Gmsh's Approach to Robust Mesh Generation of Surfaces with Irregular Parametrizations

Jean-François Remacle and Christophe Geuzaine

Abstract This paper proposes a robust and effective approach to overcome a major difficulty associated to surface finite element mesh generation: the handling surfaces with irregular (singular) parametrizations such as spheres, cones or other surfaces of revolution produced by common Computer Aided Design tools. The main idea is to represent triangles incident to irregular points as trapezoids with one degenerated edge. This new approach has been implemented in Gmsh and examples containing thousands of surfaces with irregular points are presented at the end of the paper.

1 Introduction

Computer Aided Design (CAD) systems are used extensively for industrial design in many domains, including automotive, shipbuilding, and aerospace industries, industrial and architectural design, prosthetics, and many more. Engineering designs are encapsulated in such CAD models, which up to manufacturing tolerances exactly represent their geometry. While the engineering analysis process begins with such CAD models, the predominant method of analysis (the finite element method) requires an alternative, discrete, representation of the geometry: a finite element mesh. In such a mesh, the CAD model is subdivided into a (large) collection of simple geometrical shapes such as triangles, quadrangles, tetrahedra and hexahedra, arranged in such a way that if two of them intersect, they do so along a face, an edge or a node, and never otherwise.

J.-F. Remacle (✉)
Institute of Mechanics, Materials and Civil Engineering (iMMC), Université catholique de Louvain, Louvain-la-Neuve, Belgium
e-mail: jean-francois.remacle@uclouvain.be

C. Geuzaine
Department of Electrical Engineering and Computer Science, Montefiore Institute B28, Université de Liège, Liège, Belgium
e-mail: c.geuzaine@uliege.be

R. Sevilla et al. (eds.), *Mesh Generation and Adaptation*, SEMA SIMAI Springer Series 30, https://doi.org/10.1007/978-3-030-92540-6_5

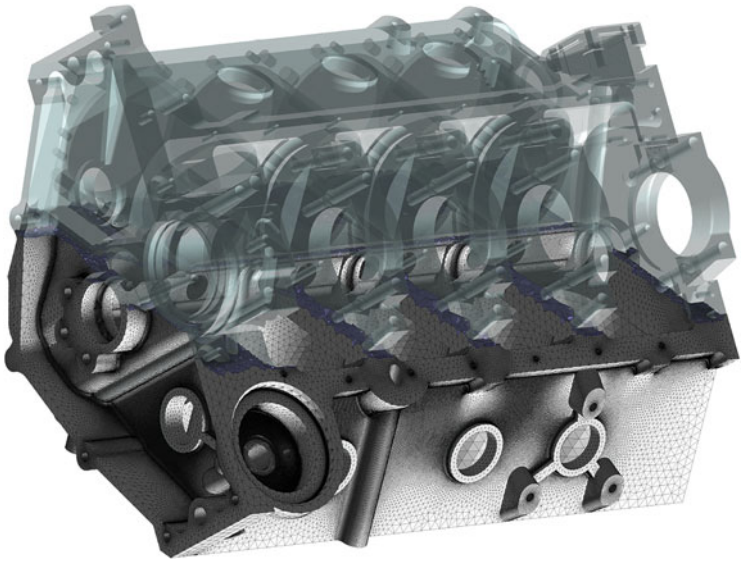

Fig. 1 An engine block

Three-dimensional CAD models are represented on a computer using a "Boundary Representation" (BRep) [1]: a volume is bounded by a set of faces, a face is bounded by a serie of curves and a curve is bounded by two end points. The BREP is a discrete object: it is a graph that contains model entities together with all their topological adjacencies. Then a geometry is associated to each model entity. Figure 1 presents a moderately complex CAD model together with its 3D mesh generated using Gmsh [2].

As an example, consider a model face F with its boundary

$$\partial F = \{C_1, \ldots, C_n\}.$$

Face F is topologically closed, i.e. $\partial(\partial F) = \emptyset$: each endpoint of the bounding curves C_j is considered twice in F, one time positively and one time negatively. The geometry of a model face F is its underlying surface $\mathcal{S}$ with its parametrization

$$\mathbf{x} \; : \; A \mapsto \mathbb{R}^3, \quad (u, v) \mapsto \mathbf{x}(u, v)$$

where $A \subset \mathbb{R}^2$ is a rectangular region $[u_0, u_1] \times [v_0, v_1]$. A parametrization is said to be regular if $\partial_u \mathbf{x}$ and $\partial_v \mathbf{x}$ are linearly independent:

$$\partial_u \mathbf{x} \times \partial_v \mathbf{x} \neq \mathbf{0}$$

for any $u, v \in A$. Points where $\partial_u \mathbf{x} \times \partial_v \mathbf{x} = \mathbf{0}$ are called irregular or singular points of the parametrization. We assume here that irregular points are isolated. Irregular

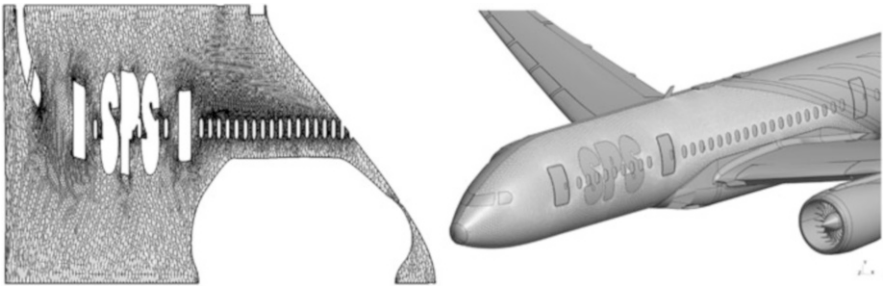

Fig. 2 Surface mesh of a model face. View of the mesh in the parameter plane (left) and on $\mathbb{R}^3$

points can occur for two possible reasons: (i) one of the partial derivatives $\partial_u \mathbf{x}$ or $\partial_v \mathbf{x}$ is equal to $\mathbf{0}$ or (ii) partial derivatives are non zero and are parallel. The second case (ii) where partial derivatives are non zero yet are parallel is not common in practice while case (i) appears quite often.

