

25th International Meshing Roundtable (IMR25)

Discrete CAD model for visualization and meshing

P. Laug^{a,*}, H. Borouchaki^{a,b}

^a*Gamma3 project-team, Inria Saclay - Île-de-France, France*

^b*Gamma3 project-team, University of Technology of Troyes, France*

Abstract

During the design of an object using a CAD (computer aided design) platform, the user can visualize the ongoing model at every moment. Visualization is based on a discrete representation of the model that coexists with the exact analytical representation of the object. Most CAD systems have this discrete representation available, and each of them applies its own construction methodology. This paper presents a new method to build a discrete (“triangulated” with quadrilaterals and triangles) model for CAD surfaces. It presents two major particularities: most elements are aligned with iso-parametric curves and the accuracy of the surface approximation is controlled. In addition, we present a new technique of surface mesh generation that is based on this discrete model. Several examples are presented to confirm the efficacy of this approach.

© 2016 The Authors. Published by Elsevier Ltd.

Peer-review under responsibility of organizing committee of the 25th International Meshing Roundtable (IMR25).

Keywords: CAD surface; geometric modeling; surface meshing; discrete model; quadtree support.

1. Introduction

In most CAD (computer aided design) systems, any object is defined from its boundary that is constituted by a collection of surface patches. Each patch is defined by a continuous parametric function (typically NURBS) over a bidimensional domain that is called the parametric domain.

A first challenging problem is to visualize such a surface efficiently, in particular on a video display. Most 3D graphics interfaces require a discrete representation of the surface as a set of simple polygons. In this context, the main objective is processing speed, whatever the shape quality of the elements, while preserving the geometric accuracy of the model.

A second problem is to generate meshes for numerical simulations of physical phenomena, applying methods like FEM (finite element method). In this case, the size and shape of the elements must comply with strict specifications tailored to the geometry and the physics of the simulated phenomenon. During the mesh generation, many queries to the CAD system are performed (to evaluate locations of points and also derivatives of parametric functions), which can be time-consuming and difficult to parallelize (due to cache defaults that are involved in most CAD systems).

* Corresponding author. Tel.: +33 1 77 57 80 34.

E-mail address: patrick.laug@inria.fr

In addition, the parametric function representing the surface may vary in a highly irregular manner, and also have degenerate derivatives, resulting in low quality elements.

In this paper, a new method for generating a discrete model (a surface “triangulation” containing quadrilaterals and triangles) for visualization is presented. The model uses a quadtree that follows a network of iso-parametric curves, which generally constitute an adequate representation of the surface (in particular for polynomial and rational functions). The triangulated model is conformal on each patch. For these reasons, it can also be used as a geometric support for mesh generation. Using the geometric support, the mesher is completely disconnected from the CAD system and is therefore faster and parallelizable. In this case, the surface is redefined by piecewise simple polynomial functions, thus avoiding any degeneracy of derivatives.

In the following, Sec. 2 presents the construction of the discrete model, while Sec. 3 presents a method of using this support to generate meshes. Finally, our approach is illustrated by some examples.

2. Discrete model of a parametric surface

A parametric surface Σ is defined by a regular map σ (having a continuity of class C^1 at least) from a domain Ω of \mathbb{R}^2 to \mathbb{R}^3 :

$$\sigma : \Omega \subset \mathbb{R}^2 \rightarrow \Sigma \subset \mathbb{R}^3, \quad (u, v) \mapsto \sigma(u, v). \quad (1)$$

When the pair of parameters (u, v) sweeps the parametric domain Ω , the point $\sigma(u, v)$ moves on the surface patch Σ . It is not uncommon that a CAD parametric domain is a simple unit square $[0, 1]^2$ but actually it can be any shape, possibly having one or more connected components, with possibly inner holes.

The discrete geometric modeling here consists in building a “triangulation” (actually made up of quadrilaterals and triangles) that accurately reflects the geometry of Σ . As in the case of curves, and in general for a variety that is a subset of a global space (\mathbb{R}^3 here), the quality of the support is controlled via the two properties of *proximity* and *regularity*. Proximity controls the distance between the elements of the geometric support and the corresponding surface. As for regularity, it controls the angular gap between the elements (or tangent planes to the elements) and tangent planes to the surface.

Iso- u curves (obtained by giving a constant value to u) and iso- v curves (constant v) generally give an accurate description of the geometry of a parametric surface. This mode of representation is particularly suited to polynomial and rational surfaces that are commonly used in CAD systems. To build the geometric support of a patch, we propose a method using a network of iso-parametric curves as privileged directions for the edges. The method consists in building this support indirectly via the parametric domain, as detailed below step by step.

2.1. Discrete model of curves in 2 and 3 dimensions

The first step consists in building a geometric support that is common to the boundary of Ω and its image that constitutes the boundary of Σ . In general, the boundary of Ω is defined by one or more smooth curves (for instance, in the simple case where Ω is a square, its boundary can be defined by four straight edges). Let Γ_2 be such a smooth curve, and let $\Gamma_3 = \sigma(\Gamma_2)$ be its image on the border of Σ . The bidimensional curve Γ_2 is defined by a regular function ω (having a continuity of class C^1 at least) from an interval $[a, b]$ of \mathbb{R} to \mathbb{R}^2 :

$$\omega : [a, b] \subset \mathbb{R} \rightarrow \Gamma_2 \subset \mathbb{R}^2, \quad t \mapsto \omega(t), \quad (2)$$

and the tridimensional curve Γ_3 is defined by the composition $(\sigma \circ \omega)$ of functions $\omega : \mathbb{R} \rightarrow \mathbb{R}^2$ and $\sigma : \mathbb{R}^2 \rightarrow \mathbb{R}^3$.

In this case, the support is a curve discretization controlled by both properties of *proximity* (here controlling the distance between the edges of the discretization and the curve) and *regularity* (controlling the angular gap between the edges and the tangents to the curve). Creating the support amounts to first defining the geometric support of the bidimensional curve Γ_2 , and then to enrich this support (by recursive subdivisions) so that it is also a geometric support of the tridimensional curve Γ_3 .

First, a coarse discretization of Γ_2 is considered, having only one edge if curve Γ_2 is open and two edges if it is closed. Then the discretization is recursively refined while the C^0 -distance (proximity) or C^1 -distance (regularity) between the curve and the edges of the discretization exceeds a given threshold. The C^0 -distance between an edge

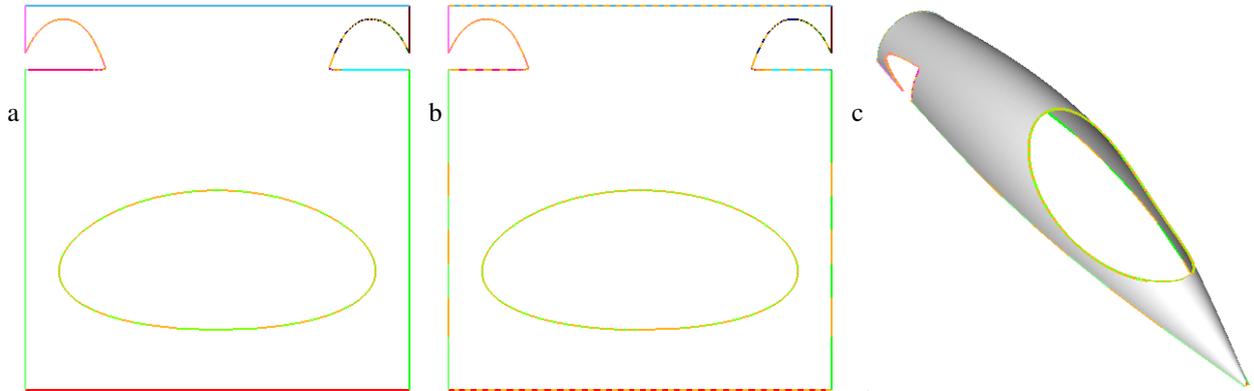


Fig. 1. (a) Initial geometric support of Γ_2 only. (b) Refined support of both Γ_2 and $\Gamma_3 = \sigma(\Gamma_2)$. (c) Surface patch with boundary curves Γ_3 .

of the discretization, which is a straight segment $[S]$ having its two extremities on the curve, and the corresponding curved segment (S) is defined by:

$$d_0((S), [S]) = \max_{M \in (S)} d_0(M, [S]). \quad (3)$$

The C^1 -distance is similarly defined with a distance $d_1(M, [S])$ representing the angular gap between the tangent to the curve at M and the straight segment $[S]$.

