Weierstraß-Institut
für Angewandte Analysis und Stochastik
Leibniz-Institut im Forschungsverbund Berlin e. V.

Preprint ISSN 0946 – 8633

A curvature-adapted anisotropic surface remeshing method

Franco Dassi[1], Hang Si[2]

submitted: October 9, 2013

[1] Politecnico di Milano
Piazza Leonardo da Vinci 32
20133 Milano
Italy
email: dassifraa@gmail.com

[2] Weierstrass Institute
Mohrenstr. 39
10117 Berlin
Germany
email: si@wias-berlin.de

No. 1850
Berlin 2013

2010 *Mathematics Subject Classification.* 65M50, 65N50, 65D18, 68U05, 68N99.

Key words and phrases. surface mesh generation, surface remeshing, curvature-adapted, anisotropic, mesh optimization, edge flip.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Leibniz-Institut im Forschungsverbund Berlin e. V.
Mohrenstraße 39
10117 Berlin
Germany

Fax: +49 30 20372-303
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Abstract

We present a new method for remeshing surfaces that respect the intrinsic anisotropy of the surfaces. In particular, we use the normal informations of the surfaces, and embed the surfaces into a higher dimensional space (here we use 6d). This allow us to form an isotropic mesh optimization problem in this embedded space. Starting from an initial mesh of a surface, we optimize the mesh by improving the mesh quality measured in the embedded space. The mesh is optimized by combining common local modifications operations, i.e., edge flip, edge contraction, vertex smoothing, and vertex insertion. All operations are applied directly on the 3d surface mesh. This method results a curvature-adapted mesh of the surface. This method can be easily adapted to mesh multi-patches surfaces, i.e., containing corner singularities and sharp features. The reliability and robustness of the proposed re-meshing technique is provided by a large number of examples including both implicit surfaces and CAD models.

1 Introduction

Surface mesh generation is a central topic in computer visualization, geometry processing, and numerical simulation. Many computational applications involve triangulation of complex surface geometry. The main challenge is to automatically generate a surface mesh which satisfies various criteria with respect to geometry approximation, mesh size, and mesh quality.

The goal of current work is to generate a surface mesh which well approximate the geometry of the surface and with the number of elements as small as possible. For this purpose, it is desired that the resulting mesh is not only a well approximation of the geometry of the surfaces, but also a reflex of the

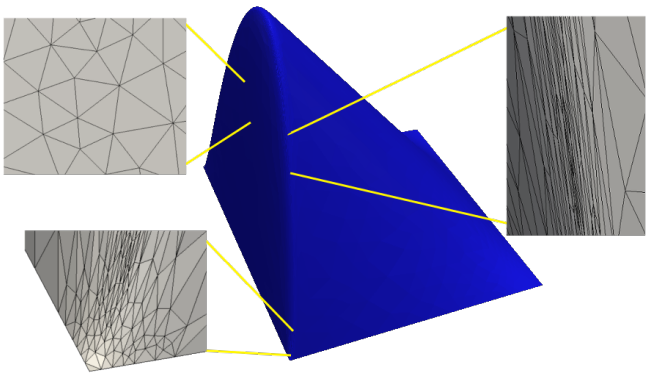

Figure 1. The best approximation of a surface (shown in the middle) must consist of mesh elements of different size, shape, and orientation that respect to the principle curvatures of that surface.

intrinsic nature of the surface, i.e., the curvature. Intuitively, more curved regions of the surface will contain small elements and a dense vertex sampling, while almost flat regions will have large elements with more sparse vertices. However, using only isotropic elements may be far from optimal for these purposes. But an anisotropic mesh, i.e., a mesh with stretched elements could offer even a better behavior “number of elements vs geometry fitting". Figure 1 shows an example.

In this paper, we propose a new method for remeshing of 3d surfaces based on the idea of higher dimensional embedding [7, 30, 32]. We use the normal informations of the surfaces, and embed the surfaces into $\mathbb{R}^6$. Our method directly optimizes a two-dimensional triangular mesh of the surface embedded in $\mathbb{R}^6$ in such a way that its triangles are as uniform as possible in $\mathbb{R}^6$. It will result a curvature-adapted anisotropic mesh of the surface. This method has the following properties:

- The core operation of this method is a uniform remeshing of a surface. It fits the well-developed mesh adaptation strategy. It is possible to re-use any optimization-based isotropic surface remeshing method.
- It can handle arbitrary complicated geometries and topologies, as well as very strong anisotropy (which has very high aspect ratio).
- It automatically preserves sharp features, corner and edge singularities.
- It is robust. For instance, the initial mesh can be very coarse or crude in geometry approximation.
- It is easy to implement.

Our method does have the following limitations:

- It only ensures the mesh quality in the embedding space, but not the usually mesh quality (such as the smallest angle) in $\mathbb{R}^3$.

- It does not support user-defined metric tensor fields.

The remainder of this paper is organized as follows: Section 2 presents the background about anisotropic meshes and reviews the related works on anisotropic surface remeshing. In Section 3, the main idea of the embedding space is introduced. The proposed method is described in detail in Section 4. Some experimental results of remeshing surfaces from implicit functions and CAD models are demonstrated in Section 5. Finally, a summarization and outlook of future works are given in Section 6.

2 Background and Related Work

2.1 Anisotropic Meshes

Many physical problems exhibit anisotropic features, i.e., their solutions change more significantly in one direction than others. Examples include in particular convection-dominated problems whose solutions have, e.g., layers, shocks, or corner and edge singularities. Anisotropic meshes have great importance in numerical methods to solve partial differential equations. They improve the accuracy of the solution and decrease the computational cost.

Anisotropy denotes the way distances and angles are distorted. It is naturally related to approximation theory and is important in function interpolation [40, 41, 9, 37]. For a smooth function, the anisotropy is best characterized by the Hessian of that function. In practice, a central question is how to efficiently distinguish the anisotropy of a given problem. Another important question is how to characterize the anisotropy in a such a way that an optimal mesh for a given problem can be defined. These are all difficult questions and are active research topics. In practice, a Riemannian metric tensor field (either provided or automatically derived) is used to guide the generation of anisotropic meshes [20, 25, 35, 36]. Mesh adaptation has been proven an efficient way to capture the anisotropy, see e.g. [18, 13].

2.2 Related Works

Surface remeshing has been an active research subject for nearly two decades, see a nice survey given by Alliez *et al.* [1]. Most of the early methods work either in a 2d parameterization space or directly in the 3d space. And they focus on creating isotropic meshes of 3d surfaces. Many recent methods have been developed for creating anisotropic meshes of a surface. In the following, we give an overview of works which are related to ours. More complete survey are available [5, 42].

Curvature evaluation and reconstruction

Heckbert and Garland [22] proposed a quadric-based metric tensor for surface simplification. It is defined for each vertex of the mesh and uses the face normals around it. They showed that this metric is directly related to surface curvature. Jiao [26] used it to derive a Riemannian metric field for anisotropic surface mesh adaptation. The method of Alliez *et al.* [2] first constructs a curvature tensor field from a given polygonal surface mesh by estimating the curvature tensor at every vertex. Then they trace lines along the principle curvature directions from which a quad-dominant anisotropic mesh can be obtained. These methods, which rely on a reconstructed metric field, are suitable on remeshing of polygonal meshes whose original geometry are not available. As we will show in Section 3, for surfaces whose the geometry are accessible, such as implicit functions and CAD models, there is a natural way to perform a remeshing of the surface respecting its curvature without using a reconstructed metric tensor field.

Delaunay refinement based methods

Voronoi diagrams and their dual Delaunay triangulations are fundamental data structures and have numerous applications [4]. The so-called *restricted Voronoi diagram* (RVD) and *restricted Delaunay triangulation* (RDT) [17] are their generalization to surfaces. Cheng *et al.* [11] proposed to generate an anisotropic RDT from a 3d anisotropic RVD [29]. Although their concept is valid in theory, it remains a big challenge in practice to ensure that the dual of an anisotropic RVD admits a valid triangulation. Boissonnat *et al* [6] proposed the notion of locally uniform anisotropic Delaunay meshes, and proposed a practical Delaunay refinement algorithm to generate an anisotropic RDT of a surface with respect to a given metric field [5]. By using the curvature tensor of the surface as input, this algorithm is able to produce a curvature-adapted anisotropic mesh of the surface. However, a fundamental difficulty of Delaunay refinement based algorithms is that it does not respect sharp features. Cheng *et al.* [10] showed that sharp features can be respected by using weighted restricted Delaunay triangulations. However, to efficiently assign appropriate weights remains a challenging problem.

Restricted centroidal Voronoi based methods

A Centroidal Voronoi Tessellation (CVT) is a particular type of Voronoi diagram such that its generating points coincide with the centroids (center of mass) of its Voronoi cells. It has a highly regular structure such that the restricted Delaunay triangulation obtained from it will have well-shaped triangles. It has been applied in many applications including surface mesh generation [14, 3, 33]. Efficient algorithms are proposed to generate CVTs [34]. It has been further generalized to anisotropic CVT with respect to a Riemannian metric field [15],

and an anisotropic restricted Voronoi diagram on surfaces can be defined. However, more theoretical analysis are needed in understanding anisotropic CVTs. And it remains a challenge to efficiently generate them. Levy and Bonneel [32] proposed a novel approach to compute CVT in higher dimensions. They used it to generate anisotropic curvature-adapted surface meshes. We will discuss it in detail in Section 3.

3 Surface Embedding in $\mathbb{R}^6$

The remeshing method proposed in this paper is inspired by the method of Lévy and Bonneel [32]. The basic idea is pioneered by Cañas and Gortler [7] and Lai *et al* [30] and is originated in the application of feature characterization [39] from image processing [27]. It treats the anisotropy by increasing the dimensions such that it becomes an isotropy in a higher dimensional space: *an isotropic mesh in a higher dimensional space will correspond to an anisotropic mesh in the lower dimensional space.* This concept has been successfully applied in generating curvature-adapted surface meshes [28, 32]

For a smooth surface, it is natural to consider the unit normals defined on surface points as the components of the codimension, i.e., using the components of the Gaussian map of the surface.

