Received: 12 January 2021 | Revised: 13 April 2021 | Accepted: 7 May 2021

DOI: 10.1002/nme.6747

RESEARCH ARTICLE

WILEY

Automatic feature-preserving size field for three-dimensional mesh generation

Arthur Bawin[1,2] | **François Henrotte**[1,3] | **Jean-François Remacle**[1]

[1]Institute of Mechanics, Materials and Civil Engineering, Université catholique de Louvain, 1348 Louvain-la-Neuve, Belgium

[2]Département de Génie Mécanique, École Polytechnique de Montréal, C.P. 6079, succ. Centre-ville Montréal, Quebec, H3C 3A7, Canada

[3]Applied and Computational Electromagnetics (ACE), Department of Electrical Engineering and Computer Science Montefiore Institute, University of Liege, 4000 Liège, Belgium

Correspondence
Arthur Bawin, Institute of Mechanics, Materials and Civil Engineering, Université catholique de Louvain, Avenue Georges Lemaitre 4, bte L4.05.02, Louvain-la-Neuve 1348, Belgium.
Email: arthur.bawin@uclouvain.be

Funding information
Canadian Network for Research and Innovation in Machining Technology, Natural Sciences and Engineering Research Council of Canada; Fonds pour la Formation à la Recherche dans l'Industrie et dans l'Agriculture, Grant/Award Number: FRIA/FC 29571; H2020 European Research Council, Grant/Award Number: ERC-2015-AdG-694020

Abstract

This article presents a methodology aiming at easing considerably the generation of high-quality meshes for complex three-dimensional (3D) domains. To this end, a mesh size field $h(\mathbf{x})$ is computed, taking surface curvatures and geometric features into account. The size field is tuned by five intuitive parameters and yields quality meshes for arbitrary geometries. Mesh size is initialized on a surface triangulation of the domain based on discrete curvatures and medial axis transform computations. It is then propagated into the volume while ensuring the size gradient ∇h is controlled so as to obtain a smoothly graded mesh. As the size field is stored in an independent octree data structure, it can be computed separately, then plugged into any mesh generator able to respect a prescribed size field. The procedure is automatic, in the sense that minimal interaction with the user is required. Applications of our methodology on CAD models taken from the very large ABC dataset are presented. In particular, all presented meshes were obtained with the same generic set of parameters, demonstrating the universality of the technique.

KEYWORDS

background mesh, curvature, feature size, mesh generation, octree, size field, surface proximity

1 | INTRODUCTION

Geometric models used in industry have considerably grown in complexity over the last decades, and it is now common to mesh models with tens of thousands of model faces. Ideally, a designer would create the CAD model, press the *generate mesh* button, and obtain in less than a minute a computational mesh valid *as is* for a finite element simulation. Practitioners in the field know however that things do not work out that easily in reality. Mesh generation for complex geometries is in practice a time-consuming task often involving intermediary meshes, progressively enhanced to fulfill specified mesh size and quality requirements.

The purpose of mesh generation is to build meshes with elements of controlled size and quality. In the context of mesh adaptation, the meshing algorithm is constrained by a mesh size field defined on the domain to be meshed, and

whose value at a point is the expected element size in the vicinity of that point. The mesh size is usually derived from an error estimation procedure performed on the solution of a prior finite element or finite volume analysis, by requesting a smaller size at places where the discretization error is deemed large.

Yet, when solving a problem for the first time, an initial mesh has to be generated without information from a prior computation, and the size field to generate that initial mesh has to be constructed from scratch on basis of the geometrical data of the model only. Given a CAD model, there exist a number of theoretical prerequisites on the size field to ensure a computable mesh, and the purpose of this article is to describe an automated algorithm to compute a mesh size field fulfilling those prerequisites a priori. The proposed approach is "user-driven," in the sense that users should be able to generate a workable computational mesh in one click on basis of a limited number of intuitive meshing parameters, understandable by any finite element practitioner with no extensive background in meshing.

An isotropic mesh size field $h(\mathbf{x})$ is thus a scalar function indicating the expected element size at any point $\mathbf{x}$ in a domain to be meshed. A first design choice concerns the mathematical representation of h. As our goal is to build a "first mesh," no background mesh is yet available against which h could be interpolated. A classical solution (e.g., Gmsh[1]) is to define element sizes directly on the geometrical entities of the model. Mesh size can be prescribed at the vertices of the CAD model, for instance, and smoothly interpolated on model edges. They are then subsequently interpolated on surface mesh vertices. Size fields interpolated this way may however be biased by geometric features of the surface mesh, such as gaps, fins or channels, which are assigned locally a small size that is not expected to spread out at distance in the bulk of the volume. This approach is therefore not 100% reliable and defining the size on auxiliary objects allows for a better control and prevents the aforementioned phenomenon.

Two kinds of representation for mesh size fields are encountered in the literature: simplicial background meshes,[2-5] and Cartesian grids, initially in the form of uniform grids,[6] and later on in the form of nonuniform or tree-based grids, such as octrees.[7-9] A graphical representation of such data-structures in the two-dimensional (2D) case can be found in the first figure of Persson.[10] As the refinement of uniform Cartesian grid is constrained by the smallest feature in the CAD model, their memory cost quickly becomes prohibitive in practice, and they were rapidly abandoned for the sake of simplicial background meshes and octrees.

Both these representations have their pros and cons. Simplicial background meshes are typically triangulations provided by the CAD modeler, where mesh size is stored on vertices and interpolated on queried nodes. They offer an accurate representation of the boundaries, and the mesh size query procedure in the 3D case is reduced to a search in the 2D parametric space.[4,11] However, size fields represented this way are rather sensitive to the location of the vertices in the background mesh,[3] and the access time to the mesh size at a point in the background mesh might be linear in the number of nodes in the worst case. To improve on this, Chen et al.[4] proposed a *walk-through* algorithm based on the backward search from Shan et al.,[12] although it requires a well-guessed element to quickly locate the point in the mesh, assuming an interconnection between the background mesh and the meshing algorithm. Octrees, on the other hand, are orientation sensitive Cartesian structures with applications in both direct meshing[13,14] and size field design. They lack the geometrical flexibility of simplicial meshes and significative refinement may be necessary to accurately resolve surface-based information, for example, curvatures. However, octree-based size fields offer adaptive capabilities to represent quickly and easily complex size distributions across the structure. Moreover, octrees offer fast access to query points in $\mathcal{O}(\log_8 n)$ time, where n denotes the number of octants, that is, the number of leaves in the octree.

In this work, we choose to store the mesh size field in an octree for adaptivity and efficiency concerns. The octree implementation is provided by p4est,[15] of which the serial version is used. In our implementation, a uniform mesh size is assigned to each octant in the octree, which is the most natural option with p4est.

As mentioned earlier, there are theoretical prerequisites to ensure the computability of a mesh on a given CAD model. Those prerequisites can be associated with the following five intuitive mesh parameters.

Bulk size. A bulk or default mesh size h_b. When creating the size field, the octree is refined uniformly until every octant size is smaller or equal to h_b. This amounts to say that all sizes are initially set to the bulk value h_b.

Curvature. When using piecewise linear elements, the main term of geometrical error produced by a mesh is related to the curvature of surfaces. The local mesh size $h(\mathbf{x})$ should hence be related to the maximal curvature $\kappa_n(\mathbf{x})$ of the surfaces. This is done with the node density parameter n_d that specifies the number of subdivisions of the perimeter of the local osculating circle.

Small features. A CAD model may also contain narrow or thin regions, or *features*, for example, longerons that are the load-bearing components of aerospace structures. As such regions may have moderate or no curvature at all, they are likely to be overlooked by an algorithm that solely links mesh size with curvature. Features are typically included using a distance function[16] or the medial axis transform.[8,9] The latter is used here to estimate the thickness of narrow regions, see

Section 2.2. On this basis a third parameter n_g is defined in our algorithm that specifies the minimal number of elements across the thickness of a narrow region. We call *feature mesh size* h_f that thickness divided by n_g. The feature mesh size h_f and the curvature mesh size h_c are the indicators used by our algorithm to recursively refine the octree containing the mesh size field, see Section 2.3.

Boundedness. A minimum mesh size $h_{\min}$ has to be defined as a fourth parameter in our algorithm to forbid unacceptably small mesh sizes, whenever curvatures are very high for instance (e.g., at corners or at the tip of a cone).

Smoothness. Accurate finite element and finite volume simulations usually require that mesh sizes vary not too abruptly across the domain of computation. Yet, curvatures and feature sizes may exhibit sharp variations in practice, resulting in unacceptably large mesh size gradations. A fifth parameter $\alpha > 1$ is thus defined that bounds (from above) the length ratio between two adjacent edges in the mesh. In Section 2.4, we show that this condition is equivalent to limit the mesh size gradient to $\|\nabla h\| < \alpha - 1$.

Our approach thus defines five parameters that are easy to understand by finite element practitioners, and can be given a reasonable and rather universal default value:

- the bulk size, or default mesh size on newly created octants, h_b. The default value is $h_b = L/20$, where L is the largest dimension of the axis-aligned bounding box of the CAD model;
- the minimal size allowed in the final mesh, $h_{\min}$. The default value is $h_{\min} = L/1000$;
- the number of elements n_d used to accurately discretize a complete circle. The default value is $n_d = 20$;
- the number of element layers in thin gaps n_g. The default value is $n_g = 4$;
- the gradation, or length ratio of two adjacent edges in the final mesh, α. The default value is $\alpha = 1.1$.

In addition to these geometry-based criteria, a user-defined mesh size function $h_u(\mathbf{x})$ can be provided to specify exterior constraints on the size field. If so, the smallest instruction between $h_u(\mathbf{x})$ and the geometry-based mesh size is considered throughout the construction of the size field.

This size field computation has been implemented in Gmsh (version 5 and higher), which is also the tool used to generate all meshes presented in this article. It is planned that the algorithm presented in this article be soon integrated as a standard procedure in the meshing pipeline of Gmsh.

2 | DESCRIPTION OF THE ALGORITHM: A WORKED-OUT EXAMPLE

To illustrate the steps of the construction of the mesh size field $h(\mathbf{x})$, the CAD model of an engine block is considered as an application example (Figure 1). This geometry contains curved surfaces and narrow features that are typical of industrial CAD models. The input data for our algorithm is a surface mesh of the CAD model, from which curvature and feature mesh size are computed. The size field is generated in an independent structure in the five following steps (Figure 1): (*i*) compute the curvature mesh size h_c from the approximate curvature on the surface mesh of the model; (*ii*) compute the feature mesh size h_f from the medial axis of the geometry; (*iii*) initialize the octree as the bounding box of the model and refine it uniformly until the size of all octants is at most the bulk size h_b; (*iv*) recursively refine the octree based on both the curvature and the feature mesh size, and assign the appropriate uniform mesh size in all newly created octants; (*v*)

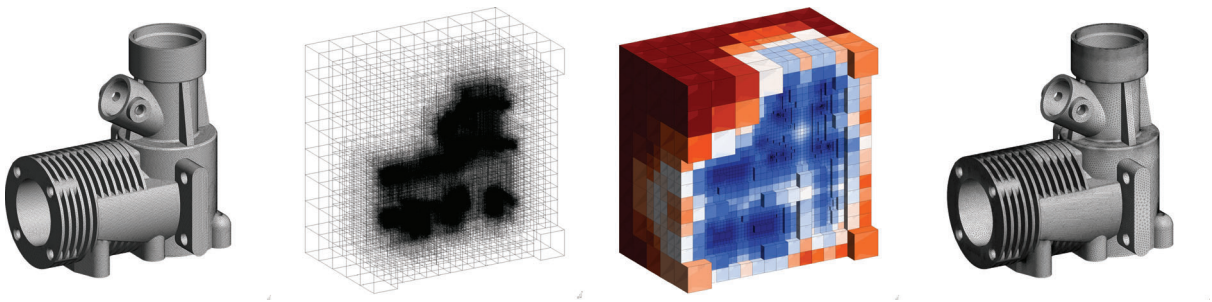


FIGURE 1 Overview of the algorithm for mesh size field computation, from left to right: ($i - ii$) surface mesh of the engine block, from which discrete curvature and feature sizes are computed; ($iii - iv$) the octree is refined based on curvature and feature mesh sizes, (v) mesh size gradient is limited, yielding smoother mesh size field; generation of the final mesh

smooth out the size field so as to limit its gradient to $\alpha - 1$. During step (v), the structure of the octree is not modified. Only its stored sizes $h(\mathbf{x})$ are limited to satisfy $|\nabla h| < \alpha - 1$, see Section 2.4.

2.1 | Approximation of surface curvatures

When a CAD model is represented with a mesh of piecewise linear elements, the main term of geometrical error is due to the curvature of surfaces. The mesh size should thus be reduced in areas of high curvature. We introduce to this end the *curvature mesh size* $h_c(\mathbf{x})$. Although definition slightly varies in the literature,[4,5,17,18] they all rely on the subdivision of the perimeter of local osculating circles. The osculating circle at a point of a planar curve is the circle that best approximates the curve in the vicinity of the point, that is, having same tangent and same curvature. On a smooth surface, there is an osculating circle in every direction, and the most critical curvature mesh size is related to the minimal radius of these circles, or reciprocally to the maximum normal curvature $\kappa_{n,\max}(\mathbf{x})$. Curvature mesh size is thus defined as follows:

$$h_c(\mathbf{x}) = \frac{2\pi r(\mathbf{x})}{n_d} = \frac{2\pi}{\kappa_{n,\max}(\mathbf{x})\ n_d}, \tag{1}$$

with n_d a user-defined node density.