The underlying geometry of a face F is thus a parametric surface $\mathbf{x}(u, v)$. Yet, its domain is often smaller than A: A is usually trimmed by boundaries C_j and the geometry of the trimming curves are algebraic curves $\mathbf{c}_j(u, v) = 0$ defined in the (u, v) plane of F. Figure 2 shows an example of a trimmed surface.

Generating a triangular surface mesh of F consists in generating a planar triangular mesh in its parameter plane whose map through $\mathbf{x}(u, v)$ is a valid mesh in $\mathbb{R}^3$ with triangles of controlled shapes and sizes.

A triangle is valid in the (u, v) plane when it is properly oriented, i.e. when its area is strictly positive. It is indeed more complicated to assess that a triangle is valid in $\mathbb{R}^3$. Assume a triangle $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ with its non unit normal $\mathbf{n} = (\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})$ and the normal to the CAD surface at the centroid

$$(u_t, v_t) = \frac{1}{3}(u_a + u_b + u_c, v_a + v_b + v_c)$$

of the triangle:

$$\mathbf{n}_{\text{CAD}} = \partial_u \mathbf{x}(u_t, v_t) \times \partial_v \mathbf{x}(u_t, v_t).$$

We say that triangle (a, b, c) is valid if $\mathbf{n}_{\text{CAD}} \cdot \mathbf{n} > 0$.

In the example of Fig. 2, the depicted trimmed surface has no irregular points and the mesh generation procedure is usually straightforward. In this specific example, the anisotropic frontal-Delaunay approach that is implemented in Gmsh [2] was used based on the metric tensor

$$\mathcal{M} = \begin{pmatrix} \|\partial_u \mathbf{x}\|^2 & \partial_u \mathbf{x} \cdot \partial_v \mathbf{x} \\ \partial_u \mathbf{x} \cdot \partial_v \mathbf{x} & \|\partial_v \mathbf{x}\|^2 \end{pmatrix} \tag{1}$$

that is of full rank everywhere.

Surfaces with isolated irregular points are however very common in CAD systems: spheres, cones and other surfaces of revolution may contain one or two irregular points. Mesh generation procedures are known to be prone to failure close to irregularities. Consider for example the parametrization of a sphere as it is used to our best knowledge in every CAD system. A sphere of radius R centered at the origin is parametrized as

$$\begin{aligned} x(u,v) &= R\ \sin u\ \cos v \\ y(u,v) &= R\ \sin u\ \sin v \\ z(u,v) &= R\ \cos u \end{aligned}$$

where $u \in [0, \pi]$ is the inclination and $v \in [0, 2\pi[$ is the azimuth. At the poles, i.e. when $u = 0$ or $u = \pi$,

$$\partial_v \mathbf{x} = R(-\sin u \sin v, \sin u \cos v, 0) = (0, 0, 0)$$

vanishes and this parametrization is irregular at the two poles of the sphere.

In this paper, a new approach is proposed that allows to generate meshes of surfaces with irregularities in an efficient and robust fashion. At first, we explain in Sect. 2 why indirect surface mesh generation procedures become fragile at the vicinity of irregular points. Then in Sects. 3 and 5, we present the critical modifications to standard meshing procedures that allow to address issues related to irregular parametrizations. Examples of CAD models with thousand of spheres and cones are finally be presented in Sect. 7.

2 The Issue of Meshing Surfaces with Irregular Parametrizations

Two main approaches exist for surface meshing. The first approach, usually called the "direct approach" [3], consists in generating the mesh directly in $\mathbb{R}^3$. Different direct approaches have been proposed in the literature: advancing front methods [4, 5], octree based methods [6, 7], methods based on local mesh modifications [8, 9], methods based on restricted Voronoi diagrams [10], ... Octree- and Voronoi-based methods have in common the need to intersect a 3D object (an octree or a Voronoi Diagram) with the surface that is to be meshed. When an octree is used, the intersection of the octree with the surface is usually irregular and local mesh modifications have to be performed in order to obtain a quality mesh. On the other hand, when the Voronoi diagram of the points is used, recovering edges (sharp features) of the surface is an issue. Other direct methods generate triangles on the surface without using any kind of 3D object. Advancing front methods and paving methods [11] add points and triangle on the surface using a frontal approach. Those methods handle sharp features without difficulties and allow to generate quality

meshes. Yet, such methods are plaged with robustness issues (front colliding and 3D intersections of 2D objects). Some direct approaches [8, 9] start from a "CAD" mesh and modify it to produce a "computational" mesh with elements of controlled shapes and sizes. The main disadvantage of such an approach is that it requires an initial mesh. One may use STL triangulations provided by CAD modelers but those are not guaranteed to be watertight on a whole CAD model and a complex preprocessing step is usually required to fix holes and T-juntions. Another issue is related to what could be called an "isogeometric" argument: the final "computational" mesh and the inital "CAD" mesh are piecewise linear complexes that do not necessary cover the same geometry. Modifying an existing surface mesh using local mesh modification like vertex repositioning leads to vertices located outside of the input geometry i.e. the "CAD" mesh. While meshing procedures of this kind that actually ensure that the distance between the "CAD" and the "computational" mesh is bounded, those are based on complex datastructures and require to compute Haussdorff distances between triangulations [12].