Given a surface Ω in $\mathbb{R}^3$, one can embed it into $\mathbb{R}^6$ by using the embedding: $\Phi : \Omega \to \mathbb{R}^6$ defined by:

$$\Phi(\mathbf{x}) = \begin{bmatrix} x \\ y \\ z \\ s\,n_x \\ s\,n_y \\ s\,n_z \end{bmatrix},$$

where (n_x, n_y, n_z) denotes the unit normal to Ω at $\mathbf{x}$, and $s \in [0, +\infty)$ is a user-specified constant. This embedding Φ allows us to approximate the geodesic edge lengths in Ω by the Euclidean edge lengths in $\Phi(\Omega)$. Each edge length in $\Phi(\Omega)$ is determined by two parts:

- its Euclidean length in $\mathbb{R}^3$; and
- the variation of the normals of its endpoints, scaled by the parameter s.

By this transformation, in flat regions of Ω, the lengths of edges remain the same in $\Phi(\Omega)$. While in regions which have high curvatures, the lengths of edges in $\Phi(\Omega)$ become much larger than theirs in $\mathbb{R}^3$, see Figure 2.

Since the distance in $\mathbb{R}^6$ are affected by the normals, an isotropic mesh of the surface $\Phi(\Omega)$ in $\mathbb{R}^6$, when transformed back into $\mathbb{R}^3$, will become a curvature-adapted anisotropic mesh of Ω.

As it is pointed out in [30] that the $6d$ embedding Φ is mainly for a simple transfer from non-isotropic to isotropic configuration. As will be seen from

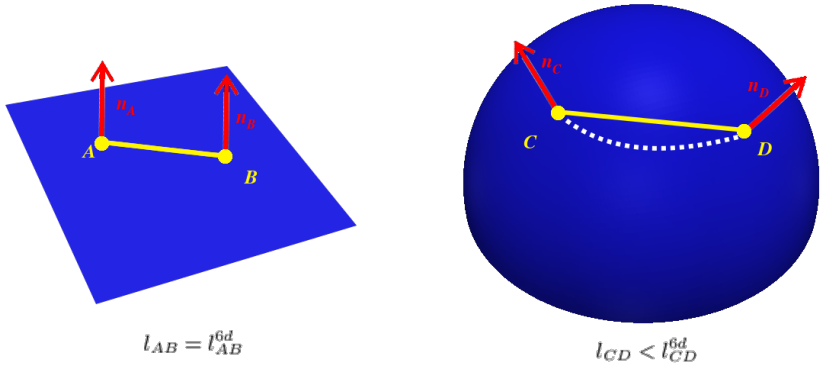

Figure 2. Different 6d-lengths with different type of surfaces, on the left a plane where the 6d and the 3d lengths coincide, $l^{6d}_{AB} = l_{AB}$, on the right a sphere where they do not coincide, $l^{6d}_{CD} > l_{CD}$.

the developments given below, we can still explain everything in $\mathbb{R}^3$ via an appropriately combined processing of points and normals and thus the use of the embedding in $\mathbb{R}^6$ does not result in any computational overhead over working in 3d.

By embedding a surface in higher dimensions motivates the new problem: *How to generate an isotropic good quality surface mesh in this embedding space?* In principle, a direct generalization of available methods in 3d is possible. But this will be impractical due to the $d!$ cost of memory requirement.

Lévy and Bonneel [32] overcome this difficulty by using their *Vorpaline* (Voronoi Parallel Linear Enumeration) technique to compute a restricted centroidal Voronoi diagram (CVT) embedded in 6d. It directly compute the Voronoi cells by iterative half-space clipping. It requires only the nearest neighbor informations for a point set in $\mathbb{R}^6$.

It is reported by Lévy and Bonneel [32] that this method may produce flipped (self-intersected) triangles, in particular in regions where the anisotropy varies too fast. This fact implies that if the normals between the neighbor vertices are varying too big, their method may not work correctly. One possible way to resolve this problem is to insert new vertices between these neighboring vertices. However, the method of Lévy and Bonneel [32] does not support inserting new vertices dynamically.

Another well-known problem in RVD- and CVT-based methods is that sharp features or details of the surfaces may be smoothed or missing in the resulting mesh. Although a theoretical solution has been proposed [10], but its efficiency is still a challenge in practice.

Due to these problems, we propose a new method in the next section. In particular, we show that a common mesh optimization framework for isotropic

surface remeshing could be applied in remeshing surfaces embedded in higher dimensional space. The primary difference between previous methods and ours is the use of lengths and angles evaluated in the embedded space.

4 The Remeshing Approach

Consider a surface Ω in $\mathbb{R}^3$. For simplicity, we assume that Ω is a smooth surface, i.e., it contains no corner and edge singularities. The non-smooth case can be easily handled and will be discuss in Subsection 4.6. In this section, we propose an optimization-based method for remeshing of 3d surfaces.

4.1 Preliminaries

Our method assumes the following two functions are provided:

(1) given a point $\mathbf{p} \in \Omega$, returns the normal of Ω at $\mathbf{p}$; and
(2) given a point $\mathbf{p} \in \mathbb{R}^3$, returns the closest point $\mathbf{q} \in \Omega$.

If Ω is represented by an implicit function or it is a parameterized surface (of CAD models), the exact normals and the closet points of Ω are provided. If Ω is given as a polygonal mesh, these two functions must be approximated from the input data, see e.g. [19, 2, 8, 38].

Our method assumes an initial surface mesh $\mathcal{T}_{init}$ of Ω is provided. It is required that all triangles in the surface mesh are oriented consistently, such that if we say the edge **ab** is the common edge of two faces **abc** and **bad**, we understand that both of the normals of **abc** and **bad** are pointing to the outwards of the surface.

We use a heuristic condition to justify geometric approximation. Intuitively, if a triangle is used to approximate a patch of a surface, the normal of the triangle should not vary too much with respect to the normals of any point in this patch. Let f be a face in the surface mesh. We say that f is *inverted* if the angle between two vectors $\mathbf{n}_f$ and $\mathbf{n_c}$ is less than a given threshold, where $\mathbf{n}_f$ is the outwards normal of f and $\mathbf{n_c}$ is the normal of surface at the closet point of the centroidal $\mathbf{c}$ of f. In other words, the angle between $\mathbf{n}_f$ and $\mathbf{n_c}$ determines if the face is inverted or not. An inverted face is considered as a bad approximation of the geometry. It is crucial to use an appropriate threshold in order to achieve best mesh quality. In our experiments in Section 5, the threshold we used is 90^o.

We found it is better to reinterpret the constant s in Φ. We consider a surface Γ and two points $A, B \in \Gamma$, we apply the map Φ and we have:

$$\Phi(A) = (x_A,\, y_A,\, z_A,\, sn_A,\, sv_A,\, sw_A)^t\,,$$
$$\Phi(B) = (x_B,\, y_B,\, z_B,\, sn_B,\, sv_B,\, sw_B)^t\,,$$

where x_A, y_A, z_A and x_B, y_B, z_B are the $\mathbb{R}^3$ coordinates of A and B, respectively, and n_A, v_A, w_A and n_B, v_B, w_B are the components of the unit normal

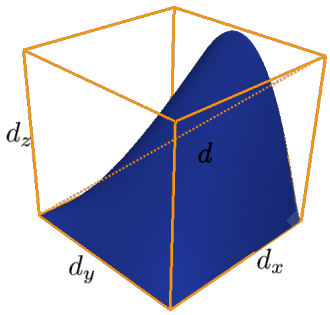

Figure 3. Bounding box of the surface Γ.

vectors at A and B, respectively. The 6d vector scalar product between these two points is

$$(A,B)_{6\mathrm{d}} = \underbrace{x_A x_B + y_A y_B + z_A z_B}_{I} + s^2 \big(\underbrace{n_A n_B + v_A v_B + w_A w_B}_{II} \big) .$$

Since the coordinates of both A and B varies in the bounding box of the surface Γ, we can say that $I \in [-d^2, d^2]$ where d is the diagonal of the bounding box, see Figure 3. Moreover, we have $n_A^2 + v_A^2 + w_A^2 = 1$ and $n_B^2 + v_B^2 + w_B^2 = 1$. We can see that the quantity $II \in [-1, 1]$. The parameter s is introduced to give more or less importance to the normals, II, on the whole value of $(A,B)_{6\mathrm{d}}$. But, since $I \in [-d^2, d^2]$ and $II \in [-1, 1]$, the contribution of I and II is unbalanced because it depends on the dimension of the bounding box. To make the quantity I and II almost comparable, we decide to modify the 6d scalar product in

$$(A,B)_{6\mathrm{d}} = x_A x_B + y_A y_B + z_A z_B + (h_\Gamma s)^2 \big(n_A n_B + v_A v_B + w_A w_B \big) .$$

where

$$h_\Gamma = \frac{d_x + d_y + d_z}{3} ,$$

and d_x, d_y and d_z are the dimension of the bounding box of Γ, see Figure 3. In this way the quantity I and II are at most comparable and the parameter s have effectively the effect to give more or less importance to the normals.

4.2 Overview of the Approach

The inputs of the algorithm are: an initial triangular mesh $\mathcal{T}_{init}$ of the surface Ω, a user-specified desired edge length L_{min} (in 6d), a user-specified minimum face angle θ_{min}, and a parameter s that specifies the desired amount of anisotropy.

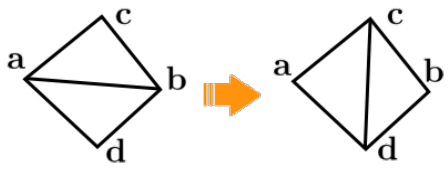

Figure 4. Edge-flip.

We initialize a mesh $\mathcal{T} := \mathcal{T}_{init}$. Then we use the map Φ to transform $\mathcal{T}$ into a surface mesh $\mathcal{T}_\Phi$ in $\mathbb{R}^6$. Our goal is to remesh $\mathcal{T}$ such that $\mathcal{T}_\Phi$ is an uniform isotropic triangular mesh in $\mathbb{R}^6$. For simplicity, we assume the map Φ is one-to-one. Therefore, we still work in $\mathbb{R}^3$, **but** we use calculated quantities (edge lengths and angles) from $\mathbb{R}^6$.

Our method works in two phases:

(i) sampling, we split edges in $\mathcal{T}$ whose lengths (measured in $\mathbb{R}^6$) are too long with respect to the given parameter L_{min} (Section 4.4); and
(ii) optimizing, we maximize the smallest face angle (measured in $\mathbb{R}^6$) such that they are not smaller than θ_{min} (Section 4.5).

The result is a curvature-adapted anisotropic triangular mesh of the surface. This method is detailed in the following subsections.

4.3 Local Mesh Modifications

Our algorithm applies a series of local surface mesh modifications directly on the mesh $\mathcal{T}$. The most well-known and commonly used local modifications are: edge-flip, edge-collapse, vertex insertion, and vertex smoothing. These operations are already extensively discussed in many papers, see e.g. [24, 23, 12, 19]. In this section, we describe how they are realized in our method.