If a CAD model is available in the background, surface parametrizations are at-hand and the maximal curvature $\kappa_{n,\max}(\mathbf{x})$ can be obtained through the solid modeler's API. Curvature queries on the CAD model can be costly however, and are usually designed for single point query. In this work, the input data considered for the size field pipeline is a triangulation of the CAD model, on which normal curvatures are approximated. We assume that the triangulation is colored, that is, triangles on each side of a feature edge, such as a ridge or a corner, are identified with a different color, so that surface discontinuities are not taken into account when computing the discrete curvature.

To approximate surface curvature, the tensor averaging methodology described by Rusinkiewicz[19] was considered. It is here briefly recalled with their notations. Let $(\mathbf{u}, \mathbf{v})$ denote an orthonormal basis in the tangent plane at a point $\mathbf{x}$ of a smooth surface, and $\boldsymbol{s} = (s_1, s_2)$ be an arbitrary direction in that plane. The normal curvature at $\mathbf{x}$ in the direction $\boldsymbol{s}$ is given by

$$\kappa_n(\mathbf{x}) = \mathbf{II}(\boldsymbol{s}, \boldsymbol{s}) = \begin{pmatrix} s_1 & s_2 \end{pmatrix} [\mathbf{II}] \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} s_1 & s_2 \end{pmatrix} \begin{pmatrix} e & f \\ f & g \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}, \tag{2}$$

where $\mathbf{II}$ denotes the second fundamental form defined hereafter, and $[\mathbf{II}]$ denotes its matrix representation in the chosen basis. The eigenvalues $\kappa_1(\mathbf{x})$ and $\kappa_2(\mathbf{x})$ of the symmetric matrix $[\mathbf{II}]$, known as the *principal curvatures*, are the maximum and minimum values of normal curvature at $\mathbf{x}$. Since $\kappa_{n,\max}(\mathbf{x}) = \max(|\kappa_1(\mathbf{x})|, |\kappa_2(\mathbf{x})|)$ is required to define the curvature mesh size (1), our goal is to build an approximation of $[\mathbf{II}]$ at each vertex of the surface mesh.

The idea proposed by Rusinkiewicz[19] is to first compute $[\mathbf{II}]$ on the triangles, and average them over adjacent triangles to obtain the needed per-vertex information. We start by computing per-vertex normal vectors $\mathbf{n}_i$ by averaging the normals of triangles adjacent to each vertex using arithmetic mean. On each triangle, an arbitrary orthonormal coordinate system $(\mathbf{u}_f, \mathbf{v}_f)$ is then defined. The components of the quadratic form $\mathbf{II}$ in that basis read

$$[\mathbf{II}] = \begin{pmatrix} \mathbf{II}(\mathbf{u}_f, \mathbf{u}_f) & \mathbf{II}(\mathbf{u}_f, \mathbf{v}_f) \\ \mathbf{II}(\mathbf{u}_f, \mathbf{v}_f) & \mathbf{II}(\mathbf{v}_f, \mathbf{v}_f) \end{pmatrix}, \tag{3}$$

and can be evaluated as $\mathbf{II}(\mathbf{u}, \mathbf{v}) = L(\mathbf{u}) \cdot \mathbf{v}$ where the *shape operator* $L(\boldsymbol{s}) = \nabla_{\boldsymbol{s}}\mathbf{n}$ is the directional derivative of the normal vector $\mathbf{n}$ along a direction $\boldsymbol{s}$ in the tangent plane. Along the particular direction in the plane $(\mathbf{u}_f, \mathbf{v}_f)$ given by the triangle edge $\boldsymbol{e}_0 = (\boldsymbol{e}_0 \cdot \mathbf{u}_f)\mathbf{u}_f + (\boldsymbol{e}_0 \cdot \mathbf{v}_f)\mathbf{v}_f$ connecting vertices $\mathbf{x}_1$ and $\mathbf{x}_2$, one has the following finite difference approximation:

$$\mathbf{II}(\boldsymbol{e}_0, \mathbf{u}_f) = L(\boldsymbol{e}_0) \cdot \mathbf{u}_f = \nabla_{\boldsymbol{e}_0}\mathbf{n} \cdot \mathbf{u}_f = (\mathbf{n}_2 - \mathbf{n}_1) \cdot \mathbf{u}_f, \tag{4}$$

where the directional derivative $\nabla_{\boldsymbol{e}_0}\mathbf{n}$ in Rusinkiewicz[19] is approximated by the difference of normal vectors along $\boldsymbol{e}_0$. On the other hand, since the second fundamental form is a bilinear form, we have

$$\mathbf{II}(\boldsymbol{e}_0, \mathbf{u}_f) = \mathbf{II}\left((\boldsymbol{e}_0 \cdot \mathbf{u}_f)\mathbf{u}_f + (\boldsymbol{e}_0 \cdot \mathbf{v}_f)\mathbf{v}_f, \mathbf{u}_f\right) = (\boldsymbol{e}_0 \cdot \mathbf{u}_f)\ \mathbf{II}(\mathbf{u}_f, \mathbf{u}_f) + (\boldsymbol{e}_0 \cdot \mathbf{v}_f)\ \mathbf{II}(\mathbf{v}_f, \mathbf{u}_f). \tag{5}$$

Combining (4) and (5) and proceeding the same way for $\mathbf{v}_f$ yields the following equality for the components of $[\mathbf{II}]$:

$$[\mathbf{II}] \begin{pmatrix} \boldsymbol{e}_0 \cdot \mathbf{u}_f \\ \boldsymbol{e}_0 \cdot \mathbf{v}_f \end{pmatrix} = \begin{pmatrix} (\mathbf{n}_2 - \mathbf{n}_1) \cdot \mathbf{u}_f \\ (\mathbf{n}_2 - \mathbf{n}_1) \cdot \mathbf{v}_f \end{pmatrix}. \tag{6}$$

Repeating the same procedure for the two remaining edges $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$, one ends up with a system of six equations for three unknowns, which can be solved using a least square method to obtain the matrix representation of $\mathbf{II}$ for the triangle. This per-triangle representation of $\mathbf{II}$ is expressed in the local basis $(\mathbf{u}_f, \mathbf{v}_f)$. To combine the contributions of triangles adjacent to a vertex p, the orthonormal basis $(\mathbf{u}_p, \mathbf{v}_p)$ is defined in the plane perpendicular to the normal vector at p. Denoting $(\mathbf{u}'_f, \mathbf{v}'_f)$ as the basis $(\mathbf{u}_f, \mathbf{v}_f)$ rotated to be made coplanar with $(\mathbf{u}_p, \mathbf{v}_p)$, the components of the triangle contribution to the per-vertex representation $[\mathbf{II}]_p$ can then be obtained as

$$e_p = \mathbf{u}_p^T\ [\mathbf{II}]\ \mathbf{u}_p = \begin{pmatrix} \mathbf{u}_p \cdot \mathbf{u}'_f \\ \mathbf{u}_p \cdot \mathbf{v}'_f \end{pmatrix}^T [\mathbf{II}] \begin{pmatrix} \mathbf{u}_p \cdot \mathbf{u}'_f \\ \mathbf{u}_p \cdot \mathbf{v}'_f \end{pmatrix}, \qquad f_p = \mathbf{u}_p^T\ [\mathbf{II}]\ \mathbf{v}_p, \qquad g_p = \mathbf{v}_p^T\ [\mathbf{II}]\ \mathbf{v}_p. \tag{7}$$

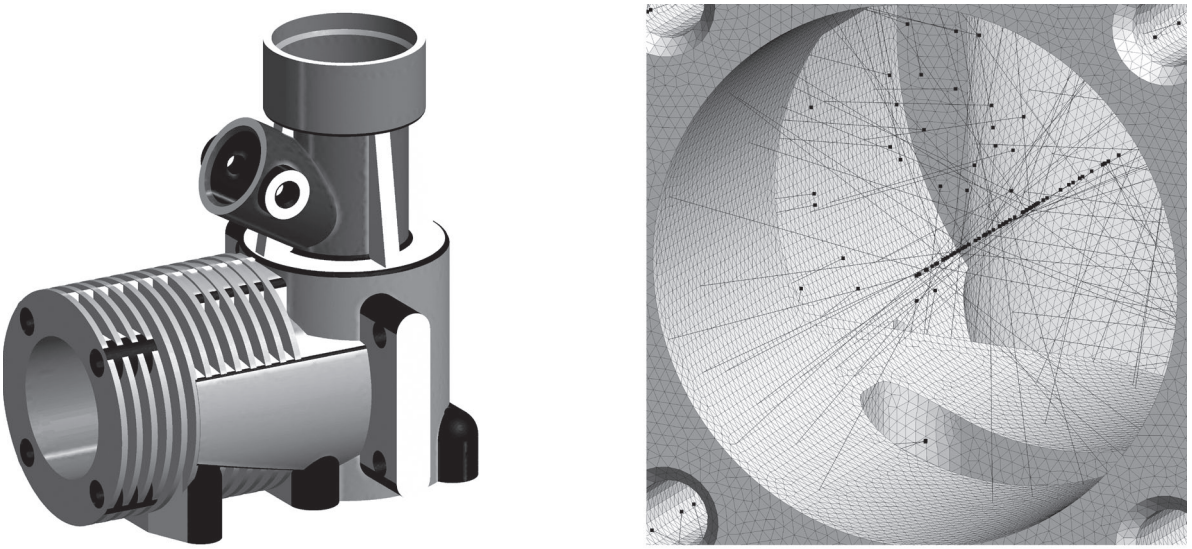
As for the normal vectors, the contributions of triangles adjacent to the vertex p are then averaged to obtain $[\mathbf{II}]_p$. Once again, arithmetic mean is used to compute the average tensor, that is, all triangles have an equal contribution regardless of their area.

Once the matrix representation of the second fundamental form $[\mathbf{II}]_p$ has been computed at a vertex, its eigenvalues are extracted and the curvature mesh size h_c at the vertex is set using (1).

The computed maximal curvature for the engine block model is shown below (Figure 2(A)): curvature is computed on each colored face of the triangulation, discounting the feature edges.

2.2 | Feature size

Whenever two surfaces with moderate or no curvature are close to each other but the distance between them is smaller than the mesh size, the mesh generator will place only one element in the gap between those surfaces. In many engineering applications like solid mechanics or fluid mechanics, having only one element in a gap means that both sides



(A) (B)

FIGURE 2 Left: Approximated maximum normal discrete curvature $\kappa_n = \max(|\kappa_1|, |\kappa_2|)$ of the surface triangulation, shown in grayscale: Curved areas are in gray-black while regions with mild curvature are in light gray. Right: A subset of the Voronoï vertices lying the farthest from the Delaunay vertices, called *poles* (black dots), approximate the medial axis as the mesh density increases

are connected by one single mesh edge. If Dirichlet boundary conditions are applied, such as a nonslip boundary condition, the gap is then essentially closed, leading to an unwanted change of the domain topology. Note that a posteriori error estimation will not detect large errors in those closed gaps where the solution is essentially constant.

Hence, special care should be given to such narrow geometrical regions, where curvature information alone is not enough to determine a suitable mesh size field. To this end, we define the *feature size* $f(\mathbf{x})$ as a measure of the local gap thickness (Figure 3(A)). If ∂V is the boundary of a volume V, $f(\mathbf{x})$, $\mathbf{x} \in \partial V$, is defined as twice the distance between $\mathbf{x}$ and the *medial axis* of volume V. If a surface bounds two volumes, then the minimum feature size is chosen. The *feature mesh size* at the considered vertices, now, is the feature size divided by the desired number of element layers in narrow regions n_g, that is, $h_f(\mathbf{x}) = f(\mathbf{x})/n_g$. The feature mesh size h_f is thus in a similar relationship to the feature size f than the curvature mesh size h_c was to the maximum principal curvature κ.

The evaluation of the feature size $f(\mathbf{x})$ requires computing an approximation of the medial axis of all volumes in the computational domain. The medial axis of a volume V, also referred to as its *skeleton*, is defined as the set of points having more than one closest point on its boundary ∂V. Equivalently, the medial axis is the set of the centers of all spheres tangent to ∂V in two or more points, and the feature size $f(\mathbf{x})$ is twice the radius of the sphere tangent at $\mathbf{x}$.

We rely on the algorithm MEDIAL introduced by Dey and Zhao[20] to compute a discrete approximation of the medial axis. The input data for MEDIAL is a Delaunay tetrahedrization of the vertices of the surface mesh only, that is, a set of tetrahedra, often called *empty mesh*, filling the computational domain and whose nodes all lie on the surfaces. This algorithm is based on the Voronoï diagram of these vertices and has suitable convergence properties, in the sense that the output set of facets converges to the medial axis as the surface mesh density increases. In this work, we implemented MEDIAL as is. For sake of completeness, we here briefly remind the rationale behind the algorithm, using their notations.

In 2D, the vertices of the Voronoï cells dual to the empty mesh give straightaway an approximation of the medial axis. The same is however not always true in 3D, because sliver tetrahedra of the empty mesh can persist close to the boundary as the surface mesh is refined. However, by pruning as explained below the Voronoï vertices dual of these sliver tetrahedra, a subset of Voronoï vertices called *poles* can be defined that do approximate the medial axis (Figure 2(B)). Given the Voronoï cell dual of a surface vertex p (Figure 3(A)), the corresponding pole p^+ is defined as the Voronoï vertex that is the most distant from p. Each surface vertex p is thus associated with a pole p^+ and a *pole vector* $v_p = p^+ - p$, the latter approximating the normal to ∂V at p (Figure 3(B)).