When mesh generation procedures have access to parameterizations of surfaces, one can generate a planar mesh in the parametric domain and map it in 3D. This surface meshing approach is called "indirect". In Gmsh, surface meshes are generated in the parameter plane (u, v) and standard "off the shelf" anisotropic 2D meshers are used for generating surface meshes. This is of course the main advantage of the indirect approach: a priori, no major coding effort is required to go from planar meshing to surface meshing. This last statement is of course a little bit too optimistic. Ensuring that a planar mesh is valid is trivial: all triangles should be positively oriented. Now, if the surface parametrization $\mathbf{x}(u, v) \in \mathbb{R}^3$ is regular, then the mapping of the (u, v) mesh onto the surface is itself valid because the composition of two regular mappings is regular. For example, the very simple mesh of the parameter plane of the whole sphere presented in Fig. 3 maps exactly the sphere as depicted in the bottom part of Fig. 3.

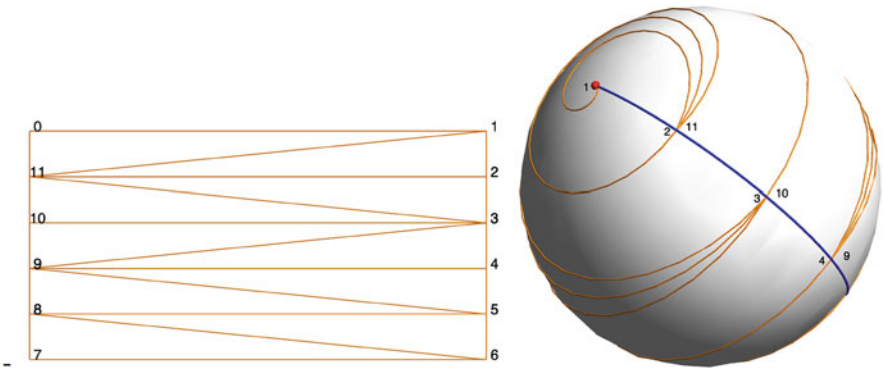

Fig. 3 A very simple mesh (left) of the parameter plane of a sphere and (right) its mapping through spherical coordinates

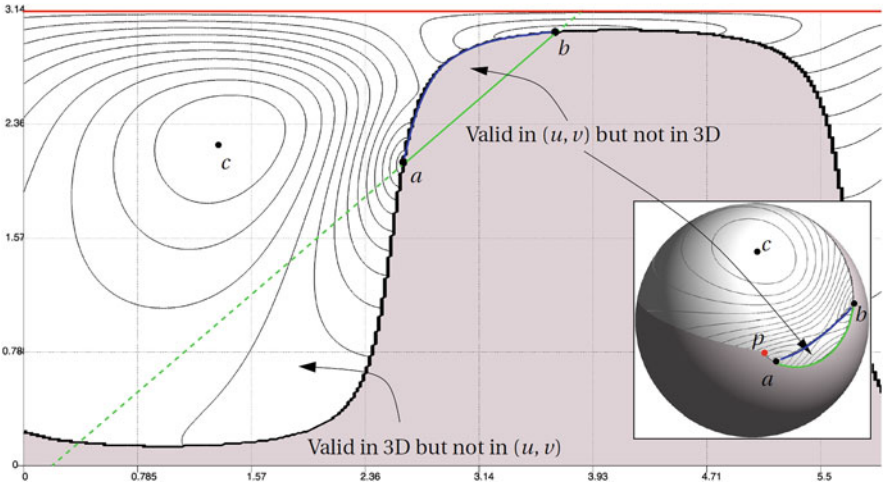

Fig. 4 Parameter plane of a sphere. We consider an edge (a, b). Figures depict the quality of a triangle (a, b, c) with c positioned anywhere in the parameter plane. The grey zone corresponds to invalid triangles in 3D

Here, the main issue is that we do not actually map entire triangles onto the surface but only their corners. The topology of the 2D mesh is simply "translated" in 3D: straight sided triangles in the (u, v) plane become straight sided triangles in 3D. Another "isogeometric" issue thus appears in the indirect approach: the mapping $\mathbf{x}(u, v)$ of a triangle in the (u, v) plane is not equal to the straight sided triangle in $\mathbb{R}^3$. So, a valid 2D triangle in the parameter plane does not necessary produce a valid 3D triangle on the surface. For example all the triangles in the parameter plane in Fig. 3 are mapped onto zero-area triangles in $\mathbb{R}^3$. On the other hand, an invalid 2D triangle (i.e. with a negative area) may be perfectly valid in 3D.

In order to illustrate those issues, Fig. 4 shows the example of the parameter space of a complete sphere. An edge (a, b) where b is close to the north pole p (in red) is considered. Edge (a, b) is used to form a triangle (a, b, c) where $c \in A = [0, 2\pi[\times [0, \pi]$. The iso-lines that are presented are iso-values of triangle qualities:[1] the particular point c drawn on the Figure is the only one in the parameter plane leading to a valid equilateral triangle in 3D. The grey zone in the Figure corresponds to the locations of points c that form invalid elements in 3D. Invalid elements in the parameter plane (u, v) correspond to points above the green line that passes through (a, b). It can be seen that there exists a zone where triangles are valid in 2D but not in 3D, and another zone where elements are valid in 3D but not in 2D. Some interesting comments can be made with respect to Fig. 4:

[1] We use as quality metric the ratio $2\frac{r}{R}$ between the inner-radius r and the circumradius R multiplied by 2 in order to have a quality equal to one for the equilateral triangle.

- The blue line is the 3D geodesic between a and b. This geodesic is far from being a straight line in the parameter plane, especially when the edge (a, b) is close to the pole (in red). Geodesics are straight lines in the parameter plane when the metric tensor $\mathcal{M}$ is constant (this is a sufficient condition). We will see in the next section that geodesics that are incident to an irregular point are also straight lines, even though the metric $\mathcal{M}$ has strong variations close to singularities.
- The point c in Fig. 4 that corresponds to an equilateral triangle (a, b, c) is always in the valid zone i.e. the zone where triangles are both valid in 2D and 3D. More generally, good quality triangles can always be formed in the parameter plane, even when the metric is very distorted.
- The zone that is valid in 2D but not in 3D is the most problematic for mesh generation algorithm that work in the parametric plane. Hopefully, this zone only contains points c for which triangles (a, b, c) are of bad quality.