An Edge-flip Algorithm

Edge-flip is the most efficient and effective local operation to improve simultaneously the geometry approximation and the quality of the surface mesh. Therefore, it should be applied whenever it is possible. For this purpose, we developed an edge-flip algorithm. It is inspired by the well-known Lawson's flip algorithm [31] for constructing 2d Delaunay triangulations.

An *edge-flip* on **ab** will remove the two faces **abc** and **bad** and replace them by two new faces **cdb** and **dca**. As a result, edge **ab** is replaced by edge **cd**. In our method, we want to flip **ab** if the new triangles are "better" than the old ones with regard to either geometry approximation or mesh quality.

Given an edge **ab** in $\mathcal{T}$, we check if **ab** needs to be flipped if one of the following two conditions are met:

(a) (geometric approximation) either face **abc** or face **bad** is inverted; and

```
FlipEdges(S, S1)
// S is a stack of edges to be checked and flipped; and
// S1 is another stack which is empty on input.
 (1) while S is non-empty do
 (2)    count := 0;
 (3)    while S is non-empty do
 (4)      pop ab from S;
 (5)      if ab meets condition (a) or (b), then
 (6)        if ab meets conditions (c) and (d), then
 (7)          flip ab to cd;
 (8)          for xy ∈ {ac, cb, bd, da} do;
 (9)            push xy on S;
(10)          count := count + 1;
(11)        else
(12)          push ab on S1;
(13)        endif
(14)      endif
(15)    endwhile
(16)    if S1 is non-empty and count > 0, then
(17)      swap S and S1;
(18)    endif
(19) endwhile
```

Figure 5. The edge-flip algorithm.

(b) (mesh quality) both **abc** and **bad** are not inverted and the smallest 6d-angle of the two new faces (**cdb** and **dca**) is larger than the smallest 6d-angle of **abc** and **bad**.

If an edge **ab** satisfies either (a) or (b), it implies that either face **abc** or face **bad** is bad (or both of them), it (or they) should be replaced by a better face(s). However, it is not always possible to do an edge-flip. An edge **ab** is *flippable*, if both of the following two conditions are satisfied:

(c) (topology validation) the edge **cd** does not exist in $\mathcal{T}$; and
(d) (geometry validation) neither **cdb** nor **dca** is inverted.

Condition (c) guarantees that the topology of the surface will not be changed after the flip, and (d) ensures that the new created faces are not bad approximation of the geometry.

The edge-flip algorithm is shown in Figure 5. This algorithm uses two stacks S and S_1. Initially, the stack S keeps all edges to be checked and flipped, and the stack S_1 is empty, and it will keep edges which are not flippable. Edges in S_1 are tried again if any flip has been done in the inner loop (lines 3 – 15). Once an edge is get flipped, we push the boundary edges of the new triangles into stack (lines 7 – 9). Hence flips may propagate to the neighboring edges. On return, if S_1 is not empty, it means there are unflippable edges or even inverted faces.

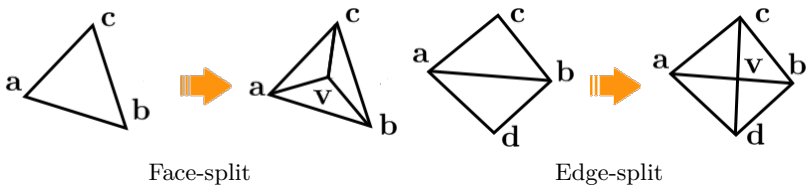

Figure 6. Vertex insertion.

A key issue for the termination of this algorithm is to show that once an edge is flipped, it will never be created again. Unfortunately, it is not yet proved. We leave it in our future work. So far, this algorithm works very well in our experiments.

Edge Collapse

Edge collapse is a common operation for simplifying meshes. An edge collapse unifies the two endpoints of the edge and two adjacent faces of this edge vanish. The unification of the two endpoints can be either one of the endpoints or a new vertex inside the cavity of adjacent faces of this edge.

For simplicity, we simply choose one of the endpoints as the new vertex by checking if there will be no inverted face at this endpoint after the edge collapse. Then we push all the link edges of this vertex into a stack, and the routine FlipEdges() is called to locally improve the mesh.

Vertex insertion

Vertex insertion is a common operation for refining meshes. Two well-known approaches for vertex insertion are the Bowyer-Watson algorithm or the incremental flip algorithm [16]. They are equivalent in generating Delaunay triangulations. We use the incremental flip algorithm for our vertex insertion in 3d surfaces.

Let **v** be a new vertex (in the surface) to be inserted. It is first located in the mesh, i.e., either a triangle **abc** or an edge **ab** of the mesh is declared to contain this vertex. We then replace the face **abc** by three new faces, **abv**, **bcv**, **cav**, see Figure 6 left; or split the edge **ab** by replacing two faces **abc**, **bad** by four faces **avc**, **vbc**, **bvd**, **vad**, see Figure 6 right. Then we put all link edges of **v** into a stack, and the routine FlipEdges() is called to improve the mesh.

Comment: Since the surface may be curved, the vertex (in the surface) is not necessary in a face or edge in current mesh. Vertex location in a 3d surface mesh is a difficult problem. We avoid this problem in our method by always choosing the midpoint of an edge, and then a new vertex is obtained by snapping this point to the surface. Hence, we can declare that the edge to be split contain this new vertex.

```
SAMPLING(Ω, T, Q, L_min)
// Q is a queue of triangles in T;
(1) while Q is non-empty do
(2)    pop a face f from Q;
(3)    Let e be the longest edge of f;
(4)    if ‖e‖_6d > 1.5 L_min, then
(5)       split e by adding v ∈ Ω into T;
(6)       update Q;
(7)    endif
(8) endwhile
```

Figure 7. The sampling algorithm.

Vertex smoothing

For a given vertex **v**, the vertex smoothing operation consists in finding a new location for **v** such that the local mesh quality is improved without changing the mesh topology.

Our approach for vertex smoothing is very basic, somewhat similar to the Laplace smoothing. Let **abv** be one of the faces connecting at **v**, we simply move **v** towards the barycenter of **abv** with a relaxation parameter. If the new location of **v** improves the local mesh quality, we actually do this move. After **v** is moved, we push all the link edges at **v** into a stack, and the routine FLIPEDGES() is called.

4.4 Sampling

The purpose of sampling is to achieve the desired mesh size with respect to the given 6d-length parameter L_{min}. Our strategy is straightforward, splitting edges of $\mathcal{T}$ which 6d-lengths is too long compared to L_{min}. For simplicity, each long edge is split by adding the point in the surface which is closet to the midpoint of the edge. Also, we do not collapse short edges in this phase, it will be applied in the optimization phase.

The sampling algorithm is shown in Figure 7. It initializes a queue Q which contains all triangles in $\mathcal{T}$. It then works in a loop until Q is empty. On each face f popped from Q, it checks the longest edge e of f and split it if the 6d-length of e is too long (lines $4-7$). Then $\mathcal{T}$ is updated (line 6) by removing old faces and adding new faces.

Inserting a new vertex into a 3d surface mesh may deform it very much. This is particularly the case when the surface mesh is only a very crude approximation of the original geometry. Recall that our vertex insertion routine will automatically improve the local mesh by FLIPEDGES(). Our experiments (shown in Section 5) showed that this edge flip algorithm is very effective in improving both of the geometry approximation and the mesh quality.

OPTMIZING(Ω, $\mathcal{T}$, L_{min}, θ_{min}, I, J, K)
// I, J, and K are user-specified iterations.
(0) $\theta_{max} := 180^o - 2 * \theta_{min}$;
(1) Collapse too short edges with respect to L_{min}.
(2) **for** $i \in \{1, ..., \mathrm{I}\}$ **do**
(3) **for** $j \in \{1, ..., \mathrm{J}\}$ **do**
(4) **for** $k \in \{1, ..., \mathrm{K}\}$ **do**
(5) Smooth all vertices;
(6) **enfor**
(7) Collapse edges for removing angles $< \theta_{min}$;
(8) **enfor**
(9) Split edges for removing angles $>= \theta_{max}$;
(10) **endfor**

Figure 8. The optimizing algorithm.

4.5 Optimizing

Since the sampling phase has removed long edges, the goal of the optimizing phase is on maximizing the smallest 6d-angle of triangles in the mesh $\mathcal{T}$. We used the following combination of these operations shown in Figure 8.

Mesh optimization is performed by iteratively combining a series of local modifications, which are, edge-flips, vertex smooth, edge-collapse, and vertex insertion:

- vertex smoothing iteratively moves the positions of vertices (stay on surface) such that the minimum 6d-angle is improved around each vertex;
- edge collapse is used to remove small angles $< \theta_{min}$, it iteratively removes the edges opposite small angles;
- edge split is used to remove large angles $> \theta_{max}$, it iteratively splits the edges opposite large angles by inserting a vertex close to the midpoints of the edges.

We call the routine FLIPEDGES() called within each local operations to improve local mesh quality.

4.6 Sharp Features

Our method can be easily adapted to mesh non-smooth surfaces, i.e., surfaces containing edges and corner singularities and sharp features. These features are commonly presented in complicated geometries. Sharp features can be preserved as follows. We consider a surface Γ, for example a geometry coming from a CAD model. The whole geometry is the sum of a finite set of patches that are joint together along their common boundaries, see Figure 9, i.e.

$$\Gamma = \bigcup_{i=1}^{n} \Gamma_i \,, \quad \text{and} \quad \Gamma_i \cap \Gamma_j = \begin{cases} \emptyset \\ \gamma_{ij} \end{cases} , \tag{1}$$

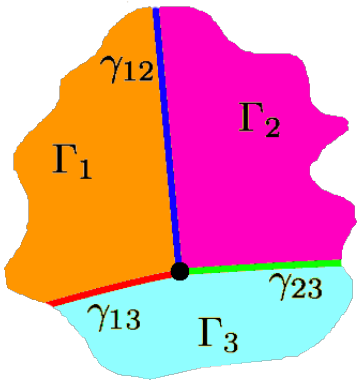

Figure 9. Example of a domain with sharp features.

where γ_{ij} is the common line between Γ_i and Γ_j, if it exists.

The proposed method will re-mesh each sub-surface Γ_i separately, like the implicit surfaces of Subsection 5.1. In the routine FLIPEDGES(), an edge belong to a sharp feature will never be flipped.