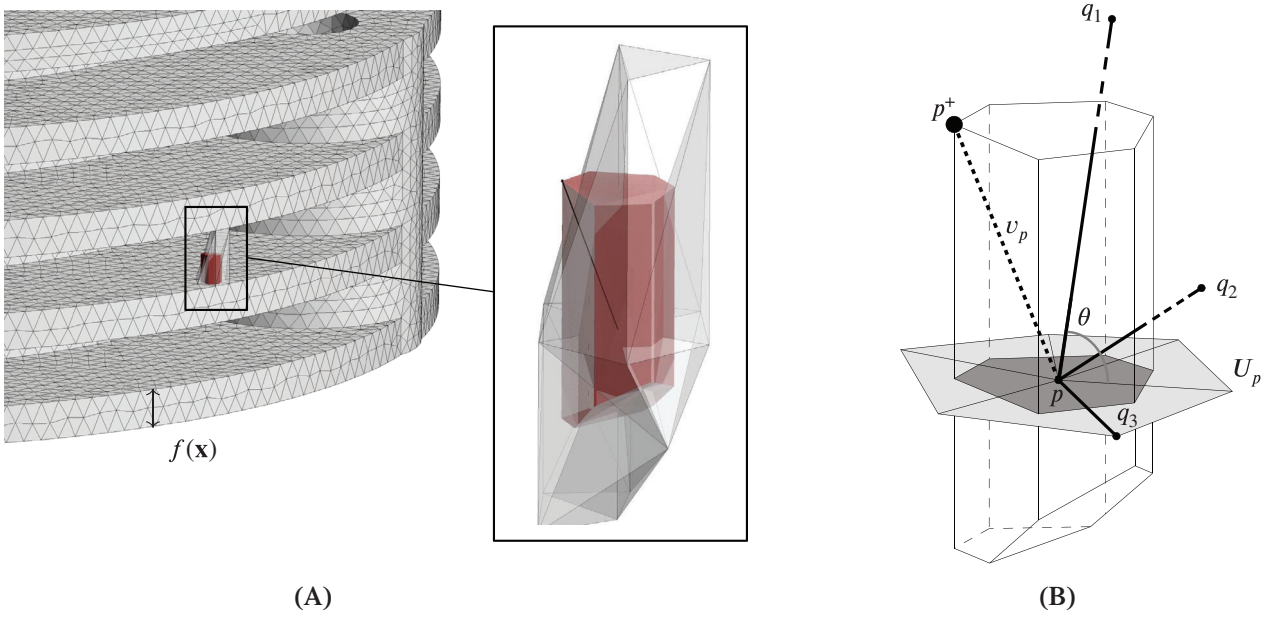


FIGURE 3 Left: View of the input triangular surface mesh and of the Delaunay tetrahedra adjacent to a mesh vertex p. Zoom in on the same tetrahedra (transparent) and the associated dual Voronoï cell (in red). Right: Voronoï cell dual to the Delaunay vertex p in the triangular surface mesh. The pole vector v_p (dashed) connects the mesh vertex p to the farthest Voronoï vertex (tetrahedron circumcenter) or *pole* p^+. The Delaunay edges pq_1, pq_2, and pq_3 connect p to neighboring Delaunay vertices q_1, q_2, and q_3, respectively. Only pq_1 and pq_2 satisfy one of the filtering conditions and are considered to compute the feature size. In gray, the umbrella U_p of p is the Delaunay facets (triangles) dual to the Voronoï edges cut by the plane through p with normal v_p (not shown)

The plane passing through p with normal v_p intersects edges of the Voronoï cell, and the Delaunay facets dual to these edges are pictorially called the *umbrella* U_p *of* p by Dey and Zhao (Figure 3(B), in light gray).

Triangles of the umbrella are used to select some of the Delaunay edges adjacent to p whose dual facets will eventually form the discrete medial axis. The idea is to select Delaunay edges pq that (a) make a sufficiently large angle with the triangles of the umbrella, or (b) are significantly longer than the circumradius of these triangles. Condition (a) measures how normal the edge pq is to the umbrella U_p and is referred to as the *angle condition*. The edge should make an angle larger than θ with each triangle of the umbrella, or conversely, an angle smaller than $\pi/2 - \theta$ with their normal vector. In practice, we evaluate:[20]

$$\max_{i \in U_p} \angle(\mathbf{pq}, \hat{\mathbf{n}}_i) < \frac{\pi}{2} - \theta, \tag{8}$$

where $\mathbf{pq}$ is a vector parallel to the edge pq, i denotes a triangle in U_p and the threshold angle is set to $\theta = \pi/8$. Condition (b) ensures that edges of the surface mesh are removed. It does so by selecting long edges for which the angle condition has failed, and is referred to as the *ratio condition*. In practice, only edges at least $\rho = 8$ times longer than the circumradius R of the triangles in U_p are considered, and the condition reads:[20]

$$\max_{i \in U_p} \frac{\|pq\|}{R_i} > \rho. \tag{9}$$

The numerical values for parameters θ and ρ are those suggested by Dey and Zhao. They yield a good approximation of the medial axis for a large variety of geometries. If a Delaunay edge pq satisfies either of these two conditions, its dual Voronoï facet is added to a set F, forming the approximate medial axis. Consider the three mesh vertices q_i connected to p through a Delaunay edge $pq_i = q_i - p$ on Figure 3(B). Edge pq_1 makes a large angle θ with the triangles of the umbrella, and edge pq_2 is several times longer than the largest circumradius: both are added to the set E dual to F. Edge pq_3 is a short surface edge mesh lying flat to the umbrella, and is thus removed from the list of candidate edges.

In our size field computation, we need not compute the dual facet to edges in E: for each Delaunay edge pq satisfying the angle or ratio conditions, the local feature size is directly given by the edge length $\|pq\|$. Mesh size at both vertices p and q is thus defined as the edge length divided by the desired number of element layers in features:

$$h_f(\mathbf{x}_p) = h_f(\mathbf{x}_q) = \frac{\|pq\|}{n_g}. \tag{10}$$

The mesh size on the octants containing these vertices is then lower bounded if necessary:

$$h = \max\left(h_{\min}, \min\left(h_f, h_c, h_u, h_b\right)\right). \tag{11}$$

The medial axis also contains the centers of spheres with radius vanishing to zero in angles and corners of the volume (Figure 4(A)). Dual edges in these corners are smaller than the feature size one wishes to identify, and should thus be disregarded to avoid spurious small mesh size in these areas. In practice, an edge pq making an angle larger than θ with either one of the normal vectors $\hat{\mathbf{n}}_p$ and $\hat{\mathbf{n}}_q$ at its ends is also filtered out (Figure 4(B)).

The quality of the approximation of the medial axis is directly related to the node density of the surface mesh. As pointed out by Dey and Zhao, the surface mesh should be a ε-sample, that is, vertices should have neighbors within a distance $\varepsilon f(\mathbf{x})$, where $f(\mathbf{x})$ is the feature size and ε is small. Of course, as the aim is here precisely to compute the feature size, it is not known beforehand whether or not the input surface mesh is a ε-sample. A first solution is to measure beforehand the most critical feature size $f_{\text{crit}} = \min_{\mathbf{x}} f(\mathbf{x})$ of the CAD model, then generate a uniform mesh with constant mesh size $h_{\text{crit}} = \varepsilon f_{\text{crit}}$, typically with $\varepsilon \leq 0.25$ as suggested in Reference 20. The characteristic size of the input surface mesh is then constrained by the smallest feature in the geometry, which may result in an expensive size field computation. While we could certainly adjust the mesh size of the input mesh to be h_{crit} only in the small features, this would amount to manually specify the mesh size field, which we want to avoid. To circumvent this, one can compute an initial size field based on a reasonably fine uniform mesh, then perform the mesh generation from this field. This intermediary mesh will include an initial refinement in the small features, and will be a better candidate for the final size field computation.

The resulting medial axis for our block example after both filtering operations (i.e., conditions (8) and (9), as well as removing the spurious branches in corners) is shown on Figure 5(A). Taking into account the local feature size allows

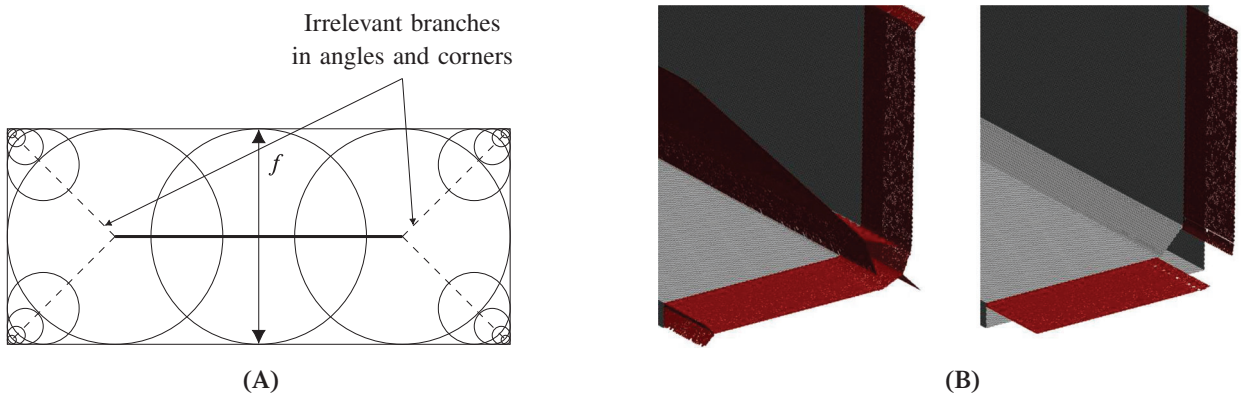


FIGURE 4 Left: The medial axis of a rectangle consists of a main branch (thick) and four secondary branches (dashed) connecting the main branch to the corners. The radii of the spheres whose center lies on the main branch are representative of the feature size f (here, the thickness of the rectangle), while the radii of the spheres on the secondary branches shrink to zero as the branch approaches the surface, and are thus not representative of f. To remove those secondary branches (Voronoï facets), the dual Delaunay edges are applied a second filtering process ensuring that the angle between the edge and the normal vectors at its extremities does not exceed θ. Right: the medial axis (in red) of an angle geometry before filtering the branches in the corners (left) and the medial axis after filtering (right)

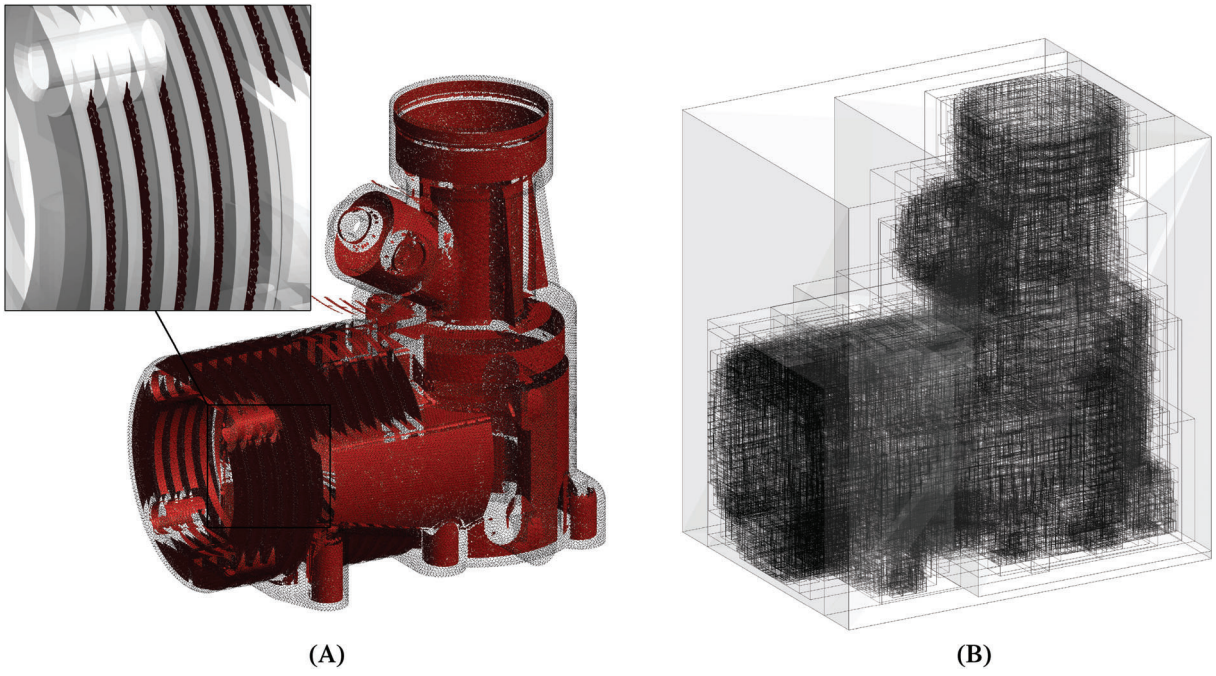


FIGURE 5 Left: Approximated medial axis of the CAD model. The Voronoï facets dual to the filtered Delaunay edges are drawn in red. In the close-up view, only the facets of the medial axis lying outside of the volume are shown. Right: The RTree structure built from the bounding boxes of the triangles of the surface mesh

for a refined mesh in small features of the geometry, especially in areas with zero curvature, which would be overlooked otherwise (Figure 6, see also Figure 10 for the volume mesh).

