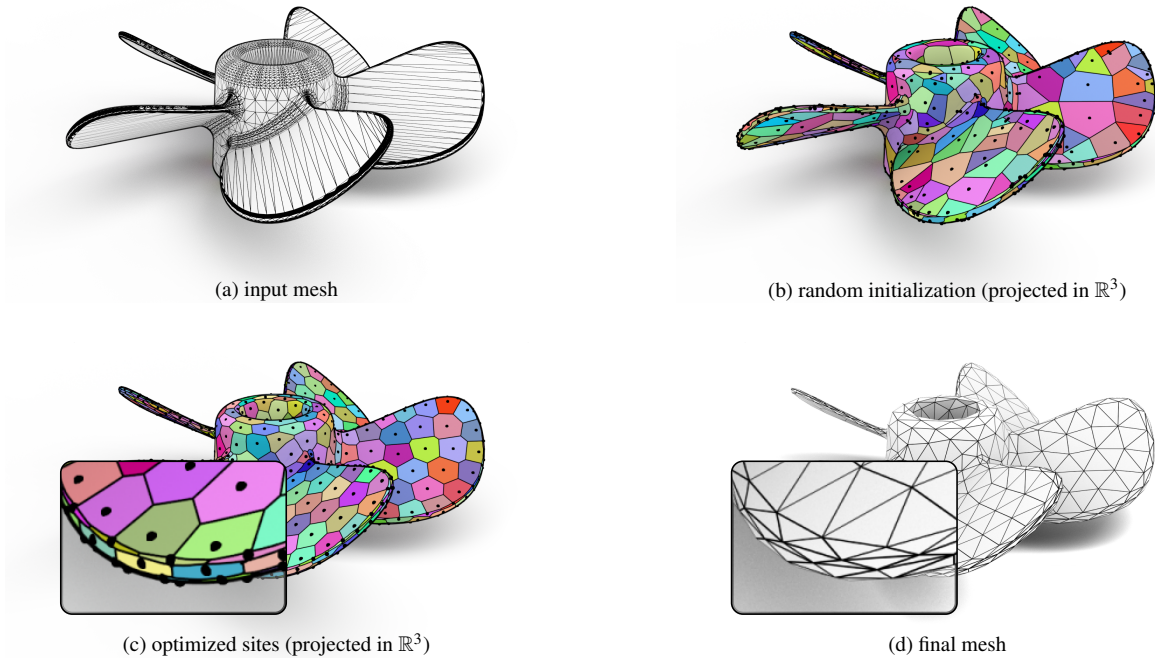


Figure 7: Our algorithm : the input surface is lifted, and randomly sampled. The sampling is optimized by minimizing our objective function, and the result is obtained using the restricted Delaunay triangulation. The anisotropy shown on the result comes from the lifting. The input model has 49k elements 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.

293 6. Discussion and future work

294 6.1. Results

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

300 6.1.1. Algorithm

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

306 6.1.2. Robustness

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

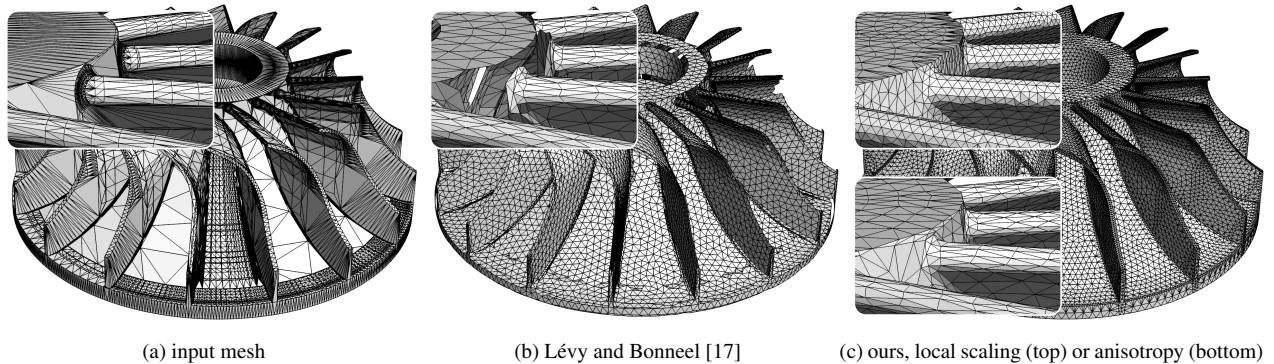


Figure 8: Robustness of our approach. The connectivity of the input surface is missing in some portions of the mesh. By computing normals using the geometric neighborhoods, we can handle such degeneracies. The input model has 66k elements, and is remeshed with 30k sites, yielding 60k elements. Using 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]

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

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 centroidal.