The C^0 -distance can be analytically evaluated from the defining mapping function and its first derivatives. In practice, a finite set of sampling points of (S) is considered by generating pseudo-random values $\{t_i\}_{i=1,\dots,n}$ in the interval $[p, q]$ defining (S) . Let t_m be the value for which the maximum distance is reached. If this distance exceeds a given threshold, the curved segment $(S) = \omega([p, q])$ is split into the subsegments $\omega([p, t_m])$ and $\omega([t_m, q])$, which are recursively analyzed. The threshold is considered as relative to the length of the straight segment $[S]$ instead of being absolute.

Then, starting with the obtained discretization which is a geometric support for Γ_2 , the same refinement process is applied in three dimensions to construct the final discrete model, which is a geometric support for both Γ_2 and Γ_3 . Actually, the whole process is fully symmetrical for Γ_2 and Γ_3 : if for instance Γ_2 is a straight segment and Γ_3 a curved segment, then the support will be oversampled for Γ_2 , and *vice versa*. At the end of this step, a set of points and straight segments in \mathbb{R}^2 is available, representing the vertices and the edges of the geometric support of the boundary curves of Ω , while the set of their images in \mathbb{R}^3 forms the vertices and the edges of the geometric support of the boundary curves of Σ . To simplify the next steps, bidimensional coordinates are normalized by a homothetic transformation sending the axis-aligned bounding box of Ω to the unit square $[0, 1]^2$.

For example, Fig. 1a shows the initial geometric support of bidimensional curves Γ_2 . This support is refined as can be seen in Fig. 1b, because the images $\Gamma_3 = \sigma(\Gamma_2)$ are curved boundaries of a surface patch illustrated by Fig. 1c (in particular, the nose of the aircraft is represented by a small half-circle in three dimensions and consequently discretized with a minimum size). The data originate from patch 114 of a STEP file made available for the meshing contest of the 22nd International Meshing Roundtable, Orlando, FL, 2013.

2.2. Build a quadtree from the vertices of the curve geometric support

In the second step, the unit square defined above is partitioned by a quadtree [5,12], where each node corresponds to a square quadrant. This quadtree is such that the interior of each leaf (node with no children) contains at most one vertex of the curve geometric support. The boundary of each leaf, however, may contain any number of vertices of this support to avoid useless refinements (unlike traditional point-region quadtrees). In order to achieve this, the quadtree is initialized with only one node corresponding to the unit square $[0, 1]^2$, and vertices are successively inserted. To insert a vertex, a fast recursive algorithm finds in which leaf it lies. If the interior of this leaf does not contain any vertex then the latter can be inserted, otherwise the leaf is recursively subdivided into four equal-sized squares until a vacant leaf is found. As an example, Fig. 2a shows the quadtree resulting from the previous curve geometric support.

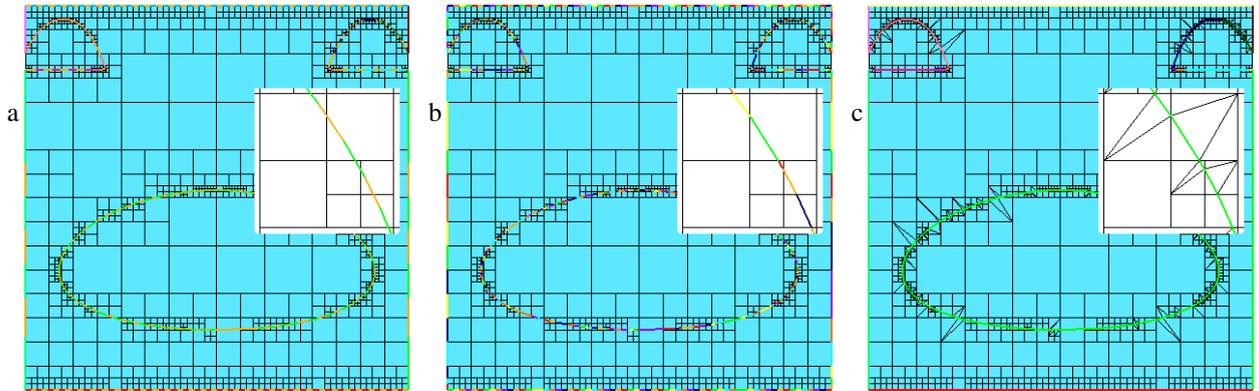


Fig. 2. (a) Quadtree such that the interior of each leaf contains at most one vertex of the curve geometric support. (b) Quadtree with inserted and merged traces. (c) Quadtree with a constrained “triangulation” of each leaf. Close-up pictures are included to clarify these three steps.

2.3. Insert and merge edges into the quadtree

The third step concerns the insertion of the edges of the curve geometric support into the quadtree. An edge can cross any number of leaves, and the points of intersection between this edge and the borders of these leaves define a partition of the edge, giving sub-edges that are referred to as *traces*. In many cases, a quadrant now contains two adjacent traces of a same curve, hence complying with the properties of proximity and regularity. Then, these traces are merged into one straight segment. Examples of resulting traces can be seen in Fig. 2b.

2.4. Construct a constrained “triangulation” of each leaf

Each leaf now corresponds to a square containing zero, one or more traces. The aim is to construct a “*triangulation*” of this square in which the traces are forced. Note that this “*triangulation*” may contain not only triangles but also quadrilaterals, but we will use this term in the following without quotation marks. To make such a triangulation, if the interior of the leaf still contains a vertex (which was necessarily an extremity of a boundary curve of Ω), this vertex is simply “starred”, that is, connected to the four corners of the square and possibly to the vertices within some sides of the square. Otherwise, frequent cases are first considered to simplify and to improve the resulting triangulation. Indeed, if there is no vertex on the border of the square, the triangulation is constructed depending on the number n of traces crossing the leaf:

- If $n = 0$, it reduces to one quadrilateral covering the leaf.
- If $n = 1$, the extremities of the trace are either on opposite sides of the square, or on sides sharing a corner C of the square. In the first case, two quadrilaterals are constructed, while in the second case four triangles are made after connecting each extremity of the trace with the opposite corner of C .
- If $n = 2$, frequently one trace connects two opposite sides of the square, and the second trace connects the same sides, so only three quadrilaterals are generated. If the extremities of the traces are distributed on the four sides of the square, one quadrilateral and four triangles are made.