5 Examples

In this section, several examples are presented to demonstrate the proposed method. Firstly, in Subsection 5.1, we re-mesh some very simple cases to ensure that this method works. In particular, we consider surfaces in which we could predict where and how the triangles will be stretched. Then we experimentally verify the reliability of the proposed re-meshing method by using some implicit functions which have very steep jumps, and using very bad-quality initial meshes.

In Subsection 5.2, we consider some complicated CAD data to show that this method can be used to re-mesh complicated geometries and it does **preserve** sharp features.

The proposed method aims at optimizing the mesh to make it as uniform as possible in $\mathbb{R}^6$, the best triangles in the resulting mesh should have its smallest 6d-angle as close as to $60°$. In all the example meshes, we use a color-map to highlight the 6d measure of the smallest angle for each triangle.

5.1 Implicit Surfaces

Example 1: We consider the disk, defined by the zero level set of the following function:

$$f_1(x\,,y\,,z) := \left(\frac{x}{0.8}\right)^2 + \left(\frac{y}{0.8}\right)^2 + \left(\frac{z}{0.4}\right)^2 - 1\,. \tag{2}$$

In Figure 10, we highlight some zones and we indicate how the triangles will look like after the re-meshing procedure.

Since there is a very big change of curvature along one direction $\overrightarrow{v}$, see zone A in Figure 10, we expect that the triangles will be stretched along $\overrightarrow{w}$, i.e. the perpendicular direction that lies on the tangent plane of the surface. Moreover we expect equilateral triangles in the zone B, where the surface is more smooth and flat.

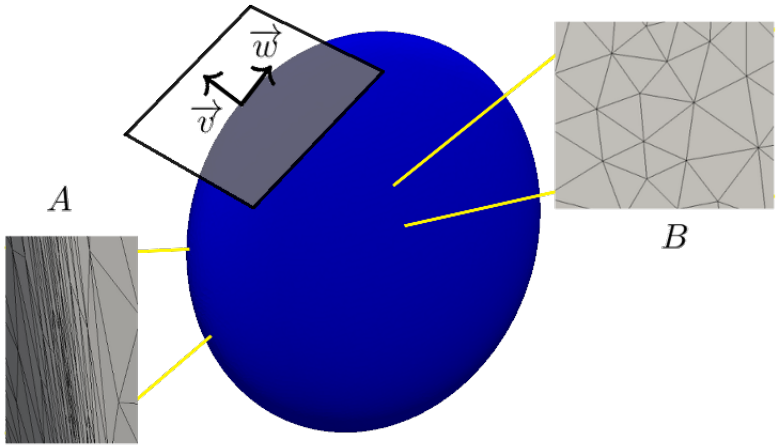

Figure 10. The zero level set of the function f_1, some zones are highlighted to show how the adapted mesh will look like.

In Figure 11 we show both the initial and the final mesh. The shapes of triangles in the resulting mesh are those as we have expected. More details of the mesh are shown in Figure 12 and 13.

Example 2: We consider the sinusoidal surface, defined by the zero level set of the following function:

$$f_2(x\,,y\,,z) := \sin(\pi x)\sin(\pi y) - z\,. \tag{3}$$

In Figure 14 we show the surface and the zones of interest. In particular we notice that there are a lot of regions where the mesh has to be isotropic, see the zones A and B in Figure 14.

In Figure 15 we show both the initial and the final mesh. It is interesting to see that in the zones like A and B, see Figure 14 and 16, where the mesh is smooth, we do **not** have any anisotropy. Moreover we notice how this method "solve" the zone where we move from an isotropic mesh to an anisotropic one, see Figure 17: from the center, where the triangles are isotropic, they progressively becomes aligned to the surface features, i.e. they progressively

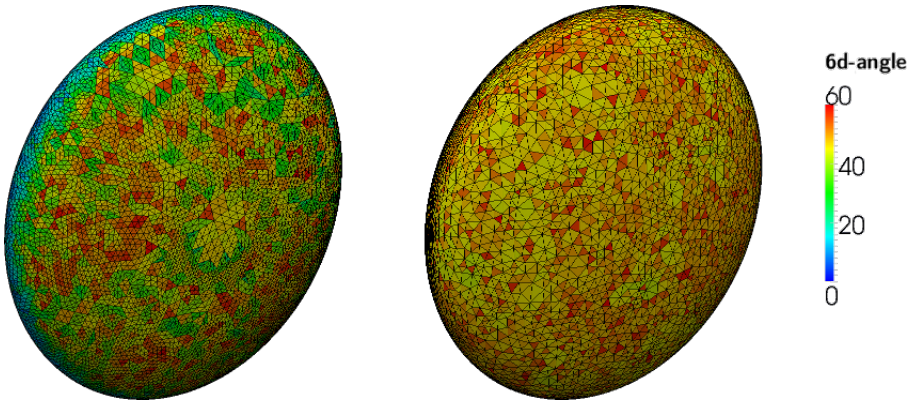

Figure 11. the initial mesh (left) and the optimized mesh (right).

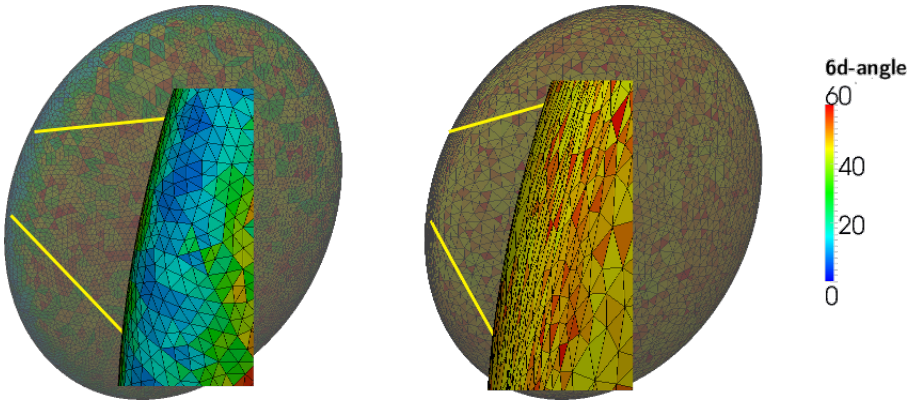

Figure 12. Detail of the initial mesh (left) and the optimized mesh (right).

becomes anisotropic due to the curvature of the surface. In Figure 12 this behavior is shown and the arrows point out the optimal orientation of the triangles.

Example 3: Now consider the following function

$$f_3(x\,,y\,,z) := \tanh\left(20x\right) - \tanh\left(20(x-y)-10\right) - z\,. \tag{4}$$

The zero level set of such a function is a surface that presents a smart change of curvature, see zones A and B in Figure 18, and it is flat in others, see zone C and D in Figure 18. This behavior allows the re-meshing procedure to create a very stretched elements in the zones like A and B, and isotropic triangles in the zones like C and D.

In this example, we stat from a very bad approximation mesh obtained by a marching cube method. As we can easily see in Figures 20, and 21, the initial

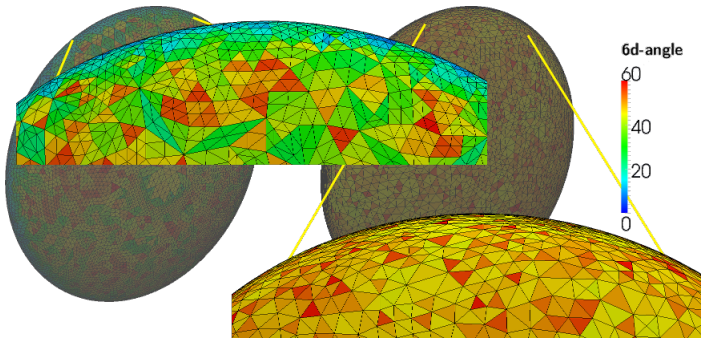

Figure 13. Detail of the initial mesh (left) and the optimized mesh (right).

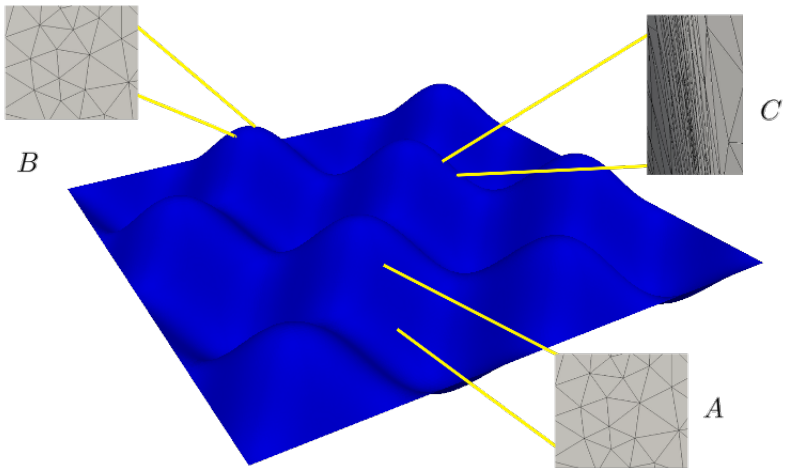

Figure 14. The zero level set of the function f_2, some zones are highlighted to show how the adapted mesh will look like.

mesh does not give a very good approximation of the surface: the triangles are not oriented in the right way, there is an over sampling and the triangles are somewhere really far away form the real geometry. This challenges the robustness of the proposed method.

From Figures 19, 20, and 21, we could appreciate that the geometry is really well fitted by the resulting mesh. In particular, from Figure 20, the initial mesh entirely change and both **shape** and **orientation** of the elements help to fit the geometry as well as possible.

In Figure 21 we have a similar configuration of the one proposed by the previous example, see Figure 17: a flat region surrounded by zones where the

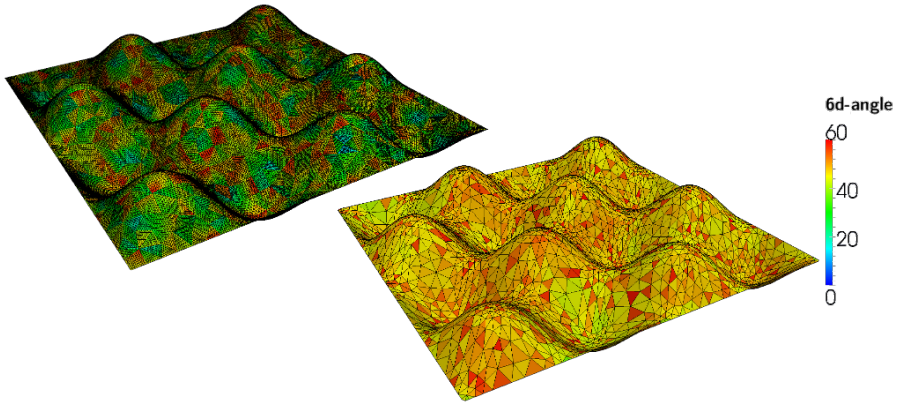

Figure 15. The initial mesh (left) and the optimized mesh (right).