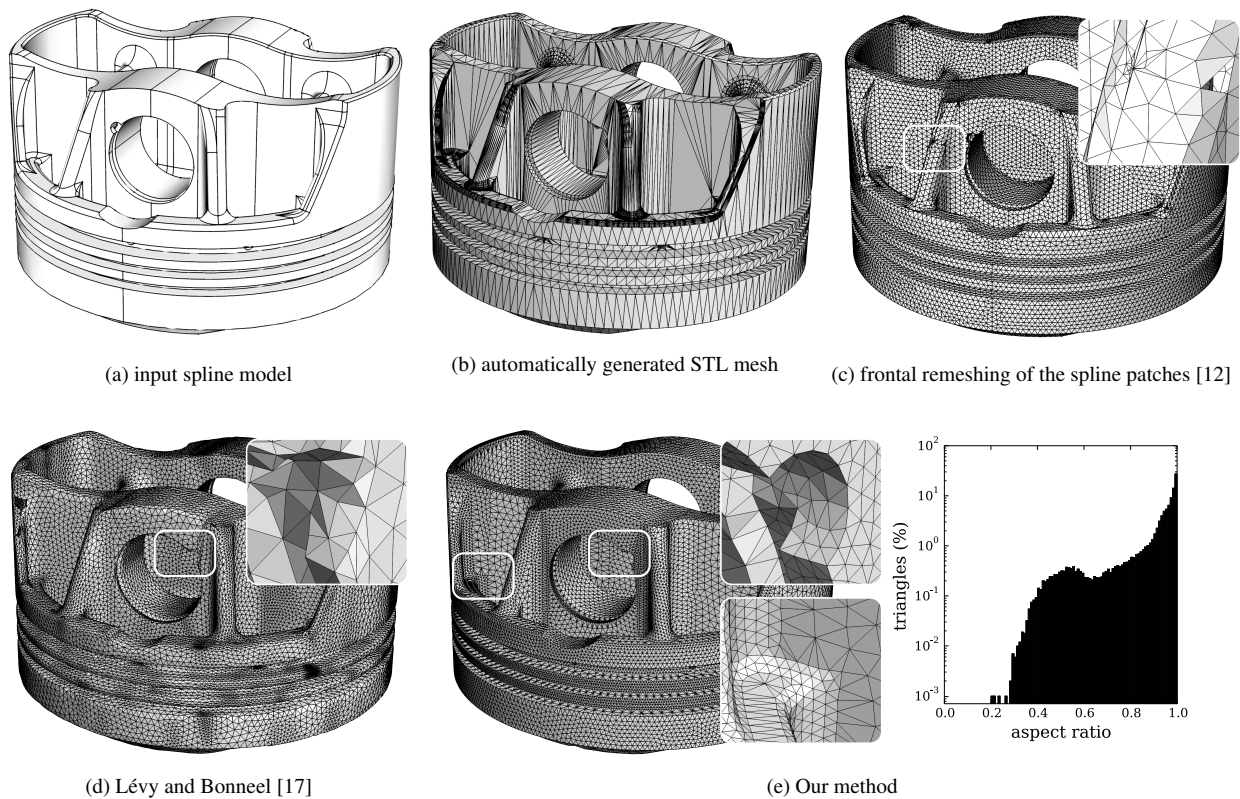


Figure 9: Feature sensitivity and local density on a CAD model (550 spline patches, 40k elements in the STL), remeshed with 100k elements. A frontal remeshing of the spline patches [12] is constrained by the patch boundaries, and leads to bad triangles when these boundaries have acute angles, or holes 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).