It is difficult to generalize those three comments to general surfaces but our experience (through numerical experiments) shows that they do indeed hold.

The main question can thus be formulated as follows: *assuming a surface with a parametrization that may contain isolated irregular points, can we always find a valid 2D mesh that corresponds to a valid 3D mesh?*

When we started to think about version 4 of Gmsh, our answer to that question was tending to be *no*, at least using the current implementation of the surface mesh generators. The typical issue that was encountered at the time is illustrated on Fig. 5. The right part of the Figure represents the mesh in the parametric plane (u, v) while the right part of the Figure represents the surface mesh in $\mathbb{R}^3$.

In Fig. 5, the surface $\mathcal{S}$ is a sphere. Points like c or g (in green) are classified on model face F. Points like d or f (in pink) are classified on regular model edges that bound F while points like b and e are classified on the seam of F (in order to have $\partial(\partial F) = \emptyset$ some CAD systems like OpenCASCADE close periodic surfaces with a seam). Point b is a pole of the sphere: it is an irregular point. The parametric mesh is perfectly valid i.e. triangles cover exactly A without overlap. Yet, even though

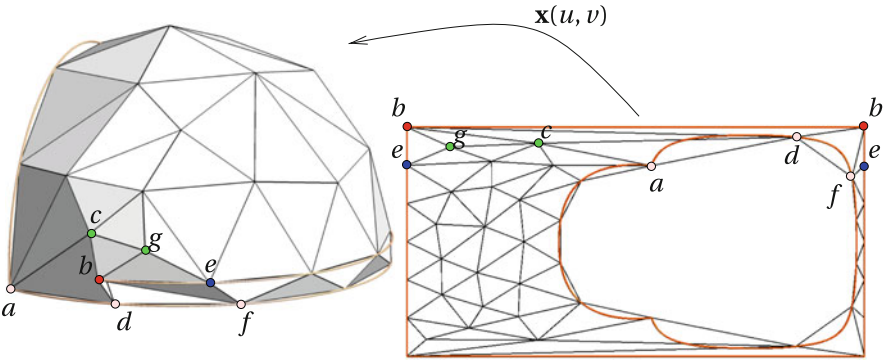

Fig. 5 A valid mesh in the parameter space that is invalid in the real space

triangle (b, c, d) is correctly oriented in the parametric plane, it is invalid in $\mathbb{R}^3$. We are here in the situation of Fig. 4 where point a is above the geodesic between c and d in the parameter plane. One single edge flip could potentially make the 3D mesh valid and of better quality: exchanging edges (c, d) and (a, b) fixes all issues. Yet, doing so makes the parametric mesh invalid. With the set of points that is depicted in Fig. 5, we found it impossible to build a quality mesh in $\mathbb{R}^3$ that is valid in the (u, v) plane.

Contrary to what one might think, the main issue here is not the fact that the metric tensor (1) is of rank 1 at irregular points and very distorted around it. In the context of mesh generation, geometrical queries like the evaluation of the metric tensor $\mathcal{M}$ are never done at irregular points; and anisotropic mesh generators are able to generate meshes for smooth metric fields even though they are very distorted. The mesh generation issue that arises here is essentially related to triangles (e.g. (b, c, d) in the Figure) and edges that have one vertex like b that corresponds to an irregular point of the parametrization.

Another minor issue will be fixed by our new approach. The existence of one degenerated mesh edge connecting points b implies the existence of an irregular triangle (d, b, b) that has one degenerated edge. This triangle can be eliminated in a post processing stage but its presence is quite annoying in the mesh generation process: computation of circumcircles, edge flips (flipping edge (b, d) does not change the mesh), ...

3 Geodesics of Surfaces of Revolution

Most of the CAD surfaces that have irregular points are surfaces of revolution. Consider a surface of revolution with respect to the z-axis and suppose that the generating curve is

$$\mathbf{c}(v) = (f(v), 0, g(v)) \quad , \quad v \in [0, T].$$

The parametrization of the surface is given by

$$\mathbf{x}(u, v) = (f(v)\cos(u), f(v)\sin(u), g(v)), \tag{2}$$

$(u, v) \in [0, 2\pi[\times[0, T]$. Geodesics of surface of revolution, even though their forms are not trivial (see for example the blue line of Fig. 4), have specific properties [13]. One interesting property of surfaces of revolution is that meridian curves $u = \text{cste}$ are geodesics.

Surfaces of revolution may have irregular points: if $f(0) = g(0) = 0$ in (2), then $\mathbf{x}(u, 0) = (0, 0, 0)$ for every u. The origin of the axis belongs to the surface and is thus an irregular point as defined above. Let us now look at the parameter plane (u, v) corresponding to a surface of revolution with a irregular point at $u = 0$.