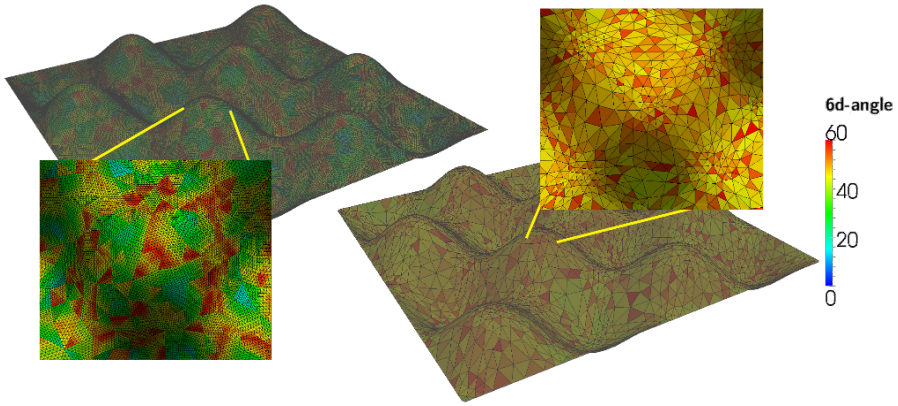

Figure 16. Detail of the initial mesh (left) and the optimized mesh (right).

curvature change. But in this example the curvature do not have the smooth behavior of the one of Example 2, so we have more stretched elements.

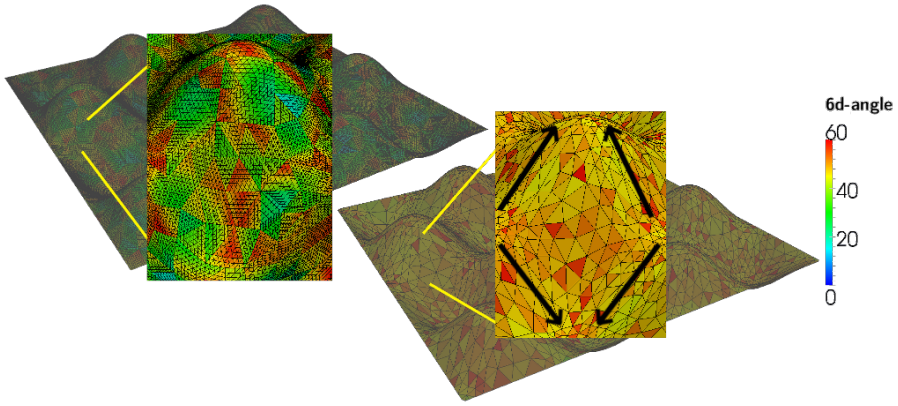

Figure 17. Detail of the initial mesh (left) and the optimized mesh (right).

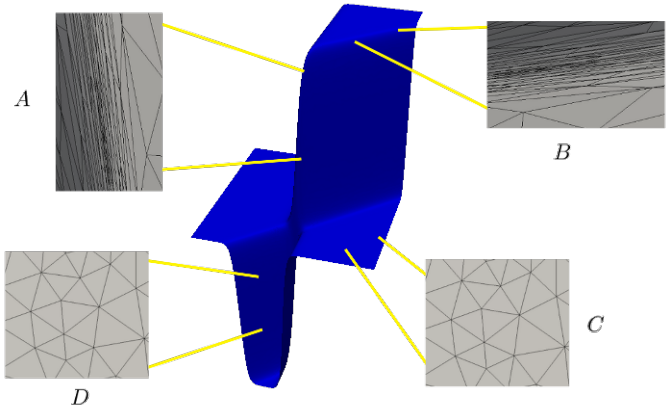

Figure 18. The zero level set of the function f_3, some zones are highlighted to show how the adapted mesh will look like.

Example 4: In all the previous examples we have considered the same s factor to proceed with the re-meshing algorithm. Now we try to change this value and see what it would be the effect on the resulting mesh. We define the function,

$$f_4(x\,,y\,,z) := \tanh\left(20y\right) - \tanh\left(20(x-y)-10\right) - z\,, \tag{5}$$

whose zero level set is the surface represented in Figure 22. This surface presents some flat regions, like the zone A in Figure 22, and a series of very deep jumps, see the arrows in Figure 22.

We **fix** a desired 6d-length and we consider these values of the parameter $s = 0.1, 1., 5., 25.$.

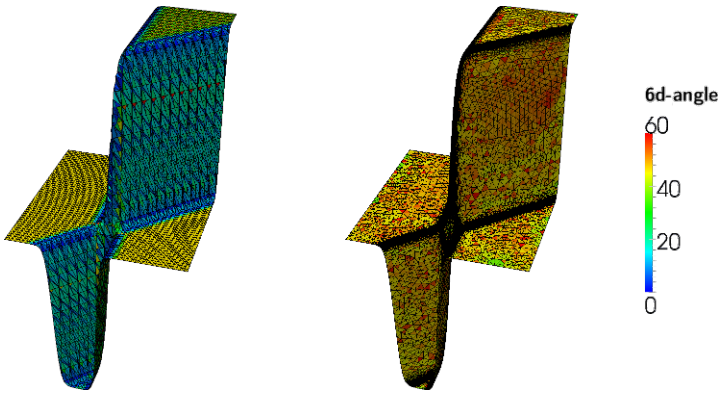

Figure 19. The initial mesh (left) and the optimized mesh (right).

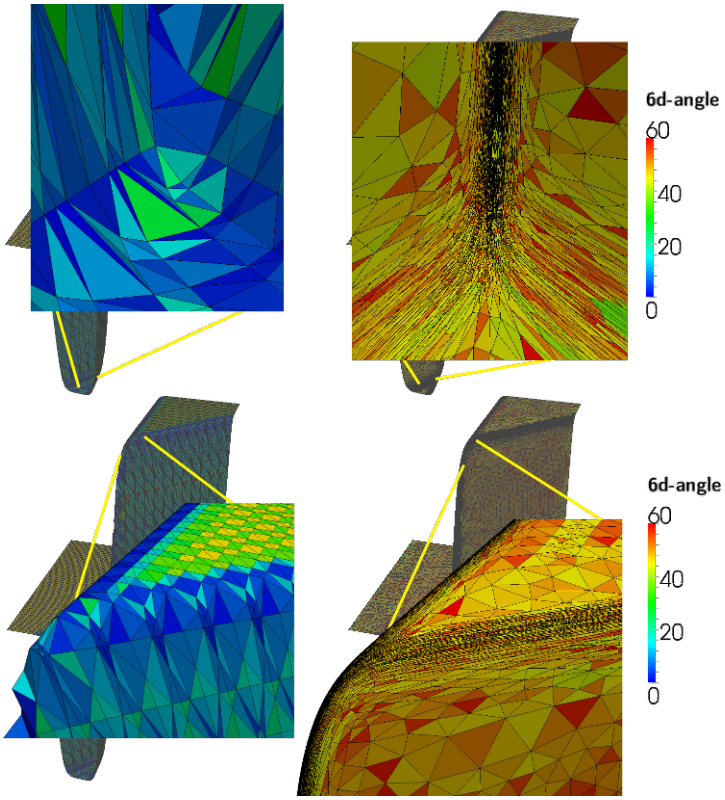

Figure 20. Detail of the initial mesh (left) and the optimized mesh (right).

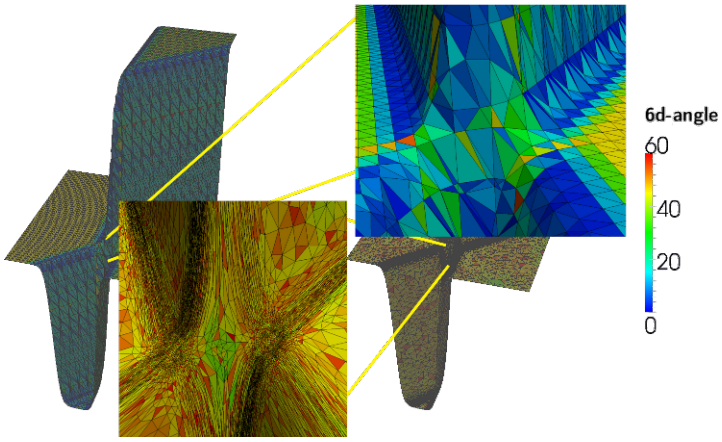

Figure 21. Detail of the initial mesh (left) and the optimized mesh (right).

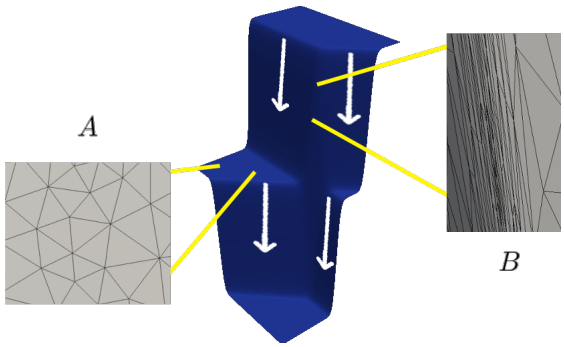

Figure 22. The zero level set of the function f_4, some zones are highlighted to show how the adapted mesh will look like.

Increasing the factor s, will increase the length of the edges of the mesh, **but** this fact it is not completely true. The **real** effect of increasing the parameter s is to **empathize** the variation of the normal, i.e. the variation of curvature. In fact, where the surface is flat, the size of the mesh elements is the same for each values of s. Instead, where it presents a variation of the curvature, it is more and more refined. When we are dealing with big values of s, small variation of the normals will correspond to large variation of the edge length, so these edges should be refined.

Figure 24 clearly shows this behavior. Nearby the jump we have a flat region, then there is a very sharp change of curvature and, finally, a region where the curvature of the surface varies very slowly. The resulting mesh does not vary with the different values of s, but the slow variation of the normals will be more empathize with higher value of s, see Figure 25.