In any other case, a general algorithm is applied to construct a triangulation (made up of triangles only). The input is composed of the four sides of the square, vertices lying on these sides, and traces crossing the square. An initial triangulation is created from the four sides and the vertices, which define a partition of the square boundary. Then, the traces are forced by recursive edge swapping (this simple algorithm converges since all triangulations in 2D are equivalent by means of swaps). In this swapping algorithm, the shape quality of the triangles is ignored since the quality of the geometric support is optimized in a further step (cf. section 2.9).

A resulting constrained triangulation of a quadtree is shown in Fig. 2c.

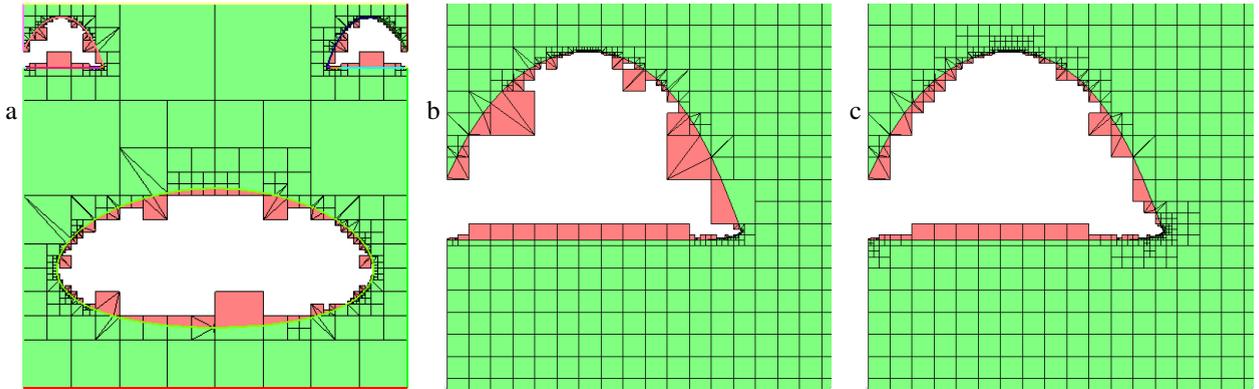


Fig. 3. (a) Elements of the triangulation inside the parametric domain. (b) Surface geometric support (close-up view). (c) Balanced quadtree (close-up view).

2.5. Identify all the elements of the triangulation inside the parametric domain

At this stage, the quadtree represents a non-conformal triangulation, containing among its edges an approximation of the edges of the curve geometric support defined in section 2.1, which is an approximation of the boundary of the parametric domain Ω . Consequently, it is possible to distinguish between the elements of the triangulation that are inside or outside an approximation of Ω . Let s (seed) be any element of the triangulation touching the contour of the unit square. If all the edges of s lying on the contour are edges of the curve support, then s is in the category “inside”. Otherwise it is in the category “outside”. Growing by adjacency starting from s without ever passing through an edge of the curve support, a whole connected component of the quadtree is identified with the same category. If some elements are remaining, a new seed with the opposite category is obtained, this time passing through an edge of the curve support, then a new connected component is identified, and so on. Since the triangulation is globally non-conformal, but locally conformal in each leaf of the quadtree, adjacency is defined by considering both the neighborhood of leaves of the quadtree and the adjacency between elements of a same leaf.

In Fig. 3a, the elements of the triangulation are represented in green if they are inside the parametric domain, and in red if they are outside. If all the elements of a leaf are outside, the leaf is not represented.

2.6. Make a surface geometric support

We can now analyze leaves or selected parts of leaves with respect to the two properties of proximity and regularity, and iteratively refine the leaves if necessary. Refining a leaf consists in subdividing it into four equal-sized squares and to insert the associated traces, if any, to the four new leaves (cf. section 2.3). For each of these four new leaves, a constrained triangulation is constructed (cf. section 2.4). New elements are simply identified as inside or outside Ω , based on the initial triangulation of section 2.5.

Let $[k]$ be an element of the triangulation of Ω , $[K]$ the corresponding element of the support whose vertices are the images of the vertices of $[k]$, and (K) the image of $[k]$ by σ , which is a part of patch Σ . The proximity criterion involves the distance $d_0((K), [K])$ between (K) and $[K]$ defined by:

$$d_0((K), [K]) = \max_{M \in (K)} d_0(M, [K]). \quad (4)$$

The regularity criterion is similarly defined with a distance d_1 representing an angular gap. Distances d_0 and d_1 are compared to absolute or relative thresholds, for example a percentage of the element size for d_0 and a maximum angular gap for d_1 .

In practice, as for curves, only a reduced sample of points of (K) are considered, namely the image of the middle of each edge of $[k]$ and the image of its barycenter. If $[K]$ is a quadrilateral, distances are approximated by subdividing $[K]$ into four triangles around its barycenter. If a relative threshold is utilized, the distance for the middle of each edge is relative to the corresponding edge length in $[K]$, and the distance for the barycenter is relative to the size of $[K]$. For

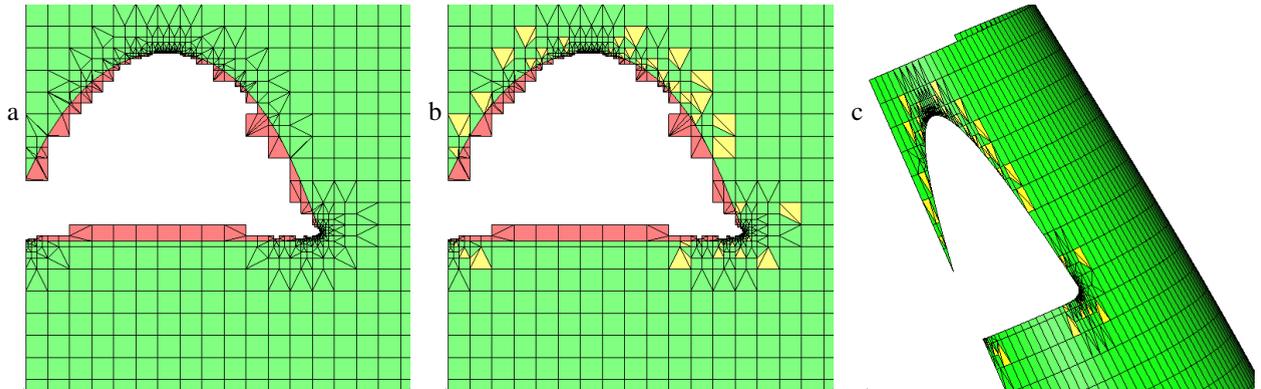


Fig. 4. (a) Conformal triangulation. (b) Smoothed geometric support. (c) Same support in 3D.

each leaf, distance d_0 is computed for each of the elements of its triangulation, and if one of the evaluated distances exceeds the threshold then the leaf is subdivided into four equal-sized squares, and constrained triangulations are constructed in the new leaves as stated before.

Fig. 3b is a close-up picture of the quadtree after the above refinement producing a surface geometric support.

2.7. Balance the quadtree

The quadtree now represents a non-conformal triangulation that accurately reflects the geometry of curves and surfaces. In this step, in order to make the next step of building a conformal triangulation easier, the tree is balanced in order to impose a 2:1 rule between neighboring leaves. As in the previous step, if a leaf violates this rule then it is subdivided into four equal-sized squares and new constrained triangulations are constructed. In fact, a stricter rule applies: if a leaf has 2 neighbors at one side, and also 2 neighbors at the opposite side, then it is also subdivided.

Fig. 3c shows the previous region of the quadtree after balancing.