2.3 | Octree initialization and refinement

The mesh size field $h(\mathbf{x})$ over the domain to be meshed is now built as an octree structure. The initial octant is defined as the axis-aligned bounding box of the surface mesh, stretched in all three dimensions by a factor 1.5. In order to ensure a

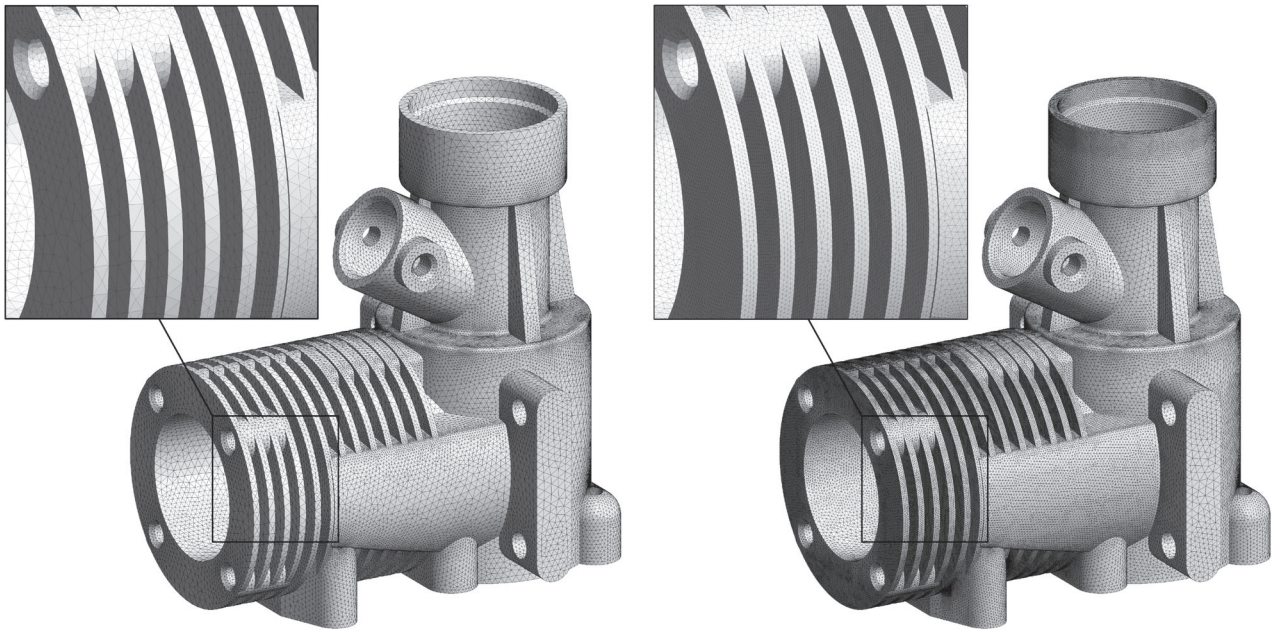


FIGURE 6 Surface mesh generated from the computed mesh size field: from curvature only (left) and considering both curvature and feature sizes (right). On the right, four layers of elements are generated in the fins around the largest cylinder

suitable gradation in the mesh, the guiding principle for the refinement of the octree is that the dimension of each octant should eventually be representative of the local mesh size. The octree is then first subdivided recursively and uniformly until the size of each octant is at most the bulk size h_b, and the local mesh size assigned to the octants created during this initial step are set to h_b. The octree is then further refined on basis of the curvature mesh size h_c and the feature mesh size h_f. This information, which is available on the surfaces, has to be transferred to the octree, which is a 3D structure. One needs for that to detect efficiently the intersections between octants and the surface mesh. To this end, the 3D bounding box of each triangle of the surface mesh is added to an R-Tree,[21] a data structure used for spatial access methods (Figure 5(B)) which acts here as the intermediary between the geometry and the octree. More specifically, the R-Tree provides for each octant of index i a list $\mathcal{T}_i$ of triangles whose bounding box intersects the octant. The octant with index i is then divided until it becomes smaller than the minimal mesh size (h_c or h_f) at the vertices of all triangles in $\mathcal{T}_i$. Whenever a user-defined mesh size function $h_u = u(\mathbf{x})$ is provided, this additional constraint is taken into account at this level. The octant size being bounded from below by the user-defined minimal mesh size $h_{\min}$, octants are thus subdivided until the condition

$$h_{\text{octant}} \leq \max\left(h_{\min}, \min\left(h_c, h_f, h_u, h_b\right)\right) \tag{12}$$

is met everywhere in the octree.

Once this refinement is completed, the octree is balanced to ensure a maximum 2:1 level ratio between two octants across adjacent faces (Figure 7), that is, the levels of two octants on each side of a face should not differ by more than one unity. This balancing is necessary to obtain suitable stencils for finite difference computations during the gradient limiting step, see Section 2.4. It is performed by P4EST. Newly created octants having an intersection with the surface mesh are assigned the mesh size determined by (12), otherwise the size is set to the bulk size h_b (Figure 9(A)). At this stage, the size field features large variations, and a limitation step is required to end up with a mesh size field suitable for high-quality mesh generation.

2.4 | Size limitation

Large mesh size gradients may be the cause of low-quality finite element solutions. In solid mechanics, for example, they may be the cause of excessive values of strains when a very small element is adjacent to a large one. One of the goals of our

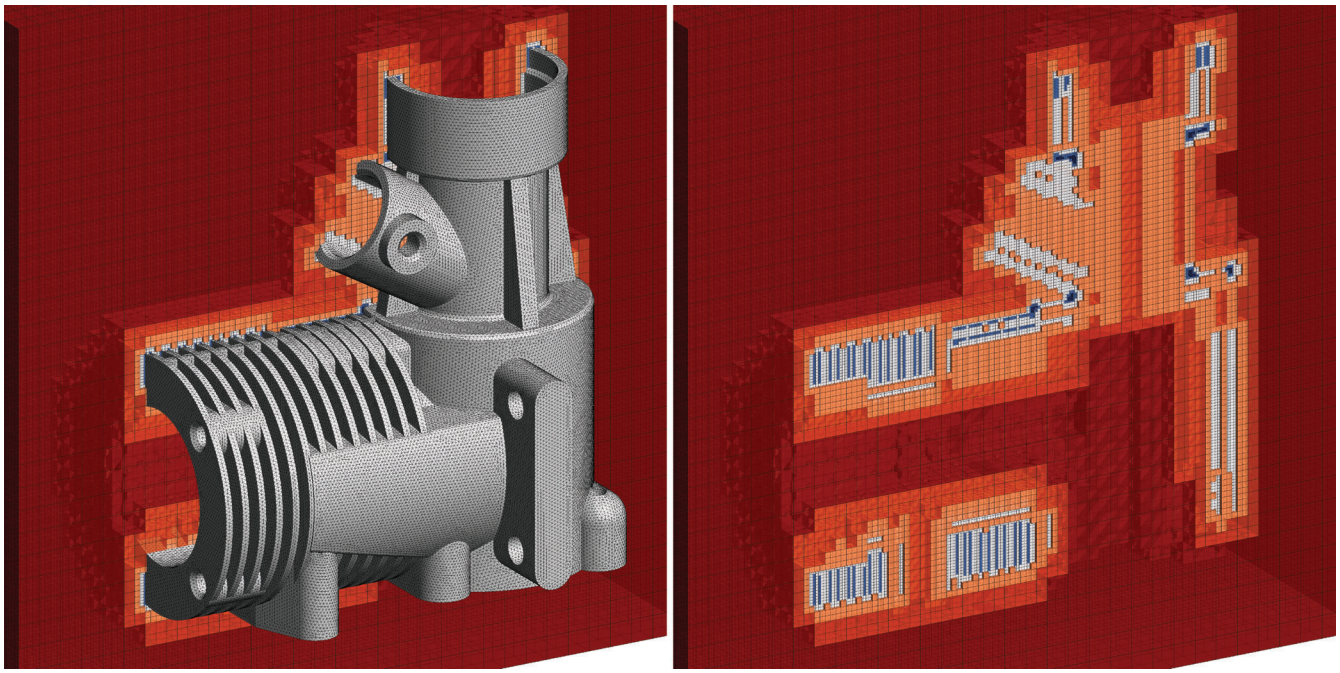


FIGURE 7 Octree after refinement: The color of each octant signifies its refinement level, from coarse (red) to fine (blue)

approach is to ensure that two adjacent edges in the mesh have their length ratio below a user-defined gradation $\alpha > 1$, that is, the length ratio along a direction follows a geometric progression. We briefly show here that such a geometric progression is achieved by limiting the size gradient along a given direction by $\alpha - 1$:

$$\left|\frac{\partial h}{\partial x}\right| \leq \alpha - 1, \tag{13}$$

that is, a constant gradation yields a geometric progression of edges length and not a linear progression as one could expect at first glance. To see this, let us consider a one-dimensional (1D) mesh in the direction x along which the size gradient is maximal. Let us denote by $\mathbf{x}_i, i = 0, \ \ldots \ n$ the vertices of the mesh and $h_i = h(\mathbf{x}_i)$ the length of the edge starting at $\mathbf{x}_i$. As the edge lengths follow a geometric progression of ratio α, we have $h_i = \alpha^i h_0$. The x-coordinate of vertex $\mathbf{x}_n$ is given by:

$$x_n = \sum_{i=0}^{n-1} h_i = h_0 \ (1 + \alpha + \alpha^2 + \ \ldots \ + \alpha^{n-1}) = h_0 \ \frac{\alpha^n - 1}{\alpha - 1} \tag{14}$$

from where follows:

$$\alpha^n h_0 = h_0 + x_n(\alpha - 1). \tag{15}$$

Using $h_n = \alpha^n h_0$, one ends up with:

$$h_n = h(x_n) = h_0 + x_n(\alpha - 1), \tag{16}$$

showing that the size field $h(x)$ is a linear function of space, despite the fact that the edge lengths are in a geometric progression. The derivative $\partial h/\partial x$ is hence constant, with magnitude $\alpha - 1$. Locally, since $\alpha = h_{i+1}/h_i$, we also have the following finite difference approximation for the size gradient:

$$\left|\frac{\partial h}{\partial x}\right| = \frac{h_{i+1} - h_i}{\frac{1}{2}(h_{i+1} + h_i)} = \frac{2(\alpha - 1)}{\alpha + 1}, \tag{17}$$

which is close to (18) for typical values of $\alpha \in [1, 1.4]$. Limiting the size derivative along any direction thus amounts to limit the 2-norm of the size gradient, that is

$$\|\nabla h(\mathbf{x})\|_2 \leq \alpha - 1. \tag{18}$$

A similar computation by Chen et al.[4] leads to a third condition $\|\nabla h(\mathbf{x})\|_2 \leq \ln \alpha$, also close to (18) for the considered range of gradation, as we have $\ln(1 + (\alpha - 1)) \simeq \alpha - 1$.

As pointed out by Persson,[10] the 1D analysis discussed above is not sufficient to ensure that the expected mesh size gradation condition is fulfilled all over the mesh. Indeed, when storing the size field $h(\mathbf{x})$ in a background mesh, the expected gradation is ensured only along the edges of the elements of the background mesh, and larger size variations might be encountered in the interior of the elements. To also constrain size gradients inside the elements of the background mesh, it is then necessary to iterate over the edges of the background mesh to modify appropriately the size stored at the nodes.[4,5] This results in a size field $h(\mathbf{x})$ that eventually satisfies (18), but in a background mesh dependent fashion.[10] This drawback is avoided with our methodology because the mesh size function is rather stored in a balanced octree. Condition (18) is satisfied everywhere in the volume, and thus also in the interior of the elements, as a result of the subdivision of the octants until convergence.

Gradient computation by finite difference. Condition (18) requires to limit the size derivative along all directions, which is a costly operation. In practice, we limit the size variation along the directions of the axes, that is, the directions of the octree, and condition (13) is enforced using a finite difference scheme. This amounts to impose

$$\|\nabla h(\mathbf{x})\|_\infty \leq \alpha - 1, \tag{19}$$

a looser requirement resulting in directions along which the gradation may only be controlled up to a factor $\sqrt{3}$ in the worst case. As the results show however, limiting the ∞-norm yields high-quality meshes for all the considered CAD models, with an average discrete size gradation matching the required gradation, see Section 3. We iterate over the octants until

$$\left|\frac{\partial h}{\partial x_k}\right| \leq \alpha - 1, \tag{20}$$

is verified everywhere, with $k = 1, \ldots, 3$. For each octant, we take advantage of the 2:1 balancing provided by P4EST to compute the gradient with a cell-centered finite difference scheme. Since the octree is balanced, only three stencils must be considered to approximate the gradient in all cases. With the P4EST terminology, one side of a face between two octants is said to be either full (F) or hanging (H), depending on whether the octant on that side is a leaf or is itself subdivided. The stencils to be considered (Figure 8) are then FFF, FFH, HFH, since the all-hanging case (HHH) can be regarded as multiple all-full (FFF) stencils. To evaluate (20) at the center of the middle octant along, say, the x-direction, we use the Taylor expansion:

$$\left.\frac{\partial h}{\partial x}\right|_i = \frac{1}{2}\left(\frac{\overline{h}_{i+1} - h_i}{\Delta x_{i+1} + \Delta x_i} + \frac{h_i - \overline{h}_{i-1}}{\Delta x_i + \Delta x_{i-1}}\right), \tag{21}$$

where $i-1$, i, and $i+1$ represent the three positions in the considered stencil, $\overline{h}_i$ denotes the arithmetic average of the mesh size of the four octants adjacent to the hanging face i, and Δx_i is the half-length of the octants in the position i. For the all-full stencil FFF, one has $\overline{h}_i = h_i$ and the Δx_i are all equal to the half length of the octants, so that the approximation (21) reduces then to the usual second-order centered scheme.

Size limitation. The computed gradient is then used to limit the mesh size field. For any two adjacent octants with gradient larger than $\alpha - 1$, the larger is reduced so as to satisfy condition (18). More precisely, if h_1 and h_2 denote the mesh size in the octants with $h_2 \geq h_1$, h_2 is modified as

$$h_2 = \min(h_2,\ h_1 + \Delta x(\alpha - 1)). \tag{22}$$

This expression is an approximation of the solution to the steady-state equation proposed by Persson[10] in his continuous formulation of the gradient limiting problem, ensuring mesh size limitation is propagated in the direction of

increasing values. Smaller mesh sizes are left unchanged by the limitation process, preserving the sharp features of the geometric model. The limitation is performed iteratively in the three directions, until condition (18) is satisfied everywhere in the octree (Figure 9(B)).