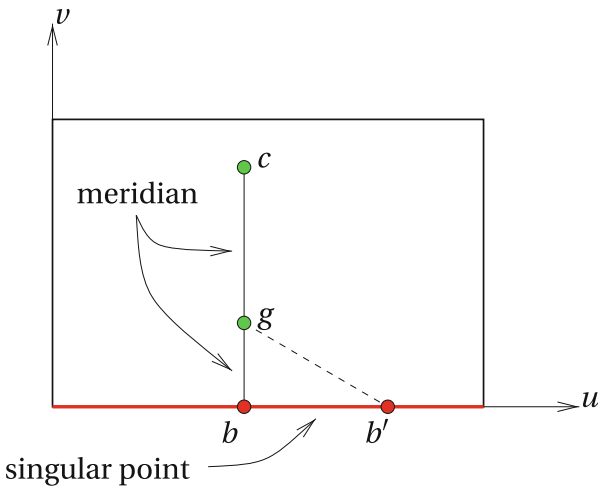

Fig. 6 Parameter space corresponding to a surface of revolution with an irregular point at $u = 0$

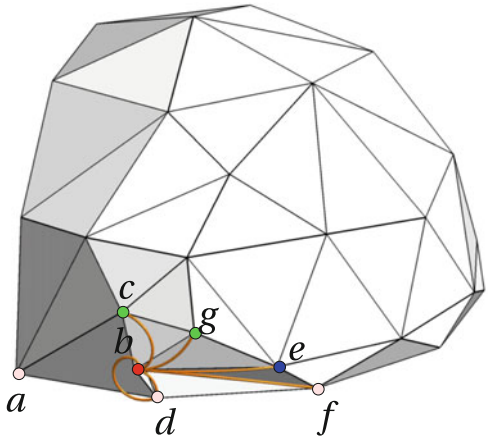

Fig. 7 True mapping of straight lines in the parameter space onto $\mathbb{R}^3$ close to an irregular point

Figure 6 gives an illustration of that situation. The thick red line $u = 0$ is mapped onto one single point $\mathbf{x} = (0, 0, 0)$. Thus, edges (g, b) and (g, b') have the same end-points in $\mathbb{R}^3$ but the only geodesic between those two points is the meridian (g, b).

This simple result allows us to critically examine Fig. 5: edges like (c, b), (g, b) or (d, b) are far from being geodesics and are thus far from the their corresponding straight edges in $\mathbb{R}^3$, as depicted in Fig. 7. On the other hand, edge (b, f) is close to be a geodesic and its 3D representation is close to the corresponding straight line.

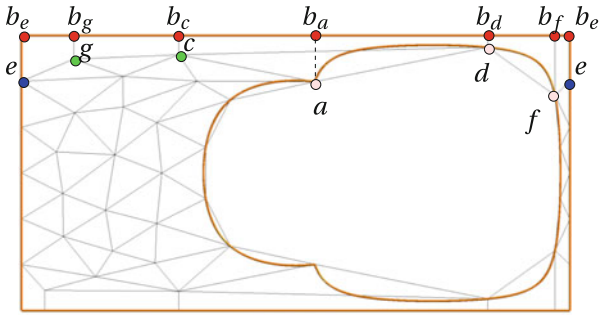

Fig. 8 New representation in the parameter space where every edge connected to an irregular point is a meridian. Point b_i's all belong to edge (b_i, i) even though all b_i's 3D locations are equal

Coming back to the mesh generation problem, it should be interresting to replace all the edges that are incident to irregular points by meridians. The new representation of the mesh in the (u, v) plane is depicted in Fig. 8.

With this representation, the unique edge flip that allows to have a valid mesh in $\mathbb{R}^3$ is permitted. Edge (a, b_a) (in dashed lines) can replace edge (c, d) without creating invalid triangles in the parameter plane (edge (c, g) could be flipped as well even though it is not required). Note here that triangles incident to irregular points are now right trapezoids with one degenerated edge, which means that no degenerated triangles exist in that new representation.

4 Modifications of the Initial Mesh

Our surface mesh generation procedure starts with an initial "empty mesh" i.e. a mesh in the parameter space that contains only vertices of the surface boundaries. Then, in this new procedure, edges that are adjacent to singularities are transformed onto geodesics. The question that is addressed in this section is the validity of this initial transformation.

Consider the surface presented in Fig. 9 together with a mesh generated using the new version of Gmsh's *MeshAdapt* surface mesher (see Sect. 5 below). The initial mesh that contains all boundary points is presented in Fig. 9. Again, a seam and two irregular points a and b are present in the surface plus a trimming curve that contains points c and d.

In our new procedure, all edges that are adjacent to irregular points are transformed onto geodesics. Figure 9 shows the result of that transformation: points like b_c are added to the parameter space to generate a geodesic cb_c.

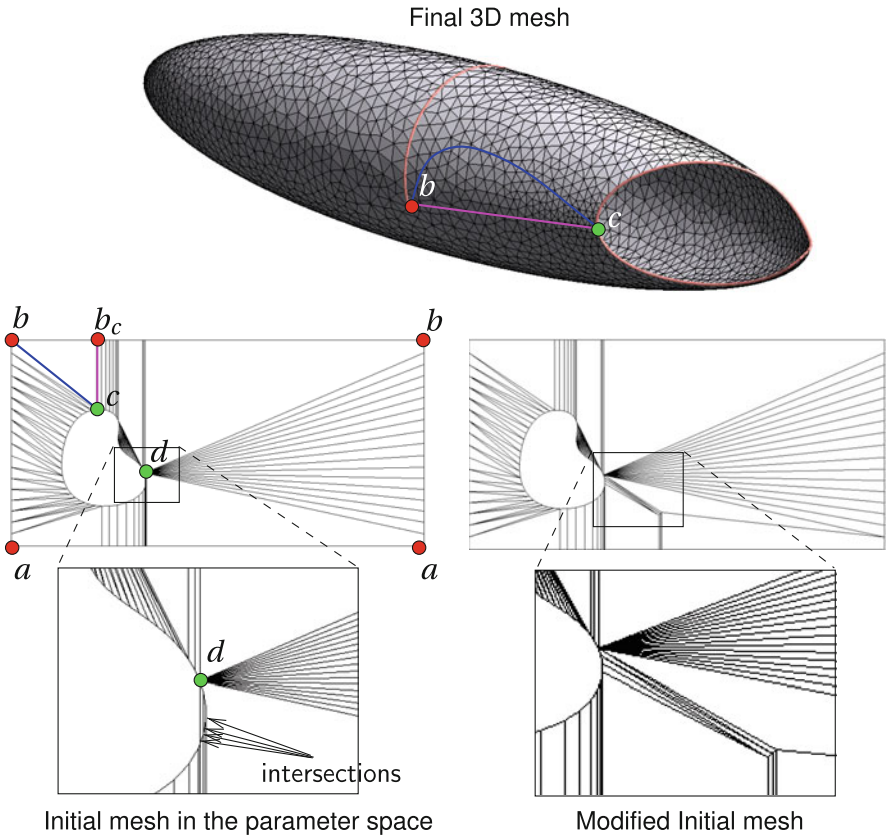

Fig. 9 Modification of the initial mesh in the parameter space in order to avoid edge intersections