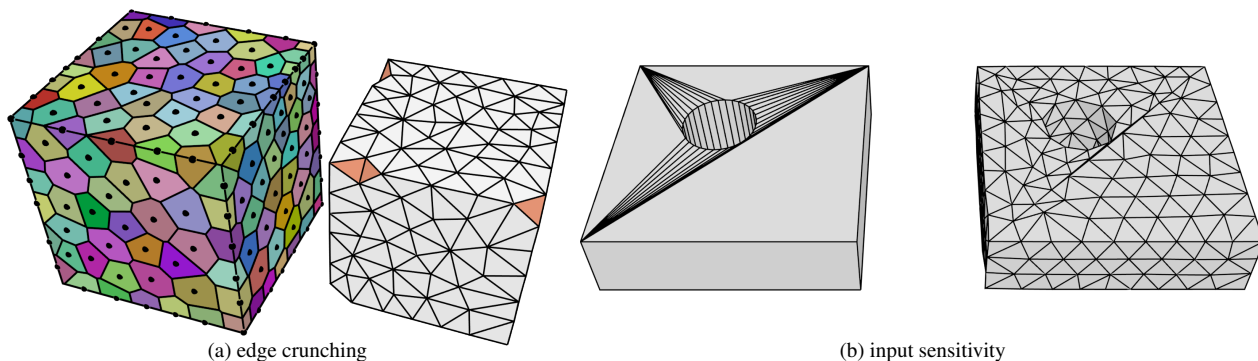


Figure 10: Limitations of our algorithm. Left : edge crunching appears when the boundary of Voronoi cells aligns with sharp features. Right : when the input mesh contains flat triangles with different normals on its vertices, flat triangles can be generated in the output.

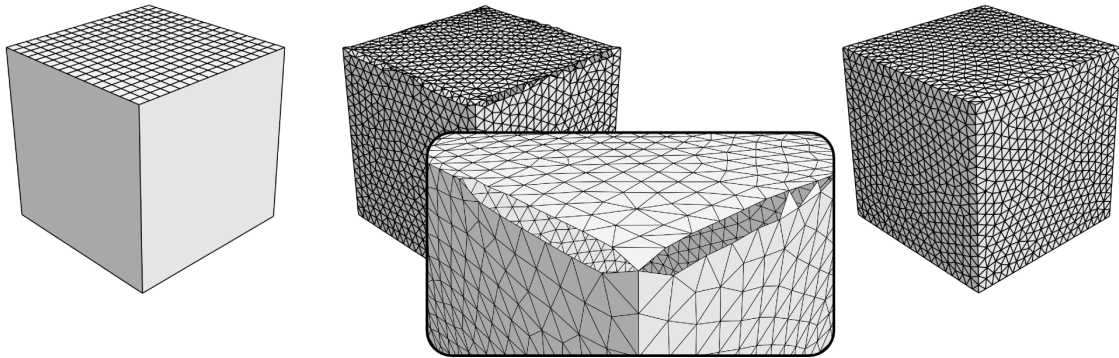


Figure 11: Remeshing problem due to the use of normal lifting on gaps with different resolutions on both sides. On the left, the input mesh is a cube with its top face disconnected and meshed with a finer resolution. In the middle, the remeshing using normal lifting has gaps, due to the fact that the 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.

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

345 6.2.3. Input triangulation sensitivity

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

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

351 6.2.4. Worsening gaps in the lifting

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

357 7. Perspectives

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

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

368 References

369 [1] Alauzet, F., 2010. Size gradation control of anisotropic meshes. *Finite Elements in Analysis and Design* 46 (1), 181–202.