2.5 | Size query in the octree

Size queries to evaluate the size at different locations on parametrized curves, surfaces and volumes during the meshing process are performed by Gmsh. The implementation of the query routines in the octree is provided by P4EST. Since both

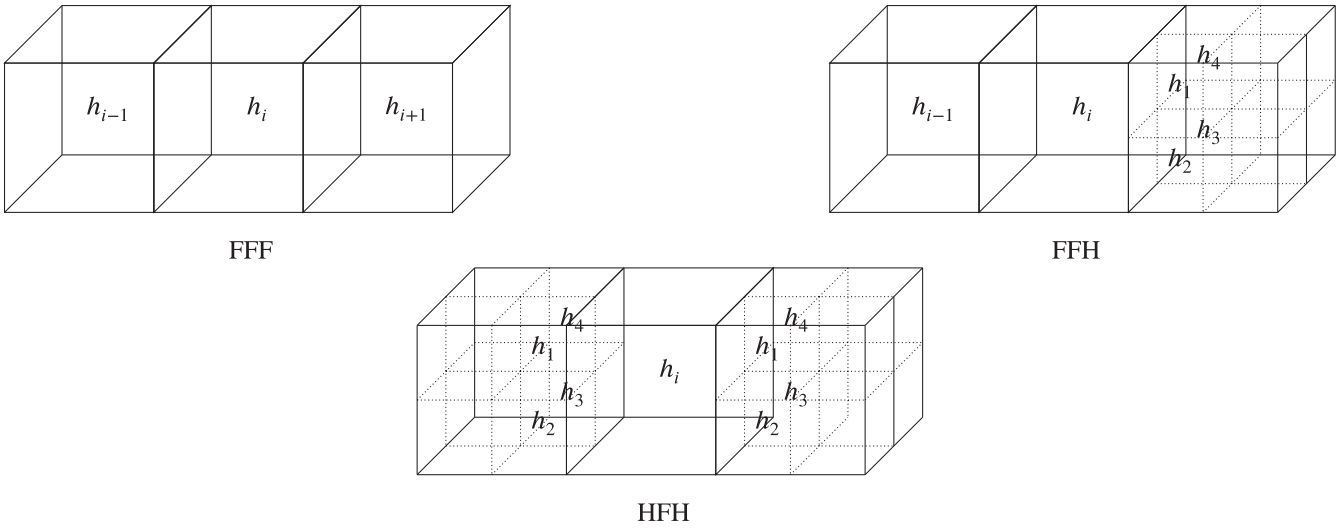


FIGURE 8 Finite difference stencils to compute ∇h in the octree

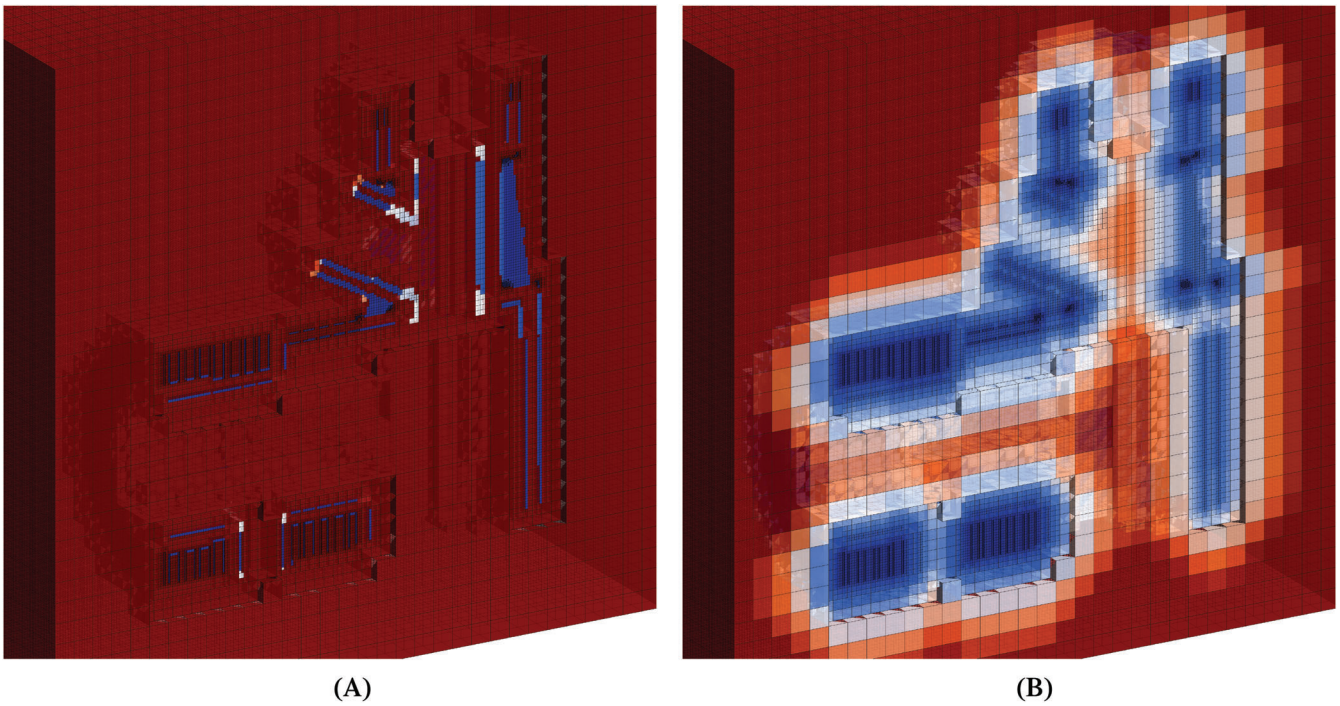


FIGURE 9 Limiting the mesh size stored in the octants: (A) Initial mesh size field computed from curvature and feature size, assigned in octants intersecting the surface mesh (B) size field after limitation

the size $h(\mathbf{x})$ and its gradient are known in each octant, the mesh size at the query point $\mathbf{x}$ is evaluated using a first-order Taylor expansion:

$$h(\mathbf{x}) = h_i + \nabla h \cdot (\mathbf{x} - \mathbf{x_c}), \tag{23}$$

where h_i and $\mathbf{x}_c$ denote the size and the coordinates at the center of the octant, respectively.

3 | RESULTS

The proposed algorithm was applied to a variety of CAD models accessed from GrabCAD (https://grabcad.com/), the ABC Dataset model library[22] and Gmsh's benchmarks suite. For all geometries, the same set of parameters was used with values given in Section 1, that is, $n_d = 20$ elements in curved areas, $n_g = 4$ elements in thin layers and a mesh gradation of $\alpha = 1.1$. Note that those default parameters were chosen with what we consider to be common sense, and not after a series of trial and error: for example, discretizing a circle with 20 line segments is what is used by default to graphically represent a circle in Gmsh. The number of layers $n_g = 4$ was set with bending structures computations in mind, where it is known that having less than four layers leads to overestimating the stiffness of the structures. The gradation of 1.1 originates from regular discussions with CFD practitioners: this value can be set higher for computational mechanics applications for example, yet, it should not exceed 1.7–1.8, unless the generated mesh serves for visualization purposes. As shown in this section, those intuitive parameters yield quality meshes for a vast majority of the considered models. In each case, we generate a triangulation of the CAD model with a prescribed uniform mesh size. The surface mesh is the input of the algorithm to compute a mesh size field on all volumes. Surface curvatures and the approximate medial axis are then estimated, and the mesh size field $h(\mathbf{x})$ is computed as described in the previous sections. The octree is stored on disk in the native P4EST format on creation, and is then loaded as a background field in Gmsh.

This sections aims at demonstrating the efficiency and accuracy of our approach. As far as efficiency is concerned, the CPU time dedicated to queries in the octree is distinguished from the time devoted to the construction of the octree. We then compare the sum of those two times to the meshing time. In particular, we show that the proposed size field construction does not change the order of magnitude of the meshing time, that is, seconds remain seconds and do not become minutes. We then analyze how accurately the prescribed parameter are reached in the mesh generated from the size field. To this end, a discrete counterpart of the gradation is compared with the parameter α, and the number of elements in small gaps is checked to be close to n_g. An efficiency index is then introduced for each of the generated mesh: this index measures how close the resulting mesh is to a unit mesh and thus expresses the adequacy between the size field and the mesh.

All computations are run on a laptop with Intel Core i7 8750h CPU (2.2 GHz) and 16 GB memory, and execution times for two selected test cases can be found in the table hereafter.

3.1 | Surface and volume meshing

Engine block. We first consider the engine block test case depicted in Figure 2(A). The geometry, while presenting a relatively low ($\approx$ 500) number of faces, features entities of variable radii of curvature as well as thin areas, and is an interesting application example for our algorithm. The CAD model being an assembly of thin components, such as narrow fins surrounding the main cylinder, the feature mesh size is the dominant meshing criterion. Surface meshes were presented in Section 2, and show the impact of the feature mesh size (Figure 6) on the final mesh density. On the left part of Figure 6, the size field is constructed based only on curvature: one or two elements are generated in the narrow fins, whose mesh size is defined by the radius of curvature of the inner cylinder. On the right part of Figure 6, the feature mesh size h_c is also considered in the construction of the size field, and $n_g = 4$ element layers are then as expected generated in the fins. Figure 10 shows the tetrahedral mesh, where prescribed mesh size specifications are respected inside the volume. Since the spurious branches of the medial axis are removed in the rounded corners, mesh size in these locations is constrained by curvature. This results in a smaller size defined by the node density n_d (zooms 2 and 3). Mesh size in the thin cylinders and the fins is computed to fit four layers of elements (zooms 1, 4, 5, 6, and 7).

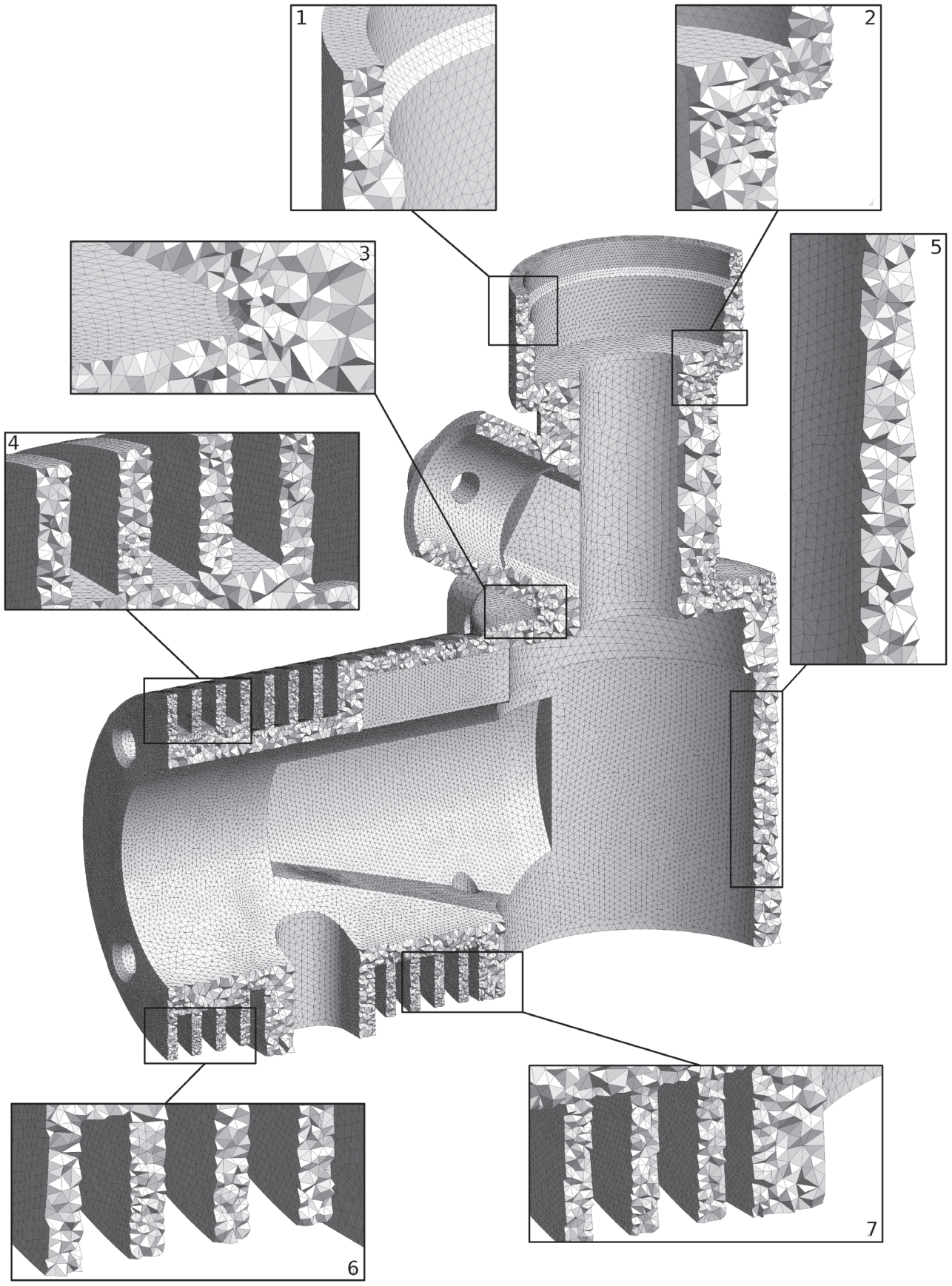


FIGURE 10 Final volume mesh of our worked-out example: The size field is computed with $n_d = 20$ nodes on the local osculating circles and $n_g = 4$ layers of elements in all geometric features. The mesh contains 583,776 nodes and 2,716,170 tetrahedra