It is actually easy to figure out that the initial mesh of Fig. 9 is actually wrong (inverted) in the parameter space. Figure 9 shows a zoom of the three problematic edges that make intersect other edges of the esh and makes it invalid. The problem comes from the fact that, in Fig. 9, four edges of the internal boundary are initially connected to the point on the bottom right of the external rectangle. Yet, those edge normals are pointing upward so the proposed correction creates edges that intersect other edges of the internal boundary.

Addressing this problem is indeed quite simple. All problematic geodesic edges are split along their original path (not along the geodesic of course) up to the point when no intersection occurs. The resulting initial mesh is presented in Fig. 9.

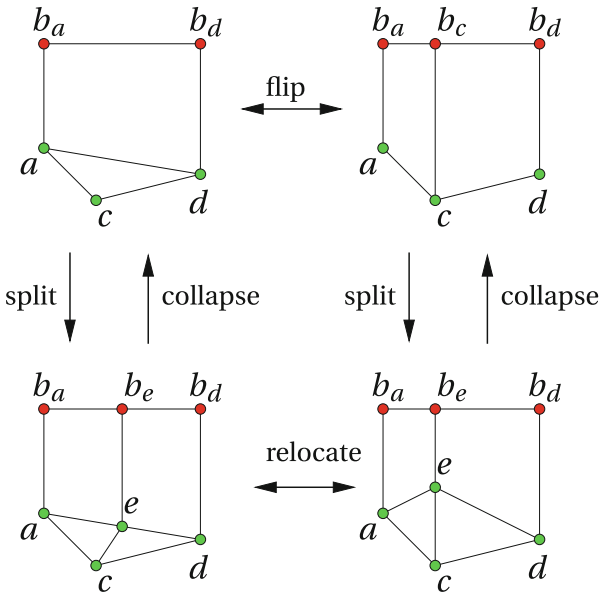

Fig. 10 Local mesh modifications with irregular points

5 Local Mesh Modifications: Gmsh's MeshAdapt Algorithm Revisited

Gmsh's most basic surface mesher is called *MeshAdapt*.[2] MeshAdapt's surface meshing strategy is based on the concept of local mesh modifications [14–16]. The algorithm works as follows. First, an initial mesh containing all the mesh points and edges of the model edges that bound a face is built in the parametric space (u, v) (see Sect. 4). Then, local mesh modifications are applied to the mesh in the parameter plane:

1. Each edge that is too long is split;
2. Each edge that is too short is collapsed;
3. Edge flips are performed in order to increase mesh quality;
4. Vertices are re-located optimally after steps 1, 2, and 3.

Figure 10 illustrates local mesh modifications applied to edges that are in the vicinity of a irregular point b. When edge (a, c) is flipped, a new instance of point b_c is created on the degenerated edge and point c becomes connected to b_c. The operation can be reversed as depicted in Fig. 10. When an edge like (a, d) is split at point e, a new point b_e is created on the degenerated line. When an edge like (c, b_c) that is

[2] `gmsh -algo meshadapt` is the commandline that forces gmsh to use that algorithm.

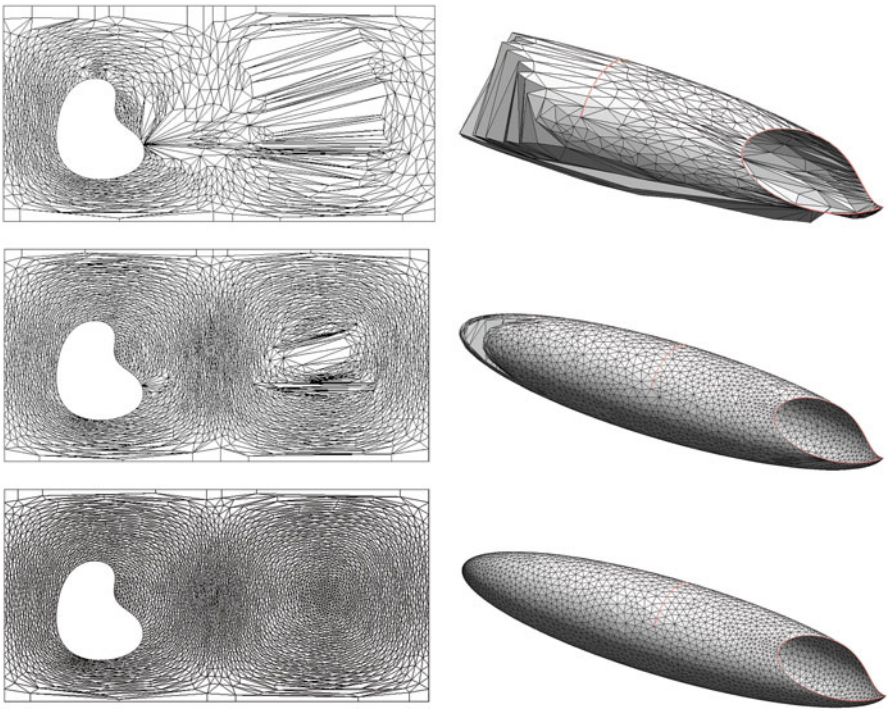

Fig. 11 Different stages of the *MeshAdapt* algorithm

connected to the irregular point is split, b_c is replaced by b_e. Note that when a point like e is relocated, point b_e is relocated as well.