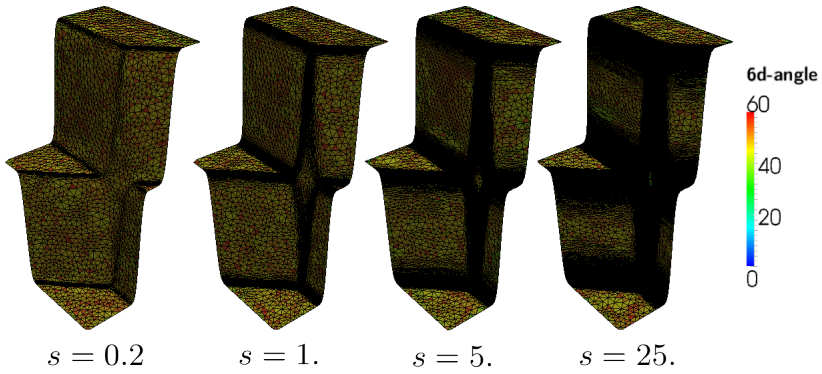

Figure 23. Meshes with different values of the parameter s.

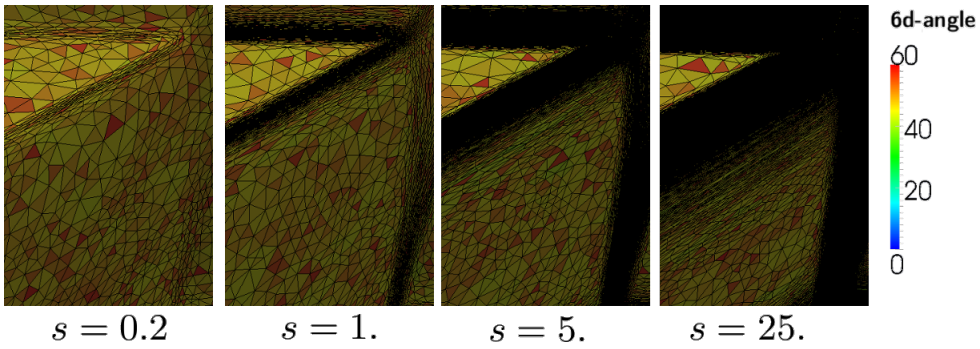

Figure 24. Detail of the optimized meshes in Figure 23.

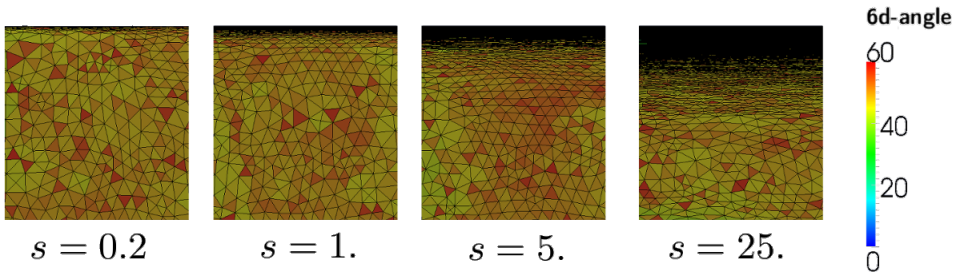

Figure 25. Detail of the deep jump of the optimized meshes in Figure 23.

5.2 Sharp Features

In this subsection, we consider CAD models as input, and to show that our method is able to preserve sharp features and to mesh complicated geometries. We used the `Gmsh` library [21]. It conveniently provided the functionalities (1)

and (2) described in Subsection 4.1 for CAD models and it builds an initial surface mesh from them.

Example 1: The method of Lévy and Bonneel [32] does not preserve the sharp features, i.e. they are oversampled and smoothed. We consider one of the CAD model in this paper and we apply the proposed re-meshing procedure to make a comparison between these two methods.

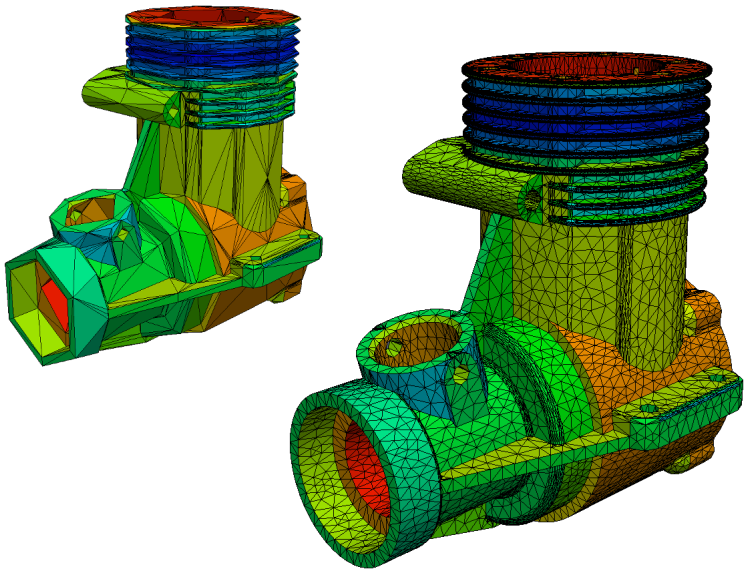

Figure 26. The initial mesh (left) and the optimized mesh (right).

In Figures 27, we compared the resulting mesh of the method in [32] and the same detail obtained by our method. Sharp features are **preserved** and they are not both smoothed and oversampled.

Example 2: Here we compared another example given in [32], where the sharp features are not preserved.

Example 3: The proposed re-meshing procedure does not have any problem in the reconstruction of really complex geometries with details of different measure respect to the size of the whole geometry and different shape.

Example 4: This example shows another complicated CAD surface remeshed by our method.

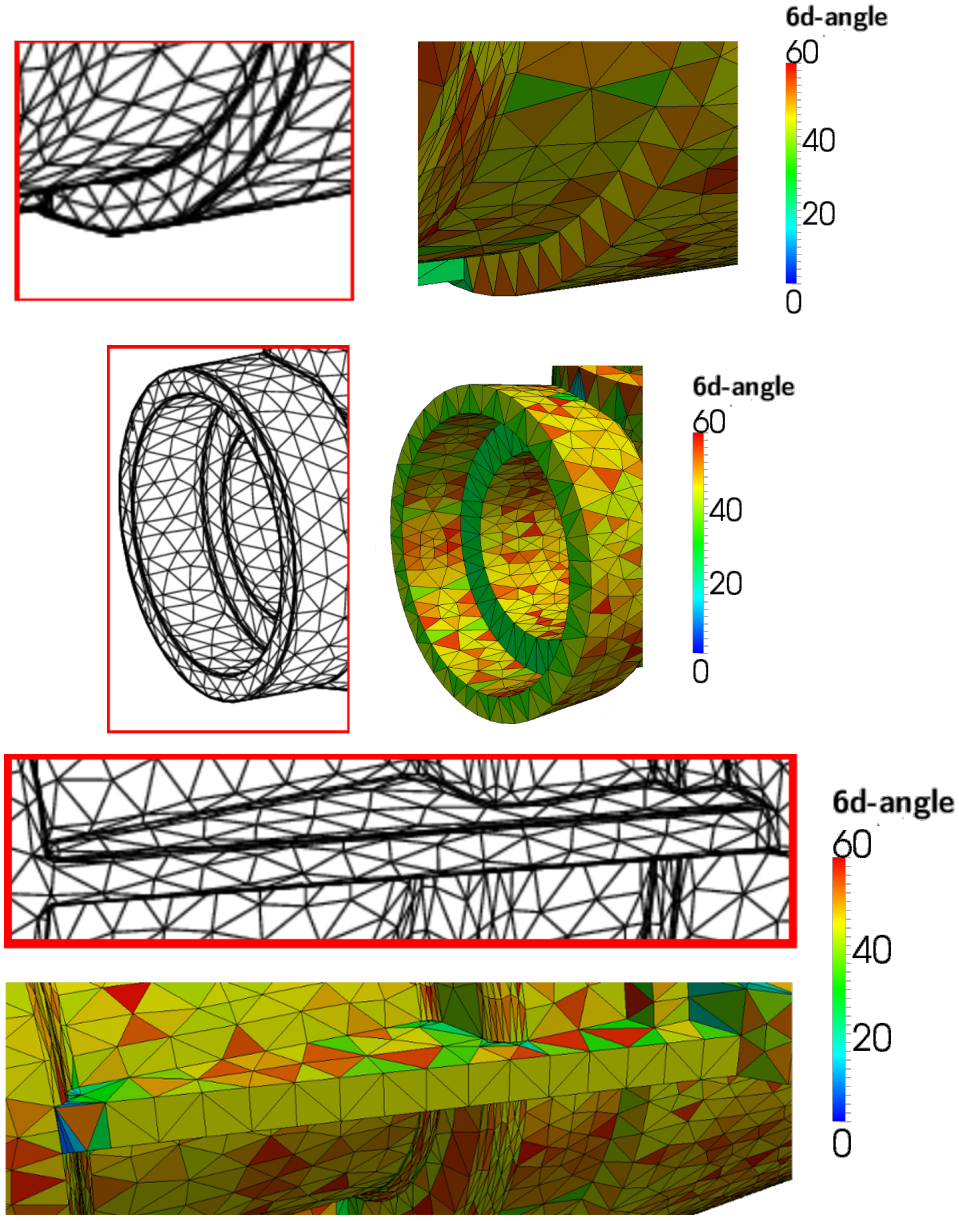

Figure 27. Detail of the mesh proposed in [32] (shown in red boxes) and the same detail of the mesh obtained by our method (colored).

6 Conclusion and Future Works

In this paper, we presented a curvature-adapted anisotropic surface remeshing method. It is based on the idea of high-dimensional embeddings of surfaces.

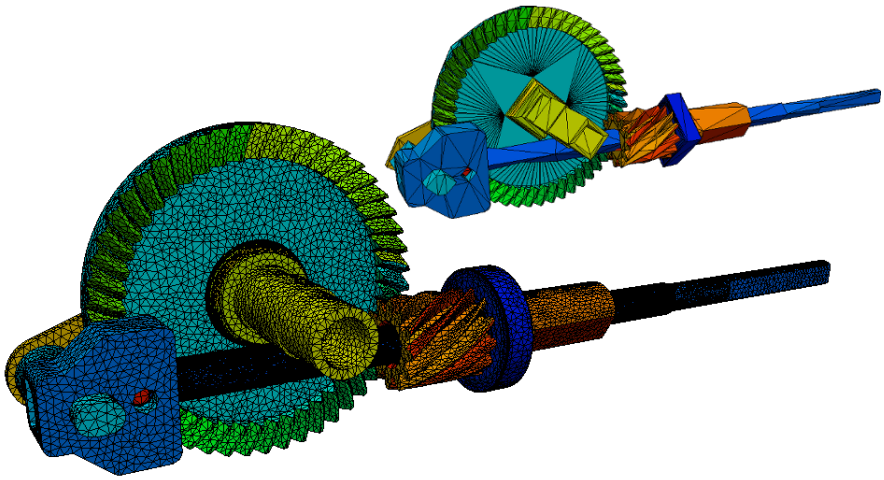

Figure 28. The initial mesh (left) and the optimized mesh (right).