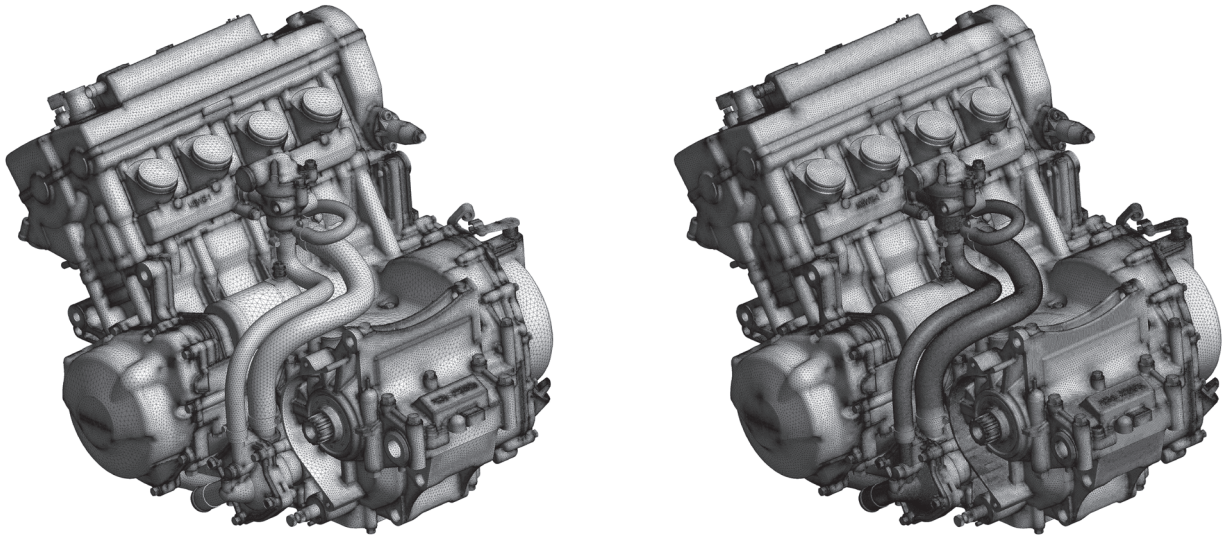


FIGURE 11 The surface mesh based only on curvature contains 1,521,410 nodes and 3,042,060 triangles (left); the mesh based on both curvature and feature size contains 2,302,630 nodes and 4,605,010 triangles (right)

Sport bike engine. The second test case is a four-cylinder engine of a Honda™ CBR600F4i sport bike[*]. The model is composed of thin areas such as pipes, gears and plates, and of regions of high curvature. The enclosing volume being not conformal (i.e., not "air-tight"), a volume mesh could not be generated without first repairing the CAD. Only surface meshes are thus presented for this example. The resulting surface meshes accurately capture areas of higher curvature (Figure 11, left) as well as small features (Figure 11, right and Figure 12), demonstrating the robustness of the algorithm on large CAD models.

ETA 6497-1 watch movement. This geometry[†] is a complete clockwork composed of flat plates and gears of different sizes. Feature size is the constraining factor on most flat pieces, while curvature determines the mesh size at, for example, gear teeth. The surface mesh and the volume mesh for this test case are, respectively, shown on Figures 13 and 14.

Space shuttle. This example was selected to illustrate specifically the influence of the node density n_d and of the gradation parameter α on the mesh. To show their influence on the whole model, including thin areas, feature size was not considered in this case, and only curvature was taken into account. As expected, larger values of n_d result in more elements on the local osculating circle to a surface, and hence in a finer mesh (Figure 15). Similarly, a gradation parameter α close to 1 limits the size gradient, also resulting in a globally finer and more homogeneous mesh (Figure 16).

Various geometries. Finally, our algorithm was also applied with the same parameters to a series of CAD model obtained from GrabCAD, the ABC Dataset library, and the Gmsh benchmarks library (Figure 19). In all cases, accurate size fields were computed within a few seconds to a few minutes, yielding smooth meshes suitable for numerical simulations. Note that feature size computation was not enabled for all models (e.g., for the lava lamp).

3.2 | Adequacy between the mesh and the size field

With the generated mesh at-hand, we define two quantitative estimates to assess its adequacy with the size field: an *efficiency index* measures how far the mesh is from a unit mesh in the isotropic metric field described by the size field, and a discrete counterpart to the gradation verifies that edge length progression is limited in the mesh. Finally, a standard isotropic quality indicator is presented.

Efficiency index. The resulting mesh should be as close as possible as a unit mesh in the metric field associated to the isotropic sizing function,[23] that is, the edges lengths should be as close as 1 when measured in the given metric. Since we are dealing with the discrete counterpart of this continuous mesh framework, we consider that the size specification

[*]https://grabcad.com/library/honda-cbr600-f4i-engine-1
[†]https://grabcad.com/library/eta-6497-1-complete-watch-movement

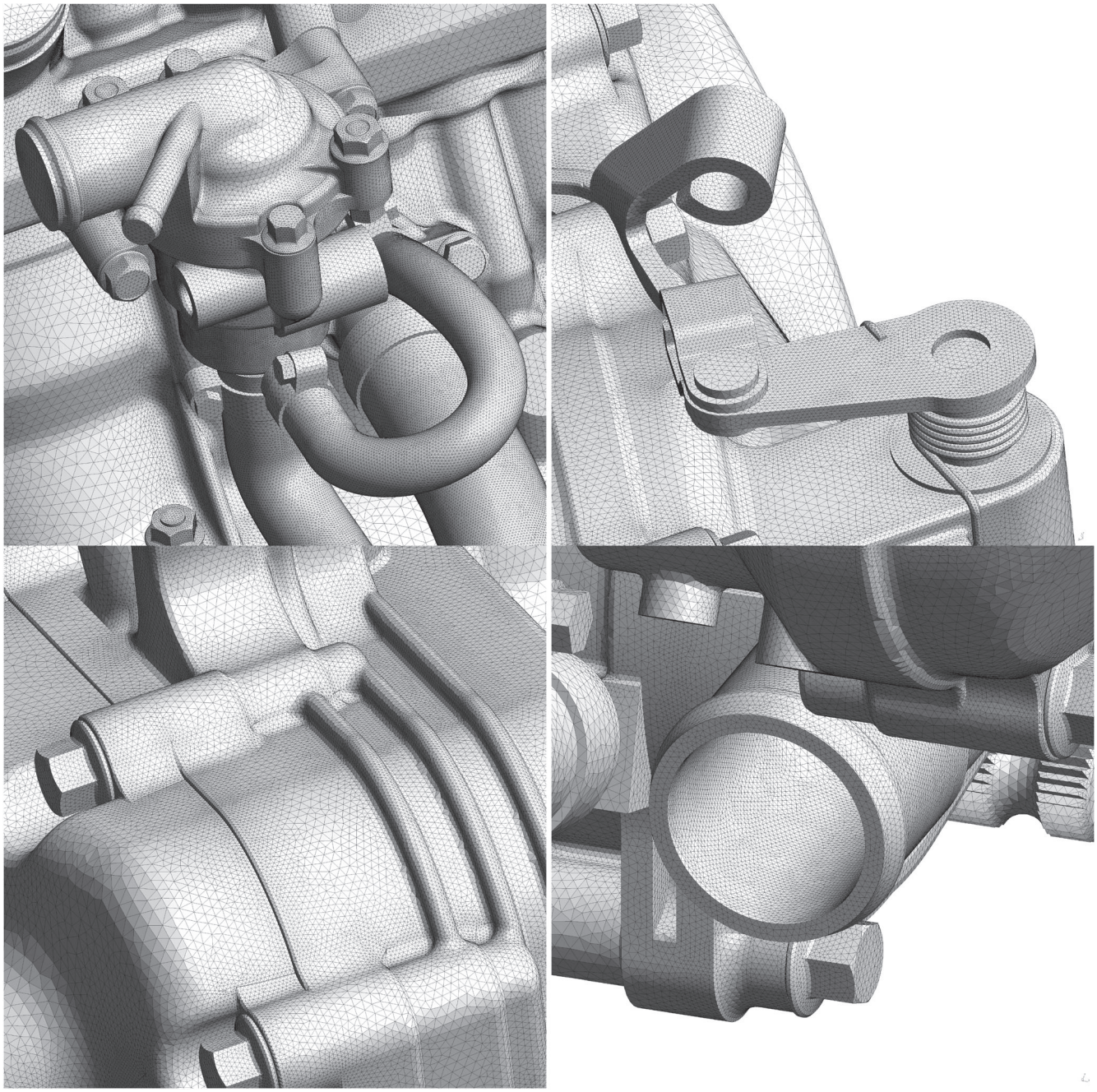


FIGURE 12 Zooms on selected parts of the Honda engine

is respected when the mesh is quasi-unit, that is, when the edges lie in $[\frac{1}{\sqrt{2}}, \sqrt{2}]$. To evaluate the adequacy between the generated mesh and the prescribed size field, we define an *efficiency index*. Let l_i be the length of an edge i computed in the local metric and n_e the number of edges in the mesh. The efficiency index τ of a mesh is defined as the exponential to one of all the edges length in the mesh, that is:

$$\tau = \exp\left(\frac{1}{n_e}\sum_{i=1}^{n_e} \bar{l}_i\right), \tag{24}$$

with $\bar{l}_i = l_i - 1$ if $l_i < 1$ and $\bar{l}_i = 1/l_i - 1$ if $l_i \geq 1$. The efficiency index thus lies in $]0, 1]$, the upper bound being reached for a unit mesh. We run our algorithm on close to two hundred CAD models and computed the efficiency index (Figure 17),

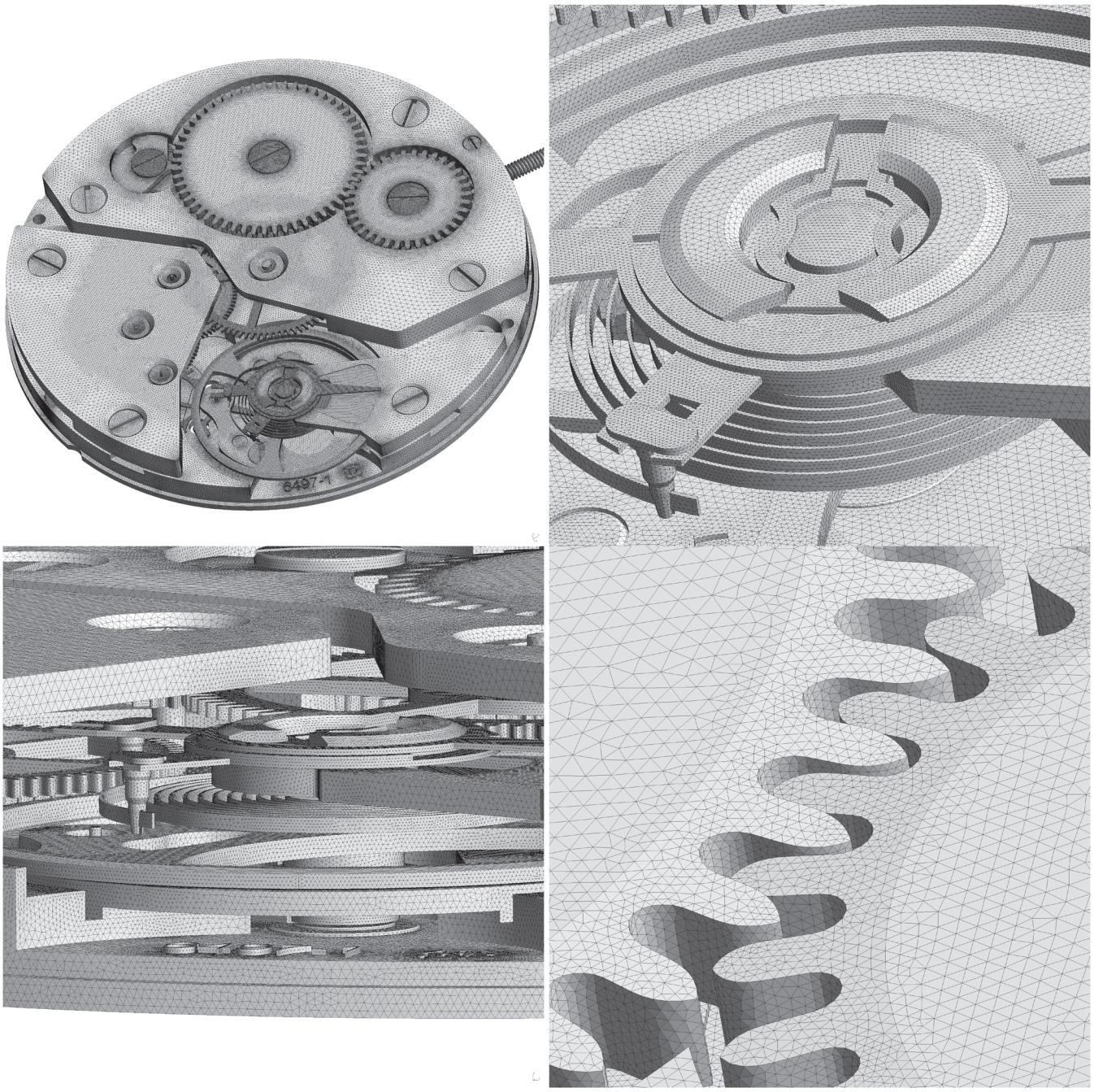


FIGURE 13 Zooms on selected parts of the surface mesh of the watch movement

whose median is slightly above 0.8. This shows that the size field given to the meshing tool is realistic, as it can generate almost unit edges for the given metric field.