2.8. Make a conformal triangulation

For each edge of each element $[k]$ of the triangulation of Ω , let us consider the number n of neighboring elements at this edge. After the previous balancing operation we always have $n \leq 2$, and the configuration $n = 2$ defines one *hanging point* on an edge of $[k]$. To make a conformal triangulation, the total number p of hanging points for each element $[k]$ – be it a triangle or a quadrilateral – is examined:

- If $p = 0$, no action is taken.
- If $p = 1$, and if $[k]$ is a triangle (resp. quadrilateral), then $[k]$ is subdivided in 2 (resp. 3) triangles by connecting the hanging point with its opposite vertex (resp. vertices).
- If $p = 2$, the only possible configuration corresponds to 2 hanging points on 2 adjacent edges, and a simple pattern is used for building 3 triangles if $[k]$ is a triangle, or 4 triangles if $[k]$ is a quadrilateral.

Again, the quality of the triangles is not considered here since it is optimized in the next step. Examples with $p = 0, 1$ and 2 can be seen in Fig. 4a.

2.9. Smooth the geometric support

In this final step, the geometric support is made smoother by edge swapping. If e_1 and e_2 are two adjacent triangles in a leaf of the quadtree, e_1 is adjacent to e_2 and possibly two other elements e_3 and e_4 , and e_2 is adjacent to e_1 and possibly two other elements e_5 and e_6 . Let $a(e_i, e_j)$ be the angle between the normal to e_i and the normal to e_j if both

elements e_i and e_j exist, or else 0, for any $i, j = 1, \dots, 6$. The *roughness* $r(e_1, e_2)$ of the adjacent triangles e_1 and e_2 involves angles between normals to triangles e_1, e_2 and their adjacent elements e_3, e_4, e_5 and e_6 :

$$r(e_1, e_2) = a(e_1, e_2) + a(e_1, e_3) + a(e_1, e_4) + a(e_2, e_5) + a(e_2, e_6). \quad (5)$$

By edge swapping, adjacent triangles e_1 and e_2 become e'_1 and e'_2 . If the roughness decreases, meaning that $r(e'_1, e'_2) < r(e_1, e_2)$, then the triangles are actually swapped, and this is repeated for each pair of adjacent triangles of the leaf until no improvement can be made.

In practice, to compute the normal to a quadrilateral, the latter is subdivided into four triangles around its barycenter. The normal is approximated by the normalized sum of the four normalized normals to these triangles.

Fig. 4b is a bidimensional view of a smoothed geometric support where swapped triangles are displayed in yellow. Fig. 4c shows the same support in the tridimensional space.

After the construction described in this section 2, the resulting discrete model constitutes a triangulation that is globally non-conformal, but conformal in each surface patch. This triangulation is close to the geometry, by construction, and represents an appropriate model for visualization. In particular, the accuracy of the geometric model can be controlled (via the different introduced thresholds). The second advantage of the discrete model is that it can also be used as a support and as a base for redefining a parametrization that is appropriate to mesh generation, as explained in the next section.

3. Surface meshing via the parametrization of the geometric support

For meshing a parametric surface, direct or indirect approaches can be used. In a direct approach, the mesh is generated on the surface directly in \mathbb{R}^3 , as in octree-based methods [14], advancing-front methods [9] and paving methods [6]. In an indirect approach, the parametric domain is meshed and then mapped onto the surface.

The indirect approach is expected to be simpler, faster and more robust since the mesh is generated in two dimensions. It was first used for visualization without considering mesh quality [3,4,11,13]. A difficulty with this method is to follow prescribed size specifications on the surface. For instance, to obtain an isotropic mesh of the surface, an anisotropic mesh of the parametric domain must generally be generated, because of metric distortions from the surface to its parametric domain. Various algorithms can be found in the literature [2,15,16]. Methods to generate a *geometric mesh* (that is, complying with the properties of proximity and regularity) in an indirect manner have been proposed in the *isotropic* and *anisotropic* cases [1,7].

Using the indirect approach, let us recall (see [7] for more details) the main necessary ingredients. The usual length $L(\mathbf{e})$ of an edge \mathbf{e} in \mathbb{R}^2 or \mathbb{R}^3 is defined by $L^2(\mathbf{e}) = \mathbf{e}^T \mathbf{e}$, and the length $L_{\mathbf{M}}(\mathbf{e})$ of \mathbf{e} in a constant metric \mathbf{M} is defined by $L_{\mathbf{M}}^2(\mathbf{e}) = \mathbf{e}^T \mathbf{M} \mathbf{e}$. In \mathbb{R}^2 , for a given point P , the locus of points X such that $L_{\mathbf{M}}^2(\overrightarrow{PX}) = 1$ is an ellipse, and in \mathbb{R}^3 an ellipsoid. The size and the shape of the elements can be controlled using such metrics.

To build an isotropic geometric mesh, the prescribed size at any point P of the surface must be $h(P) = \lambda \rho_1(P)$, where λ is a given constant and $\rho_1(P)$ is the minimum radius of curvature at P . This defines an isotropic metric $\mathbf{M}_3(P) = (1/h^2(P)) \mathbf{I}_3$ on the surface, where \mathbf{I}_3 is the 3×3 identity matrix. For an anisotropic geometric mesh, the prescribed sizes must be $h_1(P) = \lambda_1 \rho_1(P)$ and $h_2(P) = \lambda_2 \rho_2(P)$ along the principal directions of curvature, where $\rho_1(P)$ and $\rho_2(P)$ are the minimum and maximum radii of curvature at P , λ_1 is a given constant and $\lambda_2 \leq \lambda_1$ is a factor depending on the constant λ_1 and the ratio $\rho_1(P)/\rho_2(P) \leq 1$. This defines a metric in the tangent plane to the surface at P , which can be expressed as an anisotropic metric $\mathbf{M}_3(P)$ in the usual basis of \mathbb{R}^3 .

Having an isotropic or anisotropic metric \mathbf{M}_3 at any point of the surface, the induced metric $\tilde{\mathbf{M}}_2$ in the parametric space is given by:

$$\tilde{\mathbf{M}}_2 = \begin{pmatrix} \sigma_u^T \\ \sigma_v^T \end{pmatrix} \mathbf{M}_3 \begin{pmatrix} \sigma_u & \sigma_v \end{pmatrix}, \quad (6)$$

where σ_u and σ_v denote the partial derivatives of σ with respect to u and v . This yields in the isotropic case:

$$\tilde{\mathbf{M}}_2 = \begin{pmatrix} \sigma_u^T \\ \sigma_v^T \end{pmatrix} \frac{1}{h^2} \mathbf{I}_3 \begin{pmatrix} \sigma_u & \sigma_v \end{pmatrix} = \frac{1}{h^2} \begin{pmatrix} \sigma_u^T \sigma_u & \sigma_u^T \sigma_v \\ \sigma_v^T \sigma_u & \sigma_v^T \sigma_v \end{pmatrix} = \frac{1}{h^2} \mathbf{M}_\sigma, \quad (7)$$

where \mathbf{M}_σ is a 2×2 symmetric matrix which characterizes the local intrinsic metric of the surface.

This indicates that, in order to build a surface mesh using an indirect approach, function σ as well as its first partial derivatives σ_u and σ_v must be known for each patch. In particular, a constant size mesh of the surface can be generated without knowing other derivatives. For a geometric mesh, however, the curvatures of the surface must also be evaluated, which generally requires the second partial derivatives σ_{uu} , σ_{uv} and σ_{vv} . Similarly, function ω is necessary for a constant size discretization of curves, and its derivatives at order 1 and 2 for a geometric discretization. Using the discrete model instead of the continuous model, functions ω and σ defining curves and surfaces are replaced by simpler polynomial functions $\bar{\omega}$ and $\bar{\sigma}$ for which the derivatives and curvatures can easily be evaluated. Another advantage is that the code implementing $\bar{\omega}$ and $\bar{\sigma}$ can efficiently be called in parallel for different patches, contrary to ω and σ because most CAD systems use memory caches to save information [8].