- 370 [2] Asami, Y., 1991. A note on the derivation of the first and second derivative of objective functions in geographical optimization problems.
371 Journal of the Faculty of Engineering, The University of Tokyo (B) 41 (1), 1–13.
- 372 [3] Attene, M., Falcidieno, B., Spagnuolo, M., Wyvill, G., 2003. A mapping-independent primitive for the triangulation of parametric surfaces.
373 Graphical models 65 (5), 260–273.
- 374 [4] Baker, T. J., 2005. Mesh generation: Art or science? Progress in Aerospace Sciences 41 (1), 29–63.
- 375 [5] Béchet, E., Cuilliere, J.-C., Trochu, F., 2002. Generation of a finite element mesh from stereolithography (stl) files. Computer-Aided Design
376 34 (1), 1–17.
- 377 [6] Boissonnat, J.-D., Devillers, O., Teillaud, M., Yvinec, M., 2000. Triangulations in cgal. In: Proceedings of the sixteenth annual symposium
378 on Computational geometry. ACM, pp. 11–18.
- 379 [7] Borouchaki, H., Laug, P., George, P.-L., 2000. Parametric surface meshing using a combined advancing-front generalized delaunay approach.
380 International Journal for Numerical Methods in Engineering 49 (1-2), 233–259.
- 381 [8] Chen, Z., Wang, W., Lévy, B., Liu, L., Sun, F., 2014. Revisiting optimal delaunay triangulation for 3d graded mesh generation. SIAM Journal
382 on Scientific Computing 36 (3), A930–A954.
- 383 [9] Dey, T. K., Ray, T., 2010. Polygonal surface remeshing with delaunay refinement. Engineering with Computers 26 (3), 289–301.
- 384 [10] Edelsbrunner, H., Shah, N., 1997. Triangulating topological spaces. International Journal of Computational Geometry and Applications 7 (4),
385 365–378.
- 386 [11] Frey, P., George, P.-L., 2010. Mesh generation. Vol. 32. John Wiley & Sons.
- 387 [12] Geuzaine, C., Remacle, J.-F., 2009. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. International
388 Journal for Numerical Methods in Engineering 79 (11), 1309–1331.
- 389 [13] Goodman, J. E., O'Rourke, J., 2010. Handbook of discrete and computational geometry. CRC press.
- 390 [14] Iri, M., Murota, K., Ohya, T., 1984. A fast voronoi-diagram algorithm with applications to geographical optimization problems. System
391 Modelling and Optimization, 273–288.
- 392 [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.
- 393 [16] Laug, P., Borouchaki, H., 2003. Interpolating and meshing 3d surface grids. International Journal for Numerical Methods in Engineering
394 58 (2), 209–225.
- 395 [17] Lévy, B., Bonneel, N., 2012. Variational anisotropic surface meshing with voronoi parallel linear enumeration. In: IMR-21st International
396 Meshing Roundtable.
- 397 [18] Lévy, B., Liu, Y., 2010. Lp centroidal voronoi tessellation and its applications. ACM Transactions on Graphics 29, 4.
- 398 [19] Liu, D., Nocedal, J., 1989. On the limited memory bfgs method for large scale optimization. Mathematical programming 45 (1), 503–528.
- 399 [20] Lloyd, S., 1982. Least squares quantization in pcm. IEEE Transactions on Information Theory 28 (2), 129–137.
- 400 [21] Marchandise, E., Remacle, J.-F., Geuzaine, C., 2012. Optimal parametrizations for surface remeshing. Engineering with Computers, 1–20.
- 401 [22] Marcum, D. L., 2001. Efficient generation of high-quality unstructured surface and volume grids. Engineering with Computers 17 (3), 211–
402 233.
- 403 [23] Muja, M., Lowe, D. G., 2014. Scalable nearest neighbor algorithms for high dimensional data. Pattern Analysis and Machine Intelligence,
404 IEEE Transactions on 36.
- 405 [24] Nivoliers, V., Lévy, B., 2013. Approximating functions on a mesh with restricted voronoi diagrams. In: Computer Graphics Forum. Vol. 32.
406 Wiley Online Library, pp. 83–92.
- 407 [25] Rassineux, A., Breilkopf, P., Villon, P., 2003. Simultaneous surface and tetrahedron mesh adaptation using mesh-free techniques. International
408 journal for numerical methods in engineering 57 (3), 371–389.
- 409 [26] Shewchuk, J., 2002. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). University
410 of California at Berkeley 73.
- 411 [27] Tristano, J. R., Owen, S. J., Canann, S. A., 1998. Advancing front surface mesh generation in parametric space using a riemannian surface
412 definition. In: IMR. pp. 429–445.
- 413 [28] Wang, D., Hassan, O., Morgan, K., Weatherill, N., 2007. Enhanced remeshing from stl files with applications to surface grid generation.
414 Communications in numerical methods in engineering 23 (3), 227–239.
- 415 [29] Yan, D.-M., Lévy, B., Liu, Y., Sun, F., Wang, W., 2009. Isotropic remeshing with fast and exact computation of restricted voronoi diagram.
416 In: Computer graphics forum. Vol. 28. Wiley Online Library, pp. 1445–1454.