Discrete gradation. The resulting mesh should feature a geometric progression in adjacent edges lengths with a ratio up to α. Instead of monitoring the maximum edge ratio at every vertices of the final mesh, we use a looser indicator defined on the edges. The *discrete gradation* $\alpha_{d,i}$ measures the progression between the average edges lengths at both vertices of each edge e_i. Let v_1 and v_2 denote the vertices of e_i and $l_{\text{avg}}(v)$ the average edge length at vertex v. We define the discrete gradation at edge e_i as follows:

$$\alpha_{d,i} = \frac{\max[l_{\text{avg}}(v_1), l_{\text{avg}}(v_2)]}{\min[l_{\text{avg}}(v_1), l_{\text{avg}}(v_2)]}. \tag{25}$$

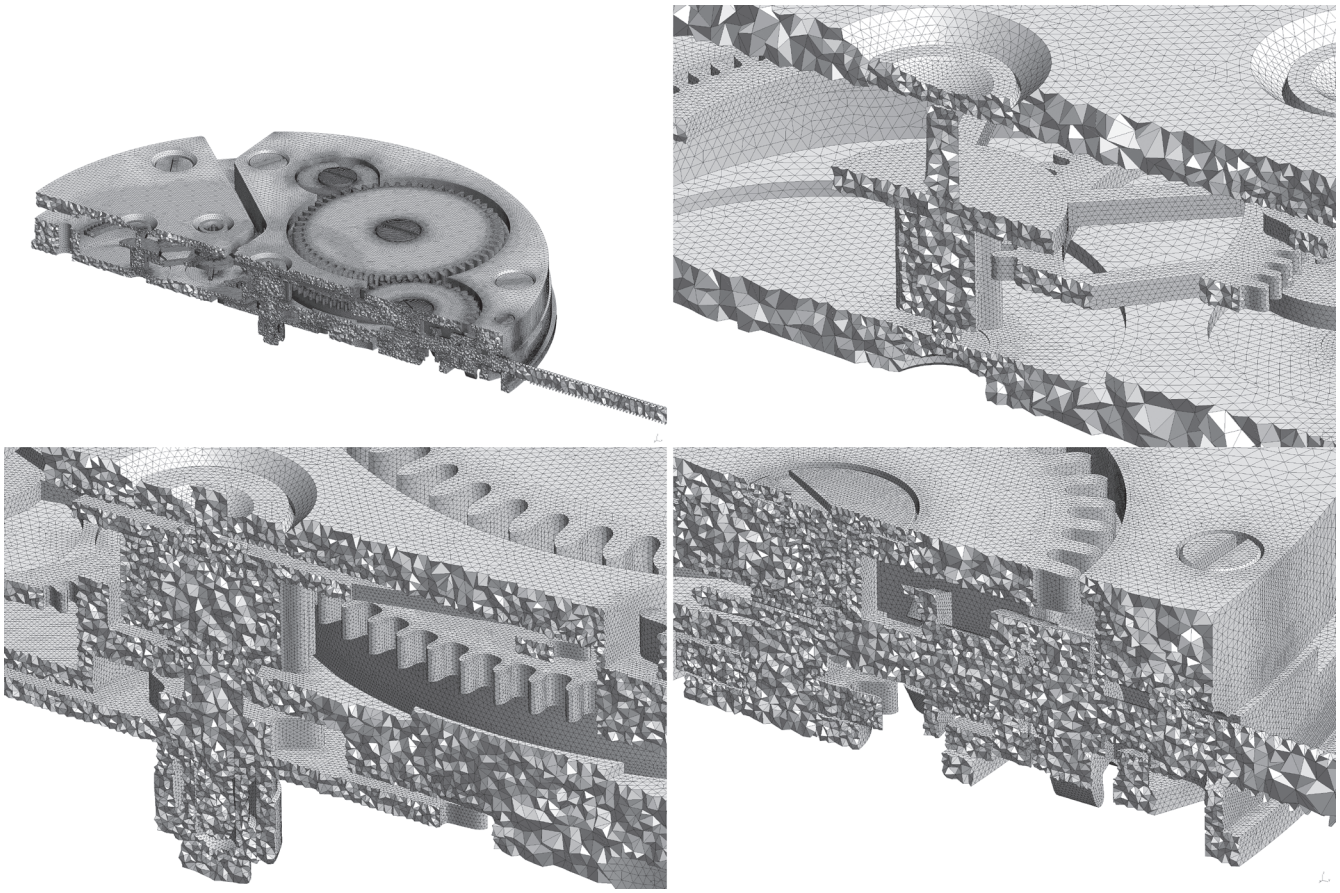


FIGURE 14 Zooms on selected parts of the volume mesh of the watch movement

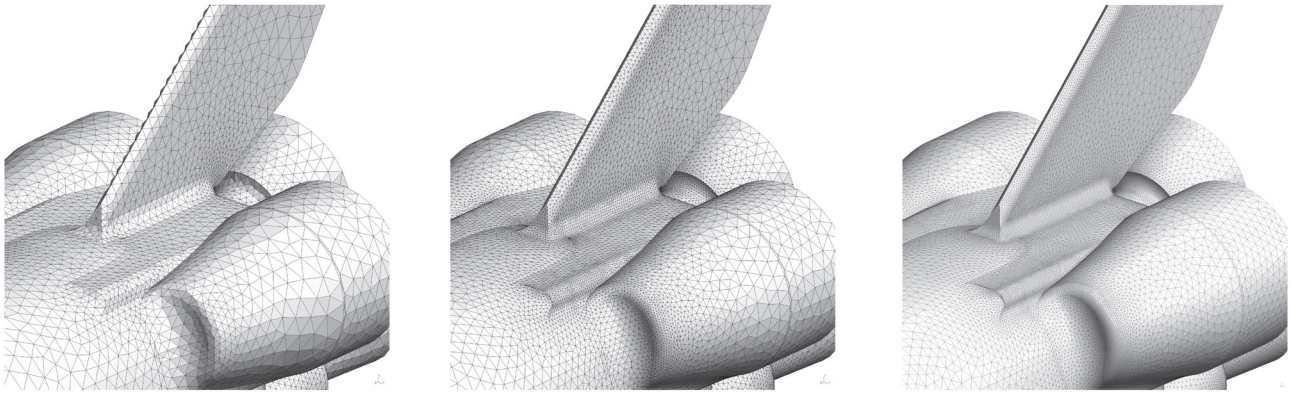


FIGURE 15 Influence of the node density n_d on the final mesh for a gradation of $\alpha = 1.1$

In the same way that the efficiency index is a global indicator for a given mesh, we define an average discrete gradation as the average of $\alpha_{d,i}$ over all edges of the mesh. The average discrete gradation should be close to the user-defined gradation α. In our results over the same sample of meshes, the discrete gradation is close to the required gradation, although it lies slightly above for a variety of volume meshes as a consequence of our choice to limit $\|\nabla h\|_\infty$ instead of $\|\nabla h\|_2$ (Figure 18).

Quality measure. In addition to the measures of adequacy, elementwise mesh quality is also evaluated using the ratio γ of the radius r of the sphere inscribed in a tetrahedron to the radius R of its circumscribing sphere:

$$\gamma = \frac{3r}{R}. \tag{26}$$

This ratio varies from 1 for a regular tetrahedron, and approaches 0 for sliver tetrahedra. The quality distribution for 50 meshes is shown on Figure 17: these meshes exhibit high-quality elements with $\gamma = 0.8$ on average. The current volume meshing algorithm[24] of Gmsh performs mesh optimization if the quality of an element is below 0.4, hence the very reduced number of elements below this threshold.

3.3 | Execution time

The execution times for the engine block and the watch movement are given in Table 1. The mesh size field time is split between the main steps of the algorithm: (*i*) insert the bounding boxes of the triangles of the surface mesh in the RTree, (*ii*) compute the approximated curvature with a least square method, (*iii*) compute the medial axis of the geometry, (*iv*) initialize the root octant and refine the octree, and (*v*) limit the size gradient to $\alpha - 1$. Computing the medial axis requires the Delaunay tetrahedrization of the points from the surface mesh: this step is detailed and included in the time associated to the medial axis computation. The step "Others" includes all the secondary steps, such as splitting the mesh into faces before computing curvature.

To limit the size gradient, we iterate over the octants until condition (12) is satisfied in all three directions. This step is dependent on the depth of the octree, hence on the minimum size $h_{\min}$, the node density n_d and the curvature of the surface mesh given as input. A highly curved region of the model due to a poor resolution of the surface mesh will result

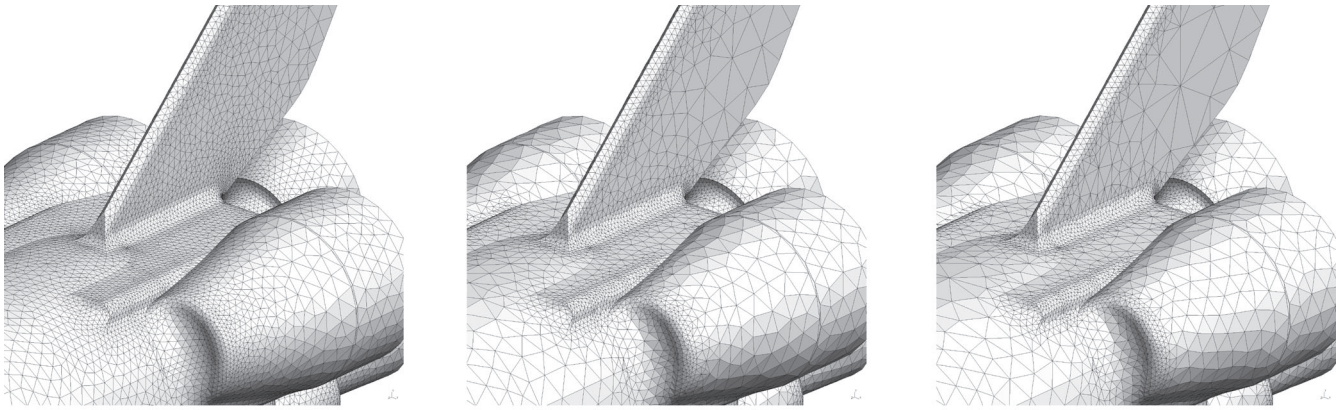


FIGURE 16 Influence of the gradation α on the final mesh for a node density of $n_d = 20$

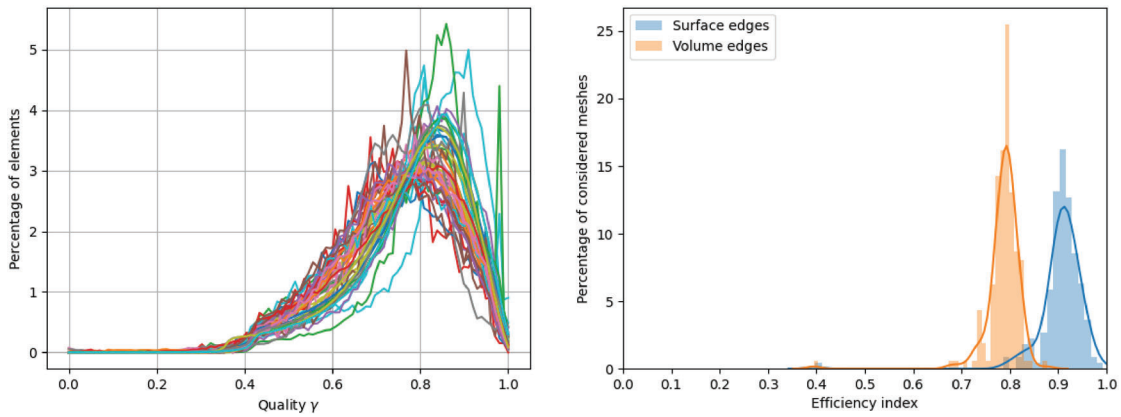


FIGURE 17 Left: Elements quality for 50 meshes. Right: Efficiency index for a sample of 190 meshes

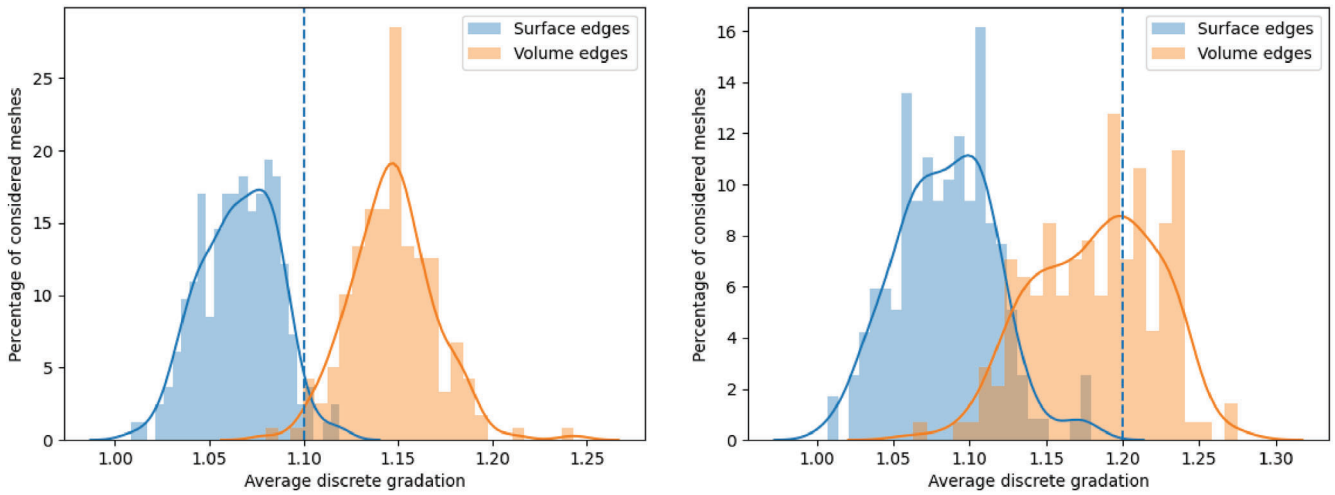


FIGURE 18 Left: Discrete gradation for a sample of 190 meshes for a prescribed $\alpha = 1.1$. Right: $\alpha = 1.2$