3.1. Approximate function $\bar{\omega}$ to define a curve

In the support defined in section 2, each boundary curve Γ_2 is represented by separate edges associated to leaves of a quadtree. However, the discretization of Γ_2 should be defined by a sequence of vertices (P_0, P_1, \dots, P_n) , where $P_0 = A$ and $P_n = B$ are the two extremities of Γ_2 , and (P_{i-1}, P_i) , $i = 1, \dots, n$, are the edges of the curve support. This can be efficiently obtained by finding $P_0 = A$ in the quadtree and get the first edge (P_0, P_1) , then finding P_1 and get the second edge (P_1, P_2) , and so on until $P_n = B$ is reached.

We can now approximate function ω defined by Eq. 2 as follows. Let t be a real parameter in the interval $[0, n]$, and let $\lfloor t \rfloor$ denote its image by the floor function. Then $it = \min(\lfloor t \rfloor, n - 1)$ is an integer between 0 and $n - 1$, and function $\bar{\omega}$ can be defined by:

$$\bar{\omega} : [0, n] \subset \mathbb{R} \rightarrow \mathbb{R}^2, \quad t \mapsto \bar{\omega}(t) = P_{it} + (t - it)(P_{it+1} - P_{it}). \quad (8)$$

This provides a piecewise linear function approximating ω . When t varies from 0 to n , $\bar{\omega}(t)$ sweeps a polyline approximating Γ_2 . For a given t and the corresponding integer it , the first derivative of $\bar{\omega}$ is $P_{it+1} - P_{it}$, and its second derivative is zero. However, curvatures can be approximated at each vertex of the discretization of Γ_2 by considering the directions of the two edges sharing the vertex, and then interpolating along the edges these curvatures.

3.2. Approximate function $\bar{\sigma}$ to define a surface

Having a point p in the discretized parametric domain $\bar{\Omega}$, a fast algorithm can find the leaf of the quadtree where p is located. This leaf contains one or more elements of the triangulated support. For the sake of simplicity, any quadrilateral element will be considered like 4 triangles around its barycenter. The triangles of the leaf, which are very few in number, are examined until a triangle contains p . To test if a triangle with vertices a, b, c contains point p , the signed barycentric coordinates α, β, γ of p are computed (here we will have $\alpha, \beta, \gamma \in \mathbb{R}$ and $\alpha + \beta + \gamma = 1$). These are defined by $\alpha = |\vec{pb} \vec{pc}| / \Delta$, $\beta = |\vec{pc} \vec{pa}| / \Delta$, $\gamma = |\vec{pa} \vec{pb}| / \Delta$, where $|\cdot|$ denotes a 2×2 determinant and $\Delta = |\vec{ab} \vec{ac}|$ is the signed area of triangle abc (positive as the vertices are in a direct order). The triangle contains p iff all the barycentric coordinates are positive, $\alpha, \beta, \gamma \geq 0$. Hence α, β, γ become classical barycentric coordinates between 0 and 1, and function $\bar{\sigma}$ can be defined by:

$$\bar{\sigma} : \bar{\Omega} \subset [0, 1]^2 \rightarrow \mathbb{R}^3, \quad p \mapsto \bar{\sigma}(p) = \alpha A + \beta B + \gamma C, \quad (9)$$

where A, B, C are the tridimensional images of the bidimensional points a, b, c by σ .

To compute the partial derivatives of $\bar{\sigma}$ with respect to u and v , let us define the coordinates of the bidimensional points $p = (u, v)^T$, $a = (a_1, a_2)^T$, $b = (b_1, b_2)^T$, $c = (c_1, c_2)^T$. Therefore, we have $\bar{\sigma}(u, v) = \alpha(u, v)A + \beta(u, v)B + \gamma(u, v)C$ with:

$$\alpha(u, v) = \frac{1}{\Delta} \begin{vmatrix} \vec{pb} & \vec{pc} \\ b_1 - u & c_1 - u \\ b_2 - v & c_2 - v \end{vmatrix} = \frac{1}{\Delta} (b_2 - c_2)u + \frac{1}{\Delta} (c_1 - b_1)v + \frac{1}{\Delta} \text{const.} \quad (10)$$

Similar expressions are obtained for $\beta(u, v)$ and $\gamma(u, v)$, hence the partial derivatives of $\bar{\sigma}$ with respect to u and v :

$$\bar{\sigma}_u(u, v) = \frac{\partial \bar{\sigma}}{\partial u}(u, v) = \frac{1}{\Delta} \left((b_2 - c_2)A + (c_2 - a_2)B + (a_2 - b_2)C \right), \quad (11)$$

$$\bar{\sigma}_v(u, v) = \frac{\partial \bar{\sigma}}{\partial v}(u, v) = \frac{1}{\Delta} \left((c_1 - b_1)A + (a_1 - c_1)B + (b_1 - a_1)C \right). \quad (12)$$

Since the first partial derivatives are constant on each triangle, they can be stored once and for all to improve computational efficiency. Second partial derivatives always equal zero. However, curvatures can be approximated at each vertex of the triangulation of Σ by considering the normals to the elements sharing the vertex, and then interpolating in the elements these curvatures.

4. Examples

The proposed methodology has been implemented in ALIEN, which is a software component that can be plugged in CAD systems. In particular, it has been added to the Open Cascade platform [10] to make an application that can read a CAD model from an IGES or STEP file, build a discrete support of this model and generate meshes from this support. Three models are utilized in this section, all provided for meshing contests at the International Meshing Roundtable: the White House (IMR25, Washington, DC, 2016), an exhaust fan (IMR25 also) and an electric guitar (IMR24, Austin, TX, 2015).

4.1. White House

The input model of the White House (IMR25) contains 1580 patches in STEP format. Tests and performance measurements have been carried out on a MacBook Pro laptop with a 2 GHz Intel Core i7 processor including 4 physical cores. Using ALIEN, a *discrete model* has been generated with a relative threshold 0.03 for curves and surfaces (Fig. 5). This discrete model contains 62251 elements (29593 triangles $\approx 47.54\%$ of elements and 32658 quadrilaterals $\approx 52.46\%$ of elements). The computation time (without times for reading and writing files) is 0.192 seconds. A *second discrete model* has been generated with a smaller relative threshold 0.01, which is the default value (Fig. 6). The triangulation is finer in regions where curves or surfaces are curved. A total of 385987 elements (49636 triangles $\approx 12.86\%$ of elements and 336351 quadrilaterals $\approx 87.14\%$ of elements) has been created in 0.969 seconds.

This finer discrete representation has been used as a geometric support for generating a *uniform mesh* with a constant size $h = 380$ (see Fig. 7a). The mesh contains 222706 triangles with an average shape quality 0.972570. The shape quality of a triangle is defined as its area divided by the sum of the squares of the lengths of its sides, normalized by a factor $4\sqrt{3}$. The elapsed times for meshing are respectively 2.440, 1.277, 0.884 and 0.675 seconds on 1, 2, 3 and 4 cores, demonstrating an almost linear scaling.

A *geometric mesh* has also been generated from the same discrete representation (see Fig. 7b). The maximum angular gap between the elements and the tangent planes to the surface at their vertices is 8 degrees and the size gradation is limited to 1.25. The resulting mesh contains 137970 triangles and the average shape quality is 0.957108.