This method is in principle simple. It fits the well-developed mesh adaptation framework. While it has several advantages. For instances, sharp features are always respected; it is robust in handling strong anisotropy, and it is easy to implement. Our experimental results showed that this method is able to produce good meshes for various 3d surfaces which may have arbitrary complicated geometry and may contain strong anisotropy.

There are many questions which are widely open. A very important theoretical question is: How well could this mapping Φ approximate the geodesic distances in 3d surfaces? Are there upper or lower bounds on distance variations by this mapping? A theoretical study of these question could lead to more efficient methods, and will result much smaller mesh size.

The edge-flip algorithm we described in this paper appears very useful in improving both geometry approximation and mesh quality. However, its termination is not yet proven. We found that the threshold angle for checking inverted faces is very crucial. A good value will give edge-flip more freedom and may produce highly stretched triangles.

In practice, many surfaces are given as a polygonal mesh, i.e., the original geometry is not available. A good recovery and estimation of the surface normals are necessary in order to achieve good results. We plan to implement such feature into our code.

The running time of our implementation is far from optimal. There are many possibilities to improve it.

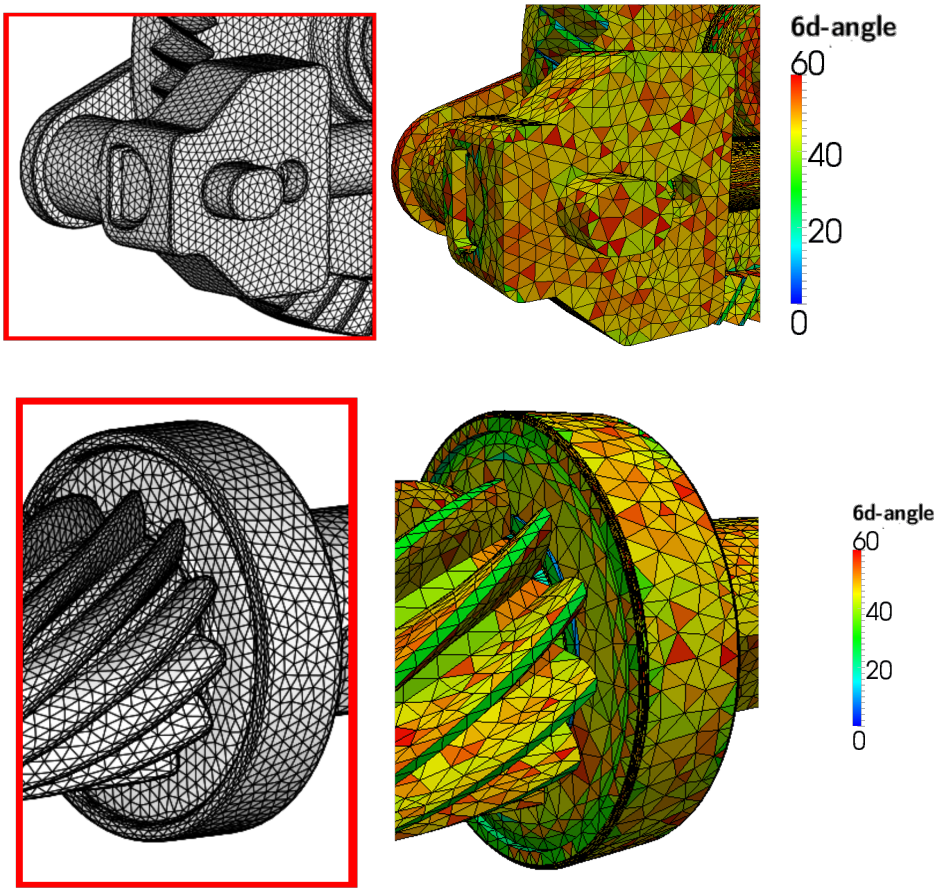

Figure 29. Detail of the mesh proposed in [32] (shown in red boxes) and the same detail of the mesh obtained by our method (colored).

References

1. P. Alliez, M. Attene, C. Gotsman, and G. Ucelli. Recent advances in remeshing of surfaces. In D. L. and M. Spagnuolo, editors, *Shape Analysis and Structuring*, pages 53–82. Springer, 2008.
2. P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 485–493, New York, NY, USA, 2003. ACM.
3. P. Alliez, E. C. d. Verdière, O. Devillers, and M. Isenburg. Isotropic surface remeshing. In *Proceedings of the Shape Modeling International 2003*, SMI '03, pages 49–, Washington, DC, USA, 2003. IEEE Computer Society.

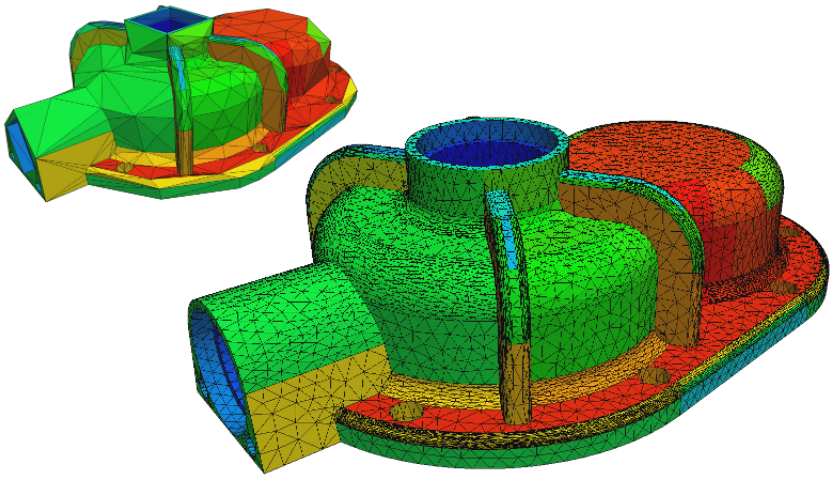

Figure 30. The initial mesh (left) and the optimized mesh (right).

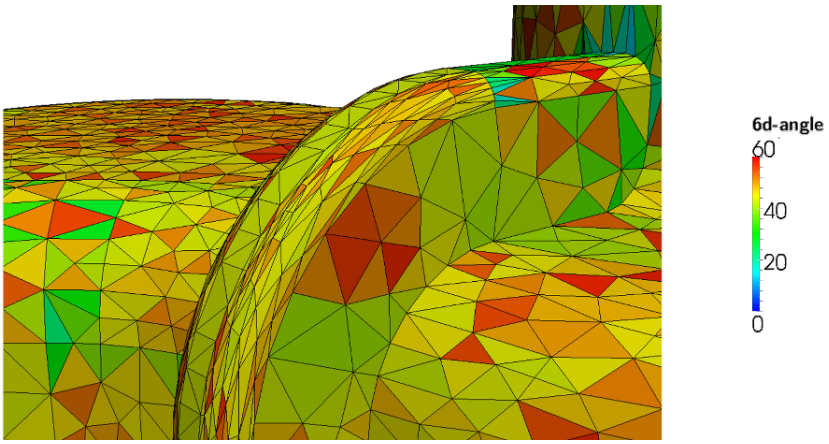

Figure 31. Detail of the mesh represented in Figure 30.

4. F. Aurenhammer. Voronoi diagrams – a study of fundamental geometric data structures. *ACM Comput. Surveys*, 23:345–405, 1991.
5. J. D. Boissonnat, K.-L. Shi, J. Tournois, and Yvinec. Anisotropic Delaunay meshes of surfaces. To appear in ACM Transactions on Graphics, 2012.
6. J. D. Boissonnat, C. Wormser, and M. Yvinec. Locally uniform anisotropic meshing. In *Proc. 24th Ann. Symp. on Comput. Geom.*, 2008.
7. G. D. Canas and S. J. Gortler. Surface remeshing in arbitrary codimensions. *Vis. Comput.*, 22(9):885–895, Sept. 2006.
8. G. D. Canas and S. J. Gortler. Shape operator metric for surface normal approximation. In *Proc. 18th International Meshing Roundtable*, 2009.

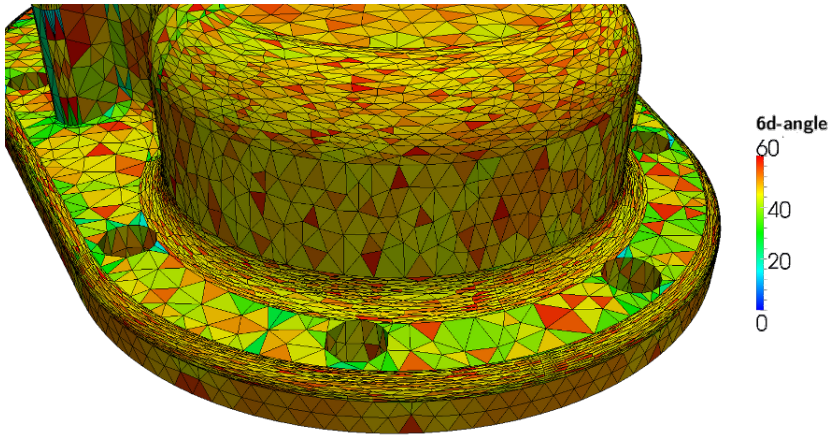

Figure 32. Detail of the mesh represented in Figure 30.