TABLE 1 Execution times

		Block		Watch movement	
		Time (s)	**%**	**Time (s)**	**%**
Create mesh size field	Insert surface mesh in RTree	0.47	3.0%	2.30	1.2%
	Compute curvature	0.36	2.3%	1.13	0.6%
	Compute medial axis	7.19	45.7%	64.06	34.1%
	incl. Delaunay tetrahedrization of surface mesh	0.61	3.9%	2.33	1.2%
	Create and refine octree	0.96	6.1%	20.24	10.8%
	Limit size gradient	5.85	37.1%	93.89	50.0%
	Others	0.69	4.3%	6.07	3.2%
	Total	15.75	100.0%	187.69	100.0%
Mesh model	Mesh 1D entities	2.48	2.5%	38.51	4.5%
	Mesh 2D entities	25.68	25.6%	355.58	41.0%
	Mesh 3D entities	72.12	71.9%	469.94	54.4
	Total	100.3	100.0%	864.03	100.0%
	including Size queries	65.13	64.9%	483.53	56.0%
	# of curves in the CAD model	1584		12,991	
	# of faces in the CAD model	533		4760	
	# of nodes in input surface mesh	145,024		654,348	
	# of elements in input surface mesh	290,116		1,308,870	
	# of octants	670,566		9,052,065	
	# of triangles in final surface mesh	573,808		3,629,000	
	# of tetrahedron in final volume mesh	2,716,170		12,273,300	

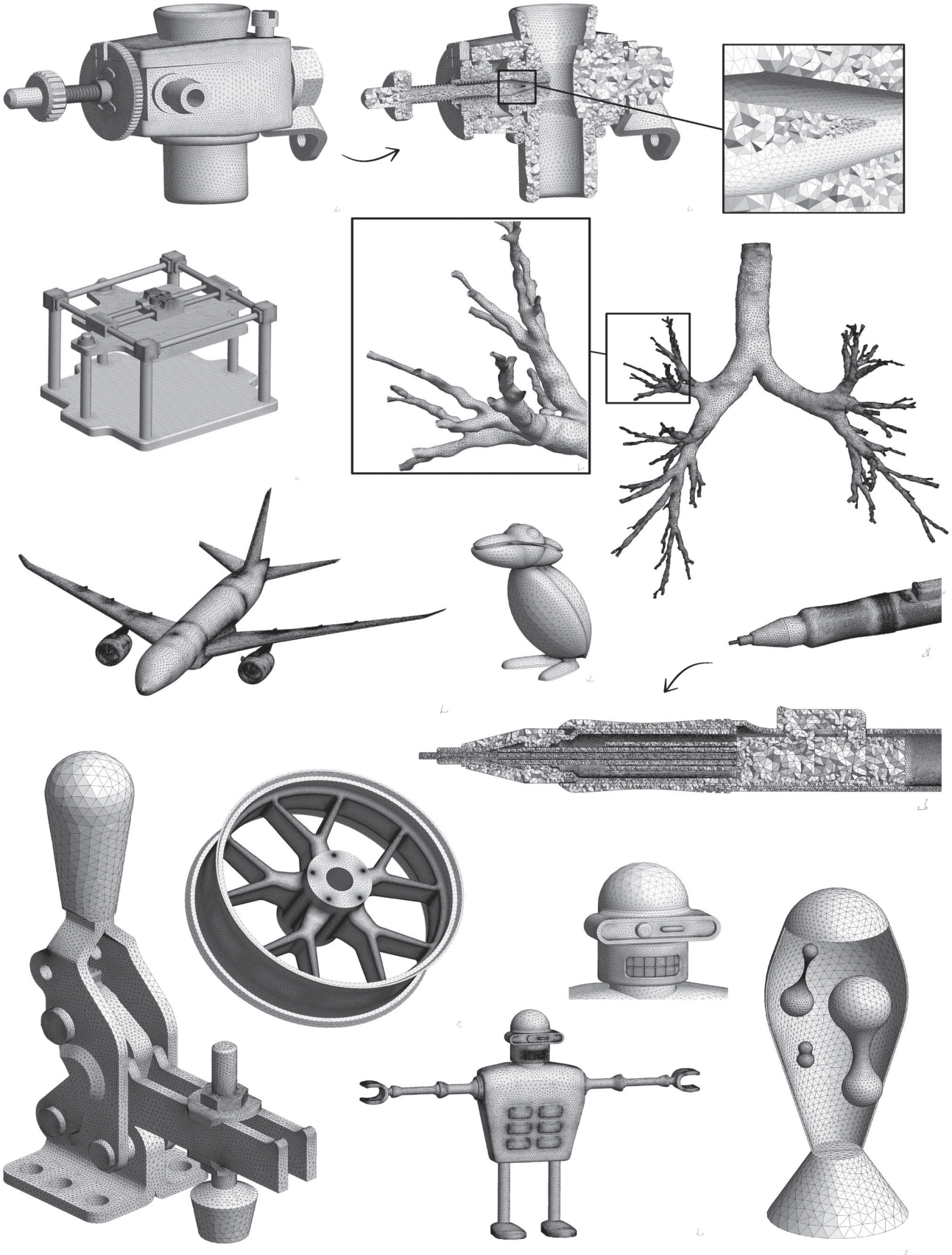


FIGURE 19 Application of the size field on various geometries from GrabCAD, the ABC Dataset and the Gmsh benchmarks

in high density in the octree, increasing the size limitation time. This can generally be circumvented by first generating a slightly refined surface mesh, which can be computed quickly and will rule out these extreme curvature magnitudes. Computation time for the medial axis, on the other hand, is linear with the number of nodes of the input mesh: this is indeed observed in Table 1.

4 | CONCLUSION

A methodology to automatically generate an accurate size field, storing the sizing information in an octree, was presented. This tool eliminates the tedious operation that is assigning by hand the mesh size on all geometric entities of a CAD model, thus saving considerable time. This algorithm was a missing piece of the standard meshing pipeline of Gmsh, and was designed to become the default tool for size specification going onward. Five user parameters are required to generate a mesh suitable for numerical simulations, two of which being the minimum and the maximum, or *bulk*, size, whose value is assigned based solely on the characteristic dimension of the CAD model. This leaves the user with only three parameters in order to tune the mesh density. Special care was given to the small features of the geometry: through an approximate medial axis computation, we ensure multiple layers of elements are always generated in narrow regions. This is particularly useful in simulations where Dirichlet boundary conditions are applied. To illustrate our size field computation, we applied our algorithm on a large variety of CAD models: for all the test cases, high-quality surface and volume meshes were obtained in a robust and automatic fashion, and are adapted to the features of the geometric model. The adequacy between the size fields and the meshing algorithm was demonstrated: the generated meshes are unit or quasi-unit according to the prescribed isotropic metric field, and the desired mesh gradation is met. In thin areas of the CAD model, the required number of layers of elements is present in the final mesh. More importantly, generated meshes are high-quality meshes ready to be used for numerical simulations. Emphasis was placed on selecting the same set of parameters for all the geometries: parameters that *make sense* to the user do indeed yield a suitable mesh for a given geometry, making this tool intuitive and user-friendly. Our choice was to use the P4EST library to implement the background tree structure: although it adds an external dependence, the octree routines are efficient and a parallel implementation is available for further improvements. Future work will focus on extending to anisotropic size prescription, as well as coupling this geometry-based size field with numerical solutions postprocessing results, to include a posteriori error estimation in the size field design. In this sense, this tool will assist users performing mesh adaptation, storing the error-based metric field while providing size prescription based on the CAD throughout the adaptation process (Figure 19).

ACKNOWLEDGMENTS

The authors would like to thank the Belgian Fund for Scientific Research (FRIA/FC 29571) for their support. Financial support from the Simulation-based Engineering Science (Génie Par la Simulation) program funded through the CREATE program from the Natural Sciences and Engineering Research Council of Canada is also gratefully acknowledged. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme (grant agreement ERC-2015-AdG-694020).

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Arthur Bawin https://orcid.org/0000-0002-8118-6809

REFERENCES

1. Geuzaine C, Remacle JF. Gmsh: a 3-D finite element mesh generator with built-in pre-and post-processing facilities. *Int J Numer Methods Eng*. 2009;79(11):1309-1331. https://doi.org/10.1002/nme.2579.
2. Cunha A, Canann S, Saigal S. Automatic boundary sizing for 2D and 3D meshes. *ASME Appl Mech Div-Publ-AMD*. 1997;220:65-72.
3. Owen SJ, Saigal S. Neighborhood-based element sizing control for finite element surface meshing. *Proceedings of the 6th International Meshing Roundtable*, Park City, Utah: Sandia; 1997:143-154.
4. Chen J, Xiao Z, Zheng Y, Zheng J, Li C, Liang K. Automatic sizing functions for unstructured surface mesh generation. *Int J Numer Methods Eng*. 2017;109(4):577-608. https://doi.org/10.1002/nme.5298.
5. Chen J, Liu Z, Zheng Y, et al. Automatic sizing functions for 3D unstructured mesh generation. *Proc Eng*. 2017;203:245-257. https://doi.org/10.1016/j.proeng.2017.09.804.

6. Pirzadeh S. Structured background grids for generation of unstructured grids by advancing-front method. *AIAA J*. 1993;31(2):257-265.
7. Zhu J, Blacker TD, Smith R. Background overlay grid size functions. In: Chrisochoides N, ed. *Proceedings of the 11th International Meshing Roundtable, IMR 2002, September 15-18*. Vol 2002. Ithaca, New York, Sandia; 2002:65-73.
8. Tchon KF, Khachan M, Guibault F, Camarero R. Three-dimensional anisotropic geometric metrics based on local domain curvature and thickness. *Comput Aided Des*. 2005;37(2):173-187. https://doi.org/10.1016/j.cad.2004.05.007.
9. Quadros WR, Vyas V, Brewer M, Owen SJ, Shimada K. A computational framework for automating generation of sizing function in assembly meshing via disconnected skeletons. *Eng Comput*. 2010;26(3):231-247. https://doi.org/10.1007/s00366-009-0164-z.
10. Persson PO. Mesh size functions for implicit geometries and PDE-based gradient limiting. *Eng Comput*. 2006;22(2):95-109. https://doi.org/10.1007/s00366-006-0014-1.
11. Chen J, Cao B, Zheng Y, Xie L, Li C, Xiao Z. Automatic surface repairing, defeaturing and meshing algorithms based on an extended B-rep. *Adv Eng Softw*. 2015;86:55-69. https://doi.org/10.1016/j.advengsoft.2015.04.004.
12. Shan J, Li Y, Guo Y, Guan Z. A robust backward search method based on walk-through for point location on a 3D surface mesh. *Int J Numer Methods Eng*. 2008;73(8):1061-1076. https://doi.org/10.1002/nme.2098.
13. Zhang Y, Wang W, Liang X, et al. High-fidelity tetrahedral mesh generation from medical imaging data for fluid-structure interaction analysis of cerebral aneurysms. *Comput Model Eng Sci*. 2009;42(2):131.
14. Liang X, Zhang Y. An octree-based dual contouring method for triangular and tetrahedral mesh generation with guaranteed angle range. *Eng Comput*. 2014;30(2):211-222.
15. Burstedde C, Wilcox LC, Ghattas O. p4est: scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM J Sci Comput*. 2011;33(3):1103-1133. https://doi.org/10.1137/100791634.
16. Zhang Y, Bajaj C, Sohn BS. 3D finite element meshing from imaging data. *Comput Methods Appl Mech Eng*. 2005;194(48-49):5083-5106.
17. Turner M, Moxey D, Peiró J. Automatic mesh sizing specification of complex three dimensional domains using an octree structure. In: Ledoux F, Lewis K, eds. *Research Note, 24th International Meshing Roundtable, October 11-14*. Austin, Texas; Elsevier: 2015.
18. Deister F, Tremel U, Hassan O, Weatherill NP. Fully automatic and fast mesh size specification for unstructured mesh generation. *Eng Comput*. 2004;20(3):237-248. https://doi.org/10.1007/s00366-004-0291-5.
19. Rusinkiewicz S. Estimating curvatures and their derivatives on triangle meshes. Paper presented at: Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 3DPVT 2004, >Thessaloniki, Greece; vol. 2004, 2004:486-493; IEEE.
20. Dey TK, Zhao W. Approximating the medial axis from the voronoi diagram with a convergence guarantee. *Algorithmica*. 2004;38(1):179-200. https://doi.org/10.1007/s00453-003-1049-y.
21. Beckmann N, Kriegel HP, Schneider R, Seeger B. The R*-tree: an efficient and robust access method for points and rectangles. Paper presented at: Proceedings of the 19 of ACM SIGMOD Record; 1990:322-331; ACM, New York, NY.
22. Koch S, Matveev A, Jiang Z, et al. ABC: a big CAD model dataset for geometric deep learning. Paper presented at: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Computer Vision Foundation / IEEE, Long Beach, CA; 2019:9601-9611.
23. Frey PJ, George PL. *Maillages: Applications Aux Éléments Finis*. Paris, France: Hermès Science Publications; 1999.
24. Marot C, Verhetsel K, Remacle JF. Reviving the search for optimal tetrahedralizations. In: Peirò J, Viertel R, eds. *Proceedings of the 28th International Meshing Roundtable, IMR 2019, October 14-17*. Vol 2019. Buffalo, New York; 2019:326-336.

How to cite this article: Bawin A, Henrotte F, Remacle J-F. Automatic feature-preserving size field for three-dimensional mesh generation. *Int J Numer Methods Eng*. 2021;122:4825–4847. https://doi.org/10.1002/nme.6747