4.2. Exhaust fan

In this second example, the STEP model of an exhaust fan (IMR25) contains 697 patches. The discrete model (Fig. 8), constructed by ALIEN with a relative threshold 0.01, contains 893765 elements (82621 triangles $\approx 9.24415\%$ of elements and 811144 quadrilaterals $\approx 90.7558\%$ of elements). As a remark, with this geometric accuracy, two intersections of boundary curves have been detected.

From this discrete support, two meshes have been generated by ALIEN. Fig. 9a shows the uniform mesh with size $h = 0.85$, containing 214264 triangles with an average shape quality 0.950649. Fig. 9b shows the geometric mesh with the same size specifications as above, containing 527989 triangles with an average shape quality 0.960465.

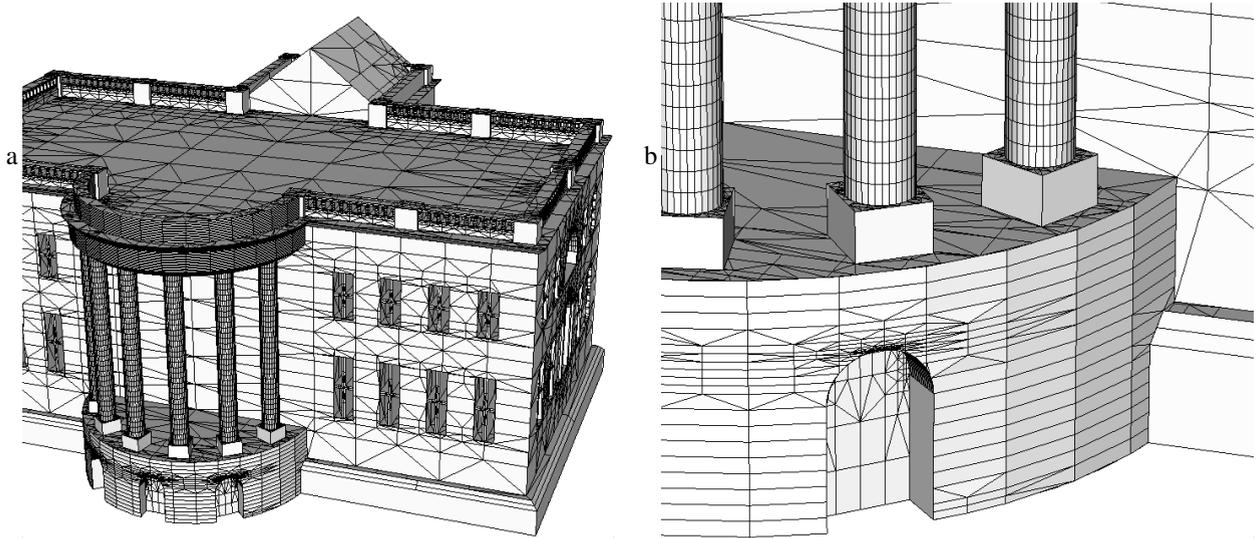


Fig. 5. (a) Discrete model of the White House (relative threshold 0.03). (b) Close-up.

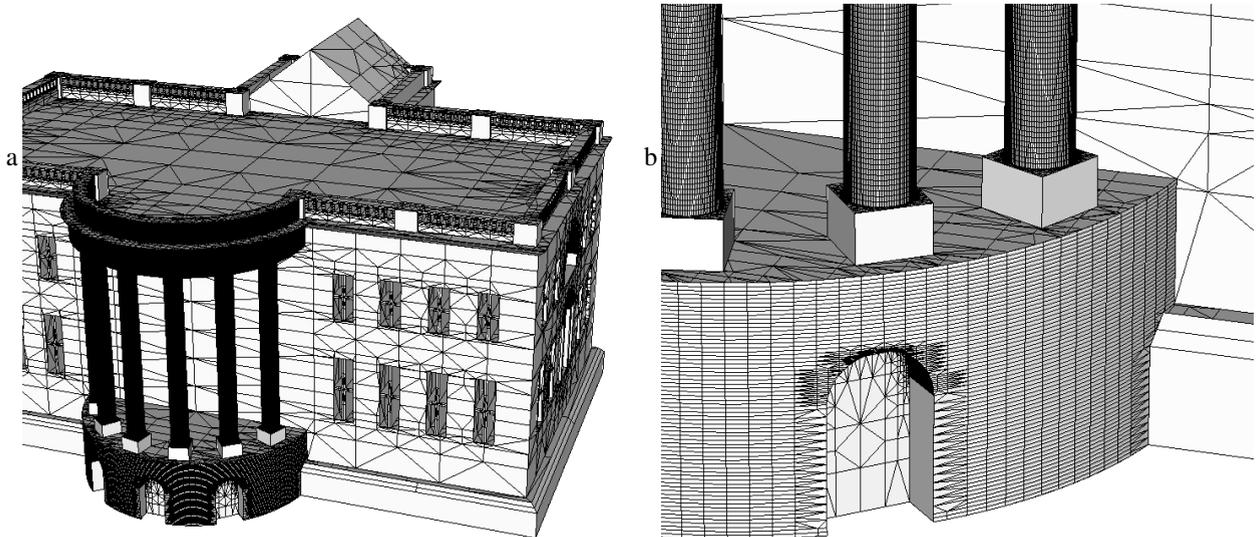


Fig. 6. (a) Discrete model of the White House (relative threshold 0.01). (b) Close-up.

4.3. Electric guitar

The third example is the STEP model of an electric guitar (IMR24) made up of 2018 patches. The ALIEN discrete model (Fig. 10), for a relative threshold of 0.01, contains 9105422 elements (1395773 triangles $\approx 15.329\%$ of elements and 7709649 quadrilaterals $\approx 84.671\%$ of elements).

The first mesh generated by ALIEN (Fig. 11a) is a uniform mesh with size $h = 1.0$, 1054754 triangles with an average shape quality 0.985242. The second mesh (Fig. 11b) is a geometric mesh, 901648 triangles and an average shape quality 0.970245.

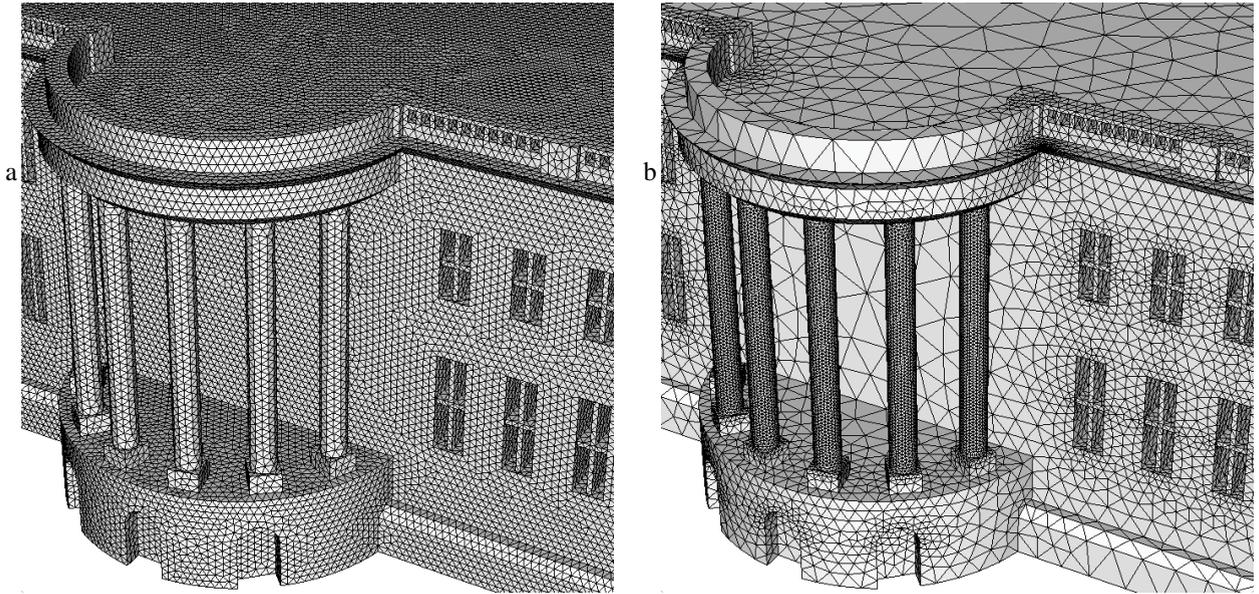


Fig. 7. (a) Uniform mesh of the White House ($h = 380$). (b) Geometric mesh (angle 8° , gradation 1.25).