All four local mesh modifications of our algorithm involve details of implementation that are too specific to be described in a paper but that are critical for robustness. Interrested readers can download the source code of Gmsh 4.3.0 that implements the algorithm that exactly corresponds to the examples of the paper. Nevertheless, the most critical part of the *MeshAdapt* surface mesher is the vertex relocation, both in term of the final mesh quality and of CPU time (it actually takes about 60% of the total mesh generation time). When parameterizations are very distorted, simple smoothing strategies do not actually produce improvements of the mesh, especially close to singularities. In this new version of the algorithm, advanced optimization procedures have been used [17] for vertex relocation. Figure 11 show the mesh of the surface of Fig. 9 at different stages of the *MeshAdapt* algorithm.

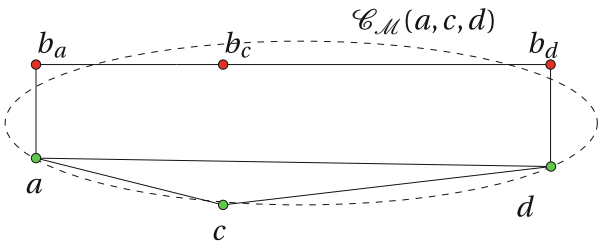

Fig. 12 Illustration of the application of the circle criterion close to an irregular point

6 Delaunay Mesh Generation Close to Irregular Points

Gmsh's frontal-Delaunay algorithm is an extension to surface meshing of the planar frontal-Delaunay mesher described in [18]. Points are inserted in the domain in a frontal fashion while always keeping a valid mesh during the process. The mesh is generated in the (u, v) plane which means that an anisotropic Delaunay criterion is required to produce isotropic meshes in 3D.

The most critical operation involved in that algorithm is the edge flip. Consider Fig. 12: we would like to figure out wether edge (a, d) should be flipped or not. In order to apply Delaunay's empty circle criterion, we actually work in the tangent plane and compute a unique metric tensor $\mathcal{M}$ at location $(a + b_c + c + d)/4$ that is symmetrical with respect to points a, b_c, c, and d. This allows to avoid "unstable flips". In this tangent plane, circle $C_{\mathcal{M}}(a, c, d)$ is an ellipsis. The new representation that is proposed here allows to provide a robust way of computing Delaunay flips.

In the example of Fig. 12, edge (a, d) should be flipped to (c, b_c) because b_c is inside $C_{\mathcal{M}}(a, c, d)$. Other occurences of b like b_a or b_d may be located outside $C_{\mathcal{M}}(a, c, d)$ but the only edge that should be considered in the circle test is the geodesic (c, b_c).

7 Examples

We have chosen two examples that were invariably creating invalid elements in all previous versions of Gmsh.

7.1 Many Spheres

One of the nastier parametrization that is constantly used in CAD systems is the sphere, with its two irregular points at the poles. As a first example, we have generated a CAD model that consist in a unit cube B containing 5000 spheres with (pseudo-)random centers and radii S_i, $i = 1, \ldots, 5000$. The final CAD model C is computed as

$$C = B \setminus S_1 \setminus S_2 \cdots \setminus S_{5000}.$$

The final model C is depicted in Fig. 14. It consist in 3 volumes, 4971 surfaces (mainly trimmed spheres resulting from the boolean operations) and 18112 curves. Gmsh's actual script that was used to generate that model is given by

```
// Gmsh's script to generate
// a CAD model with 5000 spheres
SetFactory("OpenCASCADE");

DefineConstant[
  rmin = {0.002, Name "Min radius"}
  rmax = {0.03, Name "Max radius"}
  n = {5000, Name "Number of spheres"}
];

For i In {1:n}
  r = rmin + Rand(rmax - rmin);
  x = -0.5 + Rand(1);
  y = -0.5 + Rand(1);
  z = -0.5 + Rand(1);
  Sphere(i) = {x, y, z, r};
EndFor

Block(n + 1) = {-0.5, -0.5, -0.5, 1, 1, 1};

BooleanDifference{ Volume{n + 1}; Delete; }
                 { Volume{1:n}; Delete; }
```

Some of the surfaces of the 5000 spheres model are really complex to mesh, especially when a trimmed curve is very close to an irregular point. Figure 13 shows a complicated situation.

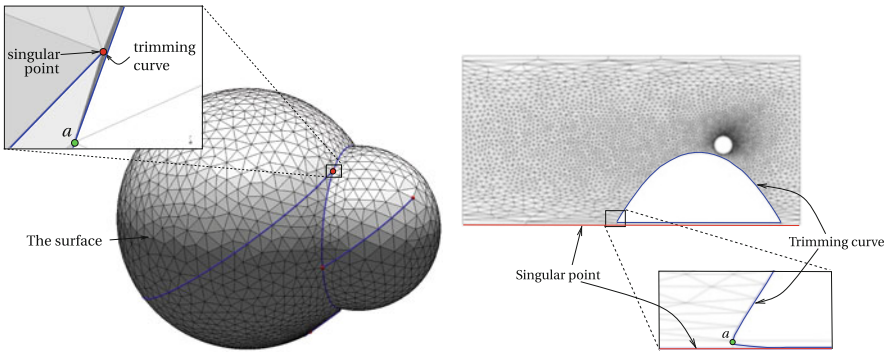

Fig. 13 One surface of the 5000 sphere model that exhibit a complex configuration close to one of the poles of the sphere