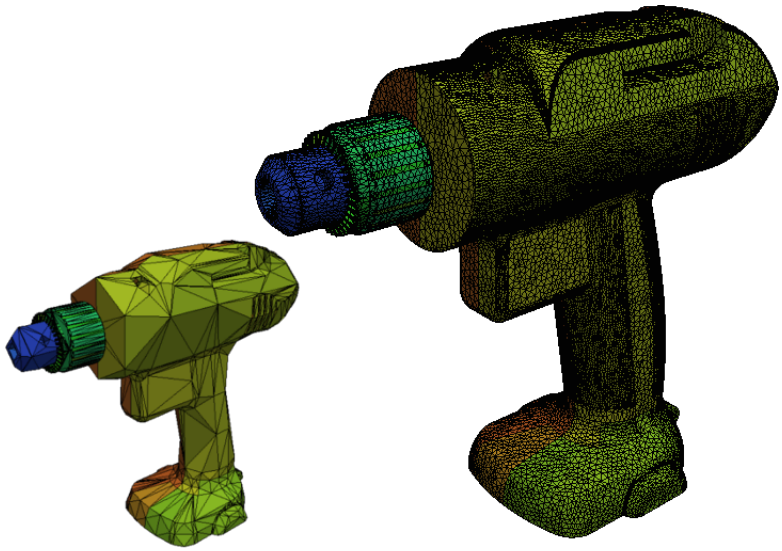

Figure 33. The initial mesh (left) and the optimized mesh (right).

9. L. Chen, P. Sun, and J. Xu. Optimal anisotropic meshes for minimizing interpolation errors in l^p-norm. *Mathematics of Computation*, 76:179–204, 2007.
10. S.-W. Cheng, T. K. Dey, and J. A. Levine. A practical Delaunay meshing algorithm for a large class of domains. In *Proc. 16th International Meshing Roundtable*, pages 477–494. Sandia National Laboratories, 2007.
11. S.-W. Cheng, T. K. Dey, and E. A. Ramos. Anisotropic surface meshing. In *Proc. ACM-SIAM Sympos. Discrete Algorithms*, pages 202–211, 2006.

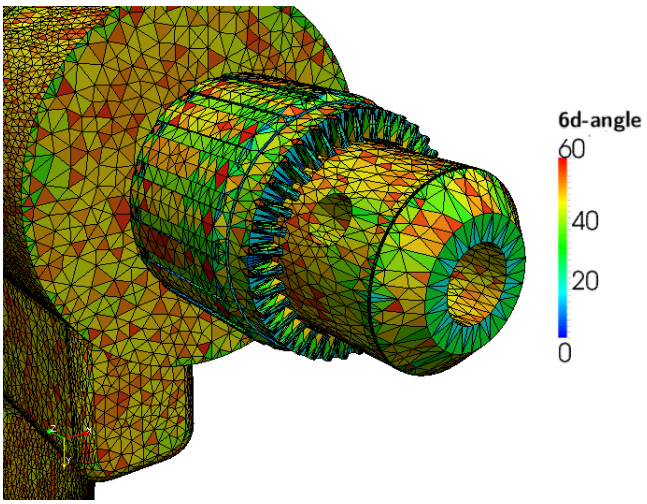

Figure 34. Detail of the mesh represented in Figure 33.

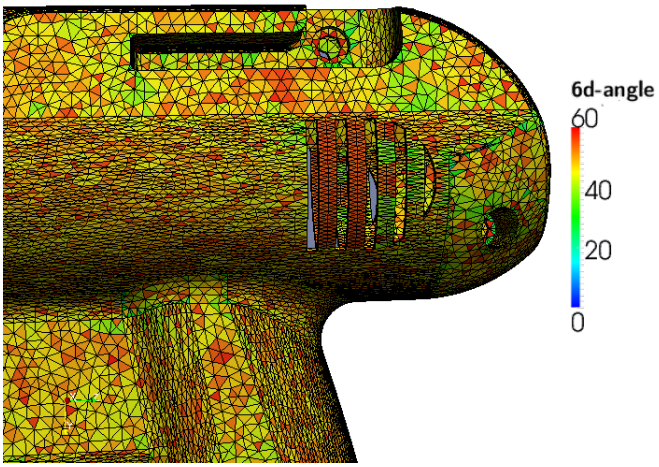

Figure 35. Detail of the mesh represented in Figure 33.

12. H. de Cougny and M. S. Shephard. Surface meshing using vertex insertion. In *In Proceedings of the 5th International Meshing Roundtable*, pages 243–256, 1996.
13. C. Dobrzynski and P. Frey. Anisotropic Delaunay mesh adaptation for unsteady simulations. In *17th International Meshing Roundtable*, 2008.
14. Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Rev.*, 41(4):637–676, Dec. 1999.
15. Q. Du and D. Wang. Anisotropic centroil Voronoi tessellations and their applications. *SIAM J. Sci. Comput.*, 26(3):737–761, 2005.

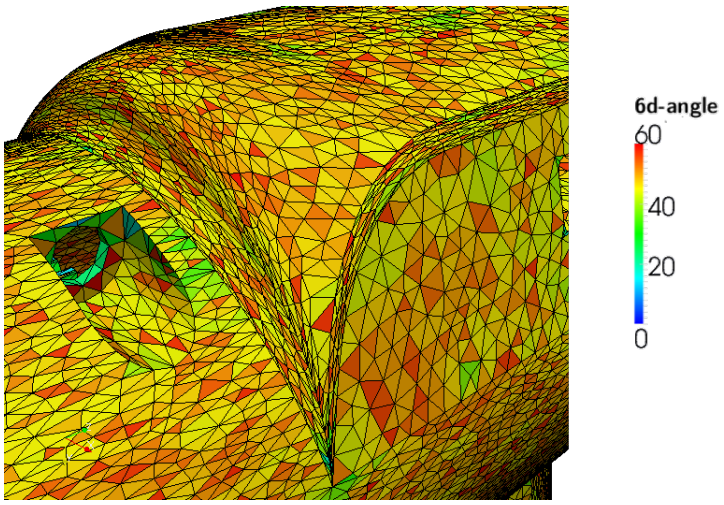

Figure 36. Detail of the mesh represented in Figure 33.

16. H. Edelsbrunner. *Geometry and topology for mesh generation*. Cambridge University Press, Cambridge, England, 2001.
17. H. Edelsbrunner and N. R. Shah. Triangulating topological spaces. *International Journal of Computational Geometry & Applications*, 7(4):365–378, 1997.
18. P. Frey and F. Alauzet. Anisotropic mesh adaption for CFD computations. *Comput. Methods Appl. Mech. Engrg.*, 194:5068–5082, 2005.
19. P. J. Frey and H. Borouchaki. Geometric surface mesh optimization. *Computing and Visualization in Science*, 1(3):113–121, 1998.
20. P. J. Frey and P. George. *Mesh Generation - Application to Finite Elements*. Hermes Science Publishing, Oxford, UK, 1st edition, 2000. ISBN 1-903398-00-2.
21. C. Geuzaine and J. F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *Int. J. Numer. Meth. Engng.*, 79:1309–1331, 2009. `http://www.geuz.org/gmsh/`.
22. P. S. Heckbert and M. Garland. Optimal triangulation and quadric-based surface simplification. *Computational Geometry*, 14(1-3):49 – 65, 1999.
23. H. Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 99–108, New York, NY, USA, 1996. ACM.
24. H. Hoppe, T. DeRose, D. Tom, M. John, and W. Stuetzle. Mesh optimization. Technical Report 93-01-01, Department of Computer Science & Engineering, University of Washington, Seattle, Washington 98195, 1993.
25. W. Huang. Metric tensors for anisotropic mesh generation. *Journal of Computational Physics*, 204:633–665, 2005.
26. X. Jiao, A. Colombi, X. Ni, and J. C. Hart. Anisotropic mesh adaptation for evolving triangulated surfaces. In *Proc. 16th International Meshing Roundtable*, 2006.
27. R. Kimmel, R. Malladi, and N. Sochen. Images as embedded maps and minimal surfaces: Movies, color, texture, and volumetric medical images. *International Journal of Computer Vision*, 39(2):111–129, 2000.

28. D. Kovacs, A. Myles, and D. Zorin. Anisotropic quadrangulation. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, SPM '10, pages 137–146, New York, NY, USA, 2010. ACM.
29. F. Labelle and J. Shewchuk. Anisotropic Voronoi diagrams and guaranteed-quality anisotropic delaunay mesh generation. In *In Proc. 19th Annual Symposium on Computational Geometry*, pages 191–200, 2003.
30. Y.-K. Lai, Q.-Y. Zhou, S.-M. Hu, J. Wallner, and H. Pottmann. Robust feature classification and editing. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):34–45, Jan. 2007.
31. C. L. Lawson. Software for c^1 surface interpolation. *Mathematical Software III, Academic Press*, pages 164–191, 1977.
32. B. Lévy and N. Bonneel. Variational anisotropic surface meshing with voronoi parallel linear enumeration. In *Proc. 21st International Meshing Roundtable*, pages 349–366, 2012.
33. B. Lévy and Y. Liu. L_p centroidal voronoi tesselation and its applications. *ACM Transactions on Graphics (SIGGRAPH conference proceedings)*, 2010. patent pending - FR 10/02920 (filed 07/09/10).
34. Y. Liu, W. Wang, B. Levy, F. Sun, D. M. Yan, L. Lu, and C. Yang. On centroidal voronoi tessellation - energy smoothness and fast compu tation. *ACM Transactions on Graphics*, 2009.
35. A. Loseille and F. Alauzet. Continuous mesh framework part I: well-posed continuous interpolation error. *SIAM J. Numer. Anal.*, 49(1):38–60, 2011.
36. A. Loseille and F. Alauzet. Continuous mesh framework part II: validations and applications. *SIAM J. Numer. Anal.*, 49(1):61–86, 2011.
37. J.-M. Mirebeau. Optimally adapted meshes for finite elements of arbitrary order and $w^{1,p}$ norms. *Numerische Mathematik*, 120:271–305, 2012.
38. S. J. Owen, D. R. White, and T. J. Tautges. Facet-based surfaces for 3d mesh generation. In *Proc. 11th International Meshing Roundtable*, 2002.
39. H. Pottmann, T. Steiner, M. Hofer, C. Haider, and A. Hanbury. The isophotic metric and its application to feature sensitive morphology on surfaces. In T. Pajdla and J. Matas, editors, *Computer Vision - ECCV 2004*, volume 3024 of *Lecture Notes in Computer Science*, pages 560–572. Springer Berlin Heidelberg, 2004.
40. S. Rippa. Long and thin triangles can be good for linear interpolation. *SIAM Journal on Numerical Analysis*, 29(1):257–270, 1992.
41. J. R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *Proceedings of 11th International Meshing Roundtable*, pages 115–126, Ithaca, New York, September 2002. Sandia National Laboratories.
42. Z. Zhong, X. Guo, W. Wang, B. Lévy, F. Sun, Y. Liu, and W. Mao. Particle-based anisotropic surface meshing. *ACM Transactions on Graphics (SIGGRAPH conference proceedings)*, 2013.