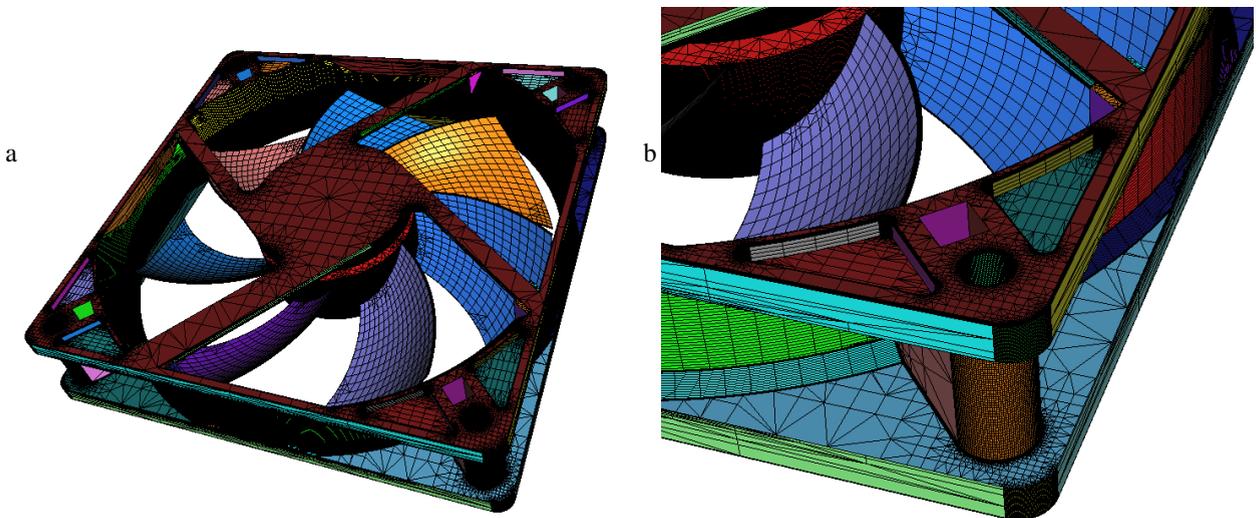


Fig. 8. (a) Discrete model of an exhaust fan (relative threshold 0.01). (b) Close-up.

5. Conclusion

We have presented a new method for building a discrete CAD model. Most elements of this model are aligned with iso-parametric curves, which generally provides an accurate description of the surface geometry. The accuracy can easily be controlled by absolute or relative thresholds. The algorithm for constructing the model is fast and robust. With all these advantages, this approach is well suited for visualization.

Such a discrete representation also concerns mesh generation for numerical simulations. In this case, CAD parametric functions are replaced by simpler polynomial functions that can be efficiently evaluated, as well as their derivatives (that are always non-degenerate and piecewise constant) and discrete curvatures. A further important benefit is the possibility of meshing surface patches in parallel. Most CAD systems use memory caches to store information for each patch, making CAD queries difficult to parallelize. Using a discrete representation, the CAD system is entirely disconnected from the mesher, and consequently the latter can readily be parallelized.

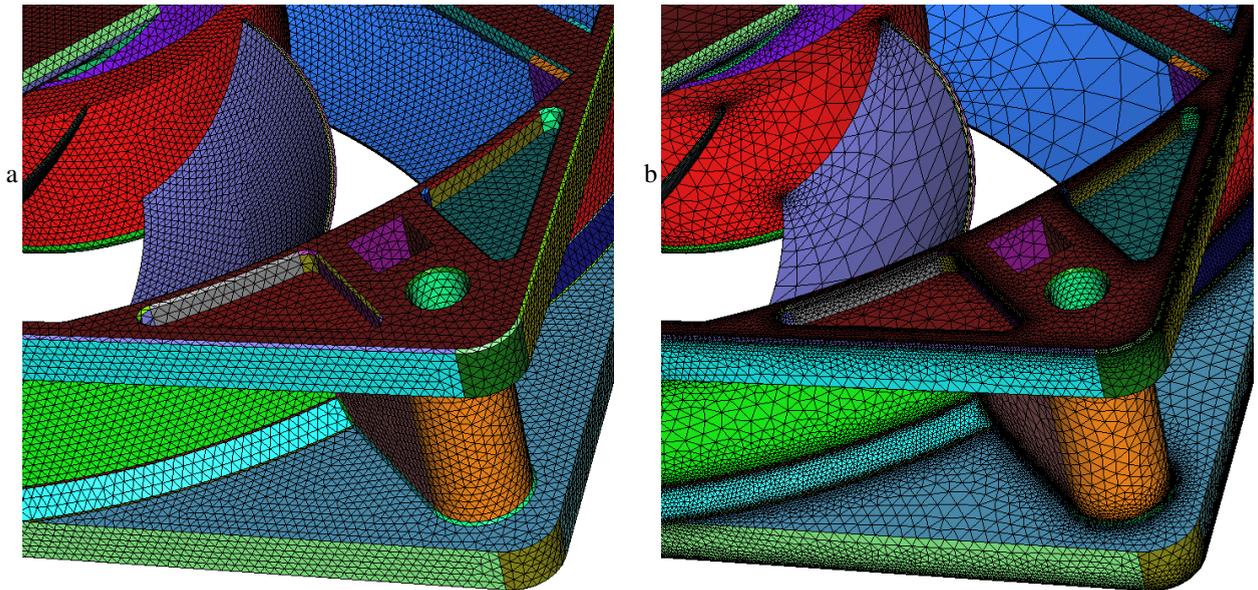


Fig. 9. (a) Uniform mesh of an exhaust fan ($h = 0.85$). (b) Geometric mesh (angle 8° , gradation 1.25).