7.2 Many Ellipsoids

Generating an ellipsoid can be done by applying an affine transformation to a sphere followed by a rotation. Using Gmsh's built-in scripting language, this can be achieved as follows:

```
R = 2;
Sphere(1) = {0, 0, 0, R};
Affine{ 1,0,0,0, 0,10,0,0, 0,0,1,0 }
        { Volume{i}; }
Rotate {{Sqrt(2), Sqrt(2) , 0},
        {0, 0, 0}, Pi/3} Volume{i};}
```

Ellipsoids have parametrizations that are even more distorted than spheres. In the following example, 450 ellipsoids E_i, $i = 1, \ldots, 450$ have been inserted into a unit cube, with random orientantions and random sizes. The final CAD model is again built as the unit cube "minus" all ellipsoids. In OpenCASCADE, ellipsoids are encoded as B-spline surfaces and their intersections takes way more effort than intersecting spheres: it actually took about 7 min to generate the CAD model while only 3 min were required to generate the surface mesh (295 surfaces for a total of 220,523 triangles) and only 14 s were required to generate the 3D mesh (55 volumes and 735 million tetrahedra). Figure 14 show a picture of the resulting mesh.

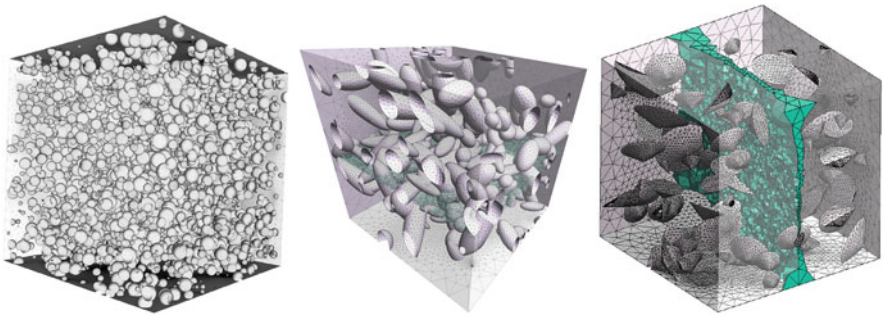

Fig. 14 A complex model made of 5000 spheres (left), another model made of 450 ellipsoids (center) and a third model made of 100 intersecting cones (right)

7.3 Many Cones

Apex of cones are singular points of their parametrizations. We have generated a geometry with 100 intersecting cones in a box. Figure 14 show an image of the mesh.

Conclusions

Generating in a reliable manner a quality surface mesh for arbitrary CAD models entails dealing with various CAD systems idiosyncrasies. In this paper, we have presented a crucial modification of Gmsh's surface meshing algorithms that is an important step forward towards this goal, by handling surfaces with a finite number of irregular points. The two test cases that are presented are only indicative: hundred of other examples were successfully tested during the writing of this paper, all with surfaces that have singularities.

The lack of a structure of proof for surface meshing that is briefly explained in the introduction is one of the curses mesh generation people have to live with. Surface meshers that are reasonably reliable are all based on heuristics and their disfunctions and bugs can only be found through extensive testing. For example, the issue that has been explained in Sect. 4 has only been encountered twice in all our test cases. Yet, it has to be addressed because the rare conditions of apparition of the bug will definitively happen at some point in a software like Gmsh that is used by a large community.

In conclusion, we are aware that other issues will show up in the long term (maybe impossible) goal of 100% reliability. Yet, the improvements that are presented in this paper definitively make Gmsh's surface meshers more reliable on a large number of test cases that were failing in previous versions. The method that is proposed does not require deep modifications of existing surface meshing algorithms. Yet, it allows to produce quality meshes for all test cases that we encountered.

References

1. Weiler, K.: Geometric Modeling for CAD Applications. Springer, Berlin (1988)
2. Geuzaine, C., Remacle, J.F.: Int. J. Numer. Methods Eng. **79**(11), 1309 (2009)
3. Borouchaki, H., Laug, P., George, P.L.: Int. J. Numer. Methods Eng. **49**(1–2), 233 (2000)
4. Hartmann, E.: Vis. Comput. **14**(3), 95 (1998)
5. Owen, S.J., Staten, M.L., Canann, S.A., Saigal, S.: Int. J. Numer. Methods Eng. **44**(9), 1317 (1999)
6. Maréchal, L.: In: Proceedings of the 18th international meshing roundtable, pp. 65–84. Springer, Berlin (2009)
7. Shephard, M.S., Georges, M.K.: Int. J. Numer. Methods Eng. **32**(4), 709 (1991)
8. Frey, P.: Yams a fully automatic adaptive isotropic surface remeshing procedure. Ph.D. thesis, Inria (2001)
9. Béchet, E., Cuilliere, J.C., Trochu, F.: Comput. Aided Des. **34**(1), 1 (2002)
10. Yan, D.M., Lévy, B., Liu, Y., Sun, F., Wang, W.: In: Computer Graphics Forum, vol. 28, pp. 1445–1454. Wiley Online Library, New York (2009)
11. Blacker, T.D., Stephenson, M.B.: Int. J. Numer. Methods Eng. **32**(4), 811 (1991)
12. Borouchaki, H., Frey, P.: Comput. Methods Appl. Mech. Eng. **194**(48–49), 4864 (2005)
13. Do Carmo, M.P.: Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition. Courier Dover Publications, New York (2016)
14. Remacle, J.F., Li, X., Chevaugeon, N., Shephard, M.S.: In: IMR, pp. 261–272 (2002)
15. Li, X., Shephard, M.S., Beall, M.W.: Comput. Methods Appl. Mech. Eng. **194**(48–49), 4915 (2005)
16. Remacle, J.F., Li, X., Shephard, M.S., Flaherty, J.E.: Int. J. Numer. Methods Eng. **62**(7), 899 (2005)
17. Knupp, P.M.: Eng. Comput. **15**(3), 263 (1999)
18. Rebay, S.: J. Comput. Phys. **106**(1), 125 (1993)