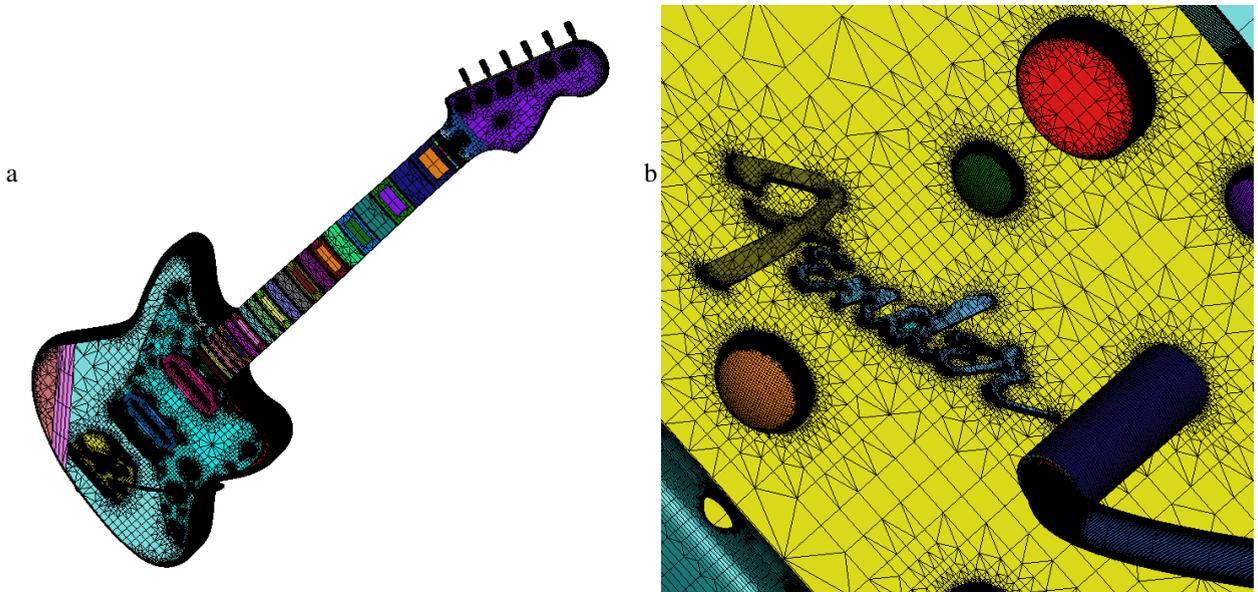


Fig. 10. (a) Discrete model of an electric guitar (relative threshold 0.01). (b) Close-up.

The extension of the method to discrete representations of degree 2 or higher is under investigation. It will significantly reduce the number of elements of the geometric support for a given accuracy, and also produce more accurate calculations of curvatures. A little more computing time will be necessary for the evaluation of higher degree polynomials and their derivatives, but the code will remain easy to parallelize.

References

- [1] H. Borouchaki, P. Laug, P.L. George, Parametric surface meshing using a combined advancing-front – generalized-Delaunay approach, *International Journal for Numerical Methods in Engineering*, vol. 49, no. 1-2, pp. 233-259, Sept. 2000.

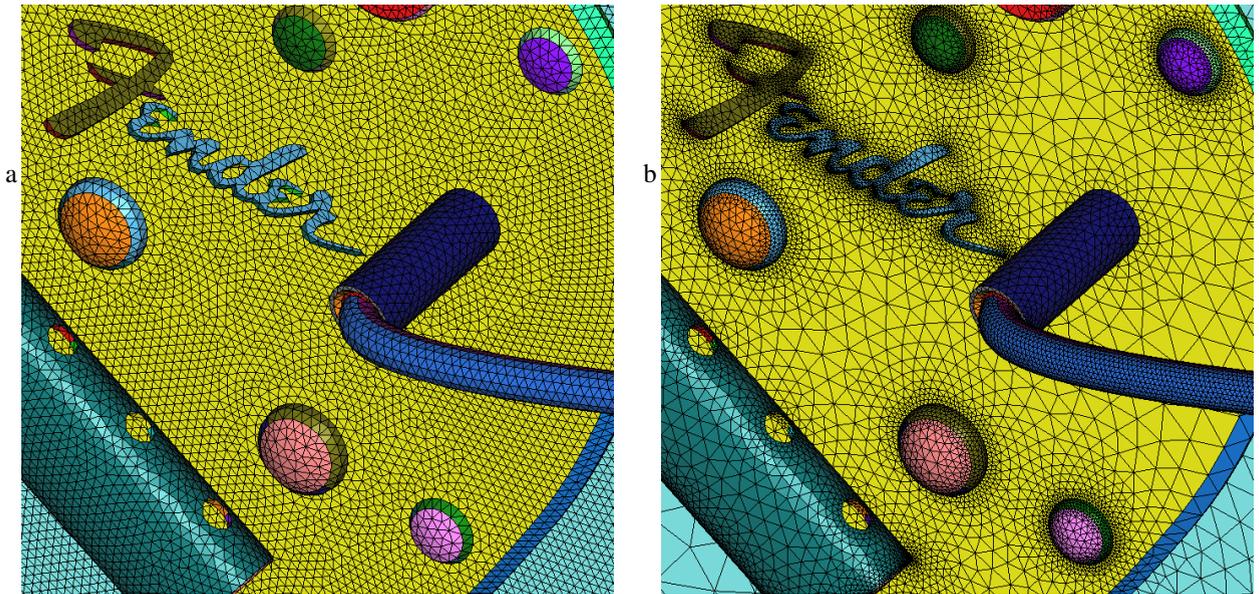


Fig. 11. (a) Uniform mesh of an electric guitar ($h = 1.0$). (b) Geometric mesh (angle 8° , gradation 1.25).

- [2] F.J. Bossen, P.S. Heckbert, A Pliant Method for Anisotropic Mesh Generation, 5th International Meshing Roundtable '96 Proceedings, pp. 375-390, 1996.
- [3] A. Dolenc, I. Mäkelä, Optimized triangulation of parametric surfaces, Mathematics of Surfaces IV, Sept. 1990.
- [4] D. Filip, R. Magedson, R. Markot, Surface algorithm using bounds on derivatives, Comput.-Aided Geom. Des., 3:295-311, 1986.
- [5] R. Finkel, J.L. Bentley, Quad Trees: A Data Structure for Retrieval on Composite Keys, Acta Informatica 4 (1): 19, 1974.
- [6] E. Hartmann, A marching method for the triangulation of surfaces, The Visual Computer, 14:95-108, 1998.
- [7] P. Laug, H. Borouchaki, Interpolating and Meshing 3-D Surface Grids, International Journal for Numerical Methods in Engineering, vol. 58, no. 2, pp. 209-225, Sept. 2003.
- [8] P. Laug, F. Guibault, H. Borouchaki, Automatic Mesh Generation of Multiface Models on Multicore Processors, in Proceedings of the Fourth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering (PARENG 2015), Dubrovnik, Croatia, March 2015, paper 38, 13 pages.
- [9] R. Löhner, Regridding Surface Triangulations, Jour. of Comput. Phys., 126(6):1-10, 1996.
- [10] Open CASCADE Technology, <http://www.opencascade.com/>
- [11] L.A. Piegl, A.M. Richard, Tessellating trimmed NURBS surfaces, Computer Aided Design, volume 27 (1), pp. 16-26, 1995.
- [12] H. Samet, The Quadtree and Related Hierarchical Data Structures, ACM Computing Surveys, June 1984.
- [13] X. Sheng, B.E. Hirsch, Triangulation of trimmed surfaces in parametric space, Computer Aided Design, volume 24 (8), pp. 437-444, 1992.
- [14] M.S. Shephard, M.K. Georges, Automatic three-dimensional mesh generation technique by the finite octree technique, Int. J. Numer. Methods Eng., 32:709-749, 1991.
- [15] K. Shimada, Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles. 6th International Meshing Roundtable '97 Proceedings, pp. 63-74, 1997.
- [16] J.R. Tristano, S.J. Owen, S.A. Canann, Advancing Front Surface Mesh Generation in Parametric Space Using a Riemannian Surface Definition, 7th International Meshing Roundtable '98 Proceedings, pp. 429-445, 1998.