Topological skeleton reconstruction for CAD surface mesh generation

E. Renaut∗
H. Borouchaki†
P. Laug∗

∗INRIA Paris-Rocquencourt, France, erwan.renaut@inria.fr, patrick.laug@inria.fr
†Université de Technologie de Troyes, France, houman.borouchaki@utt.fr

Abstract

This paper describes a method for meshing a given CAD surface after reconstructing its topology skeleton. CAD surfaces are now commonly used as input to generate suitable meshes for numerical simulations. A total or partial lack of information on the topology of the surface can lead to accuracy errors and inconsistencies, in particular in the generation of surface meshes. In this paper, we introduce an automatic approach for retrieving the topological skeleton of a CAD surface. The method is essentially based on pairing the curves between adjacent patches using an appropriate “score” function. Numerical examples are given to show the robustness and the efficiency of our approach.

1. Introduction

Surface meshes are often generated from surface models created by means of CAD systems. To build a conforming mesh of such a surface, it is necessary to know its topological skeleton precisely. The latter indicates how the different entities composing the surface must be “glued together”. This topological skeleton is made up of three levels, surface topology (a surface is composed of a conforming assembly of parametric patches), patch topology (a patch boundary is composed of a conforming assembly of curve segments) and curve topology (a curve is delimited by its two extremity vertices). Frequently, this topological representation is totally or partially missing in the CAD system used [1]. If the curve topology and the patch topology are only known, the surface topology reconstruction requires to identify the adjacency relation between the patches.
Since the mid-90s, reconstruction methods have been proposed for the case of manifold surfaces. A first idea consists in bringing geometric entities closer using tolerance distances [2]. In this approach, the problem complexity depends on the local geometric configuration of the patches composing the surface: this is a multiscale geometry problem. To improve the robustness of the reconstruction, it is possible to enhance the method by using discrete [3] or ranking [4] techniques. Experience shows that, in the case of complex CAD surfaces, these methods are unable to reconstruct the topology without the intervention of an external user. Our approach consists in obtaining the topological skeleton in a fully automatic manner for manifold surfaces. First, the topological skeleton of the surface is made conforming for curve extremities; to this end, we use a property induced by manifold surfaces concerning the adjacencies between patches through curves, namely each boundary curve shares at most two patches. Then, chaining the adjacent curves gives, in the case of a surface with borders, the different connected contours. Sample results on complex surfaces show the efficiency of our approach. The method we are proposing is an evolution of the method we have already presented before in [5]. A new strategy has been adopted to enhance the robustness for complex surfaces from different industrial fields.

2. General scheme

The CAD surface models used in this work are generally provided in the form of IGES files [6]. This CAD standard essentially describes the analytical equations of the different patches composing the surface. We use the CAD platform Open Cascade [7] to get this geometrical information. This tool also provides some “minimal” information on the topological skeleton. More precisely, only the curve topology and the patch topology are known. The surface model is organized as a set of patches, curves and vertices. Our purpose is to recover this correspondence by grouping identical (in a topological meaning) curves between neighboring patches. Assuming that the surface is manifold provides an important topological property: each curve defining this surface is shared by exactly two patches (except, for a surface with boundaries, the boundary curves which belong to only one patch). In our case where the topology between patches is missing, this means that for each curve of the surface, there is only one other corresponding curve. The problem thus reduces to finding pairs of topologically identical curves located at the interface between two patches. Our approach includes two steps. Firstly, curve extremities belonging to neighboring patches are associated, relying on the above property: two extremities of a curve are respectively associated with two extremities of
the corresponding curve. If the extremities of this corresponding curve do not match, it is necessary to split the curve so that the conformity of the skeleton is ensured. Secondly, curves belonging to neighboring patches are associated. Curves whose extremities are associated two by two are merged. Once the topological skeleton has been reconstructed, a conforming mesh of the surface can be generated. The remainder of this paper is organized as follows. In section 3, we detail the vertices association which is used in the topology reconstruction algorithm, presented in section 4. Then, section 5 describes the meshing method we use. Before concluding, results obtained for different surfaces with their respective meshes are shown in section 6.

3. Vertices association

The reconstruction algorithm we propose is based on a notion of proximity between geometrical entities, which is introduced in this section.

3.1 Balls of curve extremities

We focus more particularly on the proximity between vertices and curves. In a lot of configurations between patches, a curve extremity may have no faced vertex belonging to its neighboring patch (ex: T-joint patches). In this case, a new vertex must be created on the faced curve. In order to identify those configurations, we introduce the notions of curve ball at a curve extremity. Given a CAD surface, let us denote by \( \hat{C} = \{C_1 \ldots C_n\} \) the set of \( n_c \) curves defining its constituting patches, and \( \hat{v} = \{v_1 \ldots v_n\} \) the set of its \( n_v \) vertices.

**Definition.** Let \( \delta(v, C) \) be the minimum distance between a curve extremity \( v \in \hat{v} \) and a curve \( C \in \hat{C} \). The curve ball \( B_c(v, \varepsilon) \) of the curve extremity \( v \) and of radius \( \varepsilon \) is defined by the following set:

\[
B_c(v, \varepsilon) = \{C_i \in \hat{C}, \quad i \in \{1 \ldots n_c\} \mid \delta(v, C_i) \leq \varepsilon\}
\]  

(1)

Before each processing phase, all the curve balls of curve extremities are computed and saved in memory data structures to speed up the core steps.

3.2 Association rules

The topological reconstruction of vertices consists in bringing together topologically equal vertices of neighboring patches. If a vertex is isolated and is sufficiently close to a curve, this curve must be subdivided to make a vertex association possible. Let us recall that the vertex association is based on the curve topology of manifold surfaces. The association of vertices
is only made through curves (two vertices can be associated if they are extremities of two curves belonging to different patches). Moreover, this association is validated if these two curves make the best pair (in terms of topological correspondence) among all the possible pairs between different curves of the surface.

### 3.3 Score computation

Actually, the previous principle of association can be expressed as follows: given a curve, find the curve which (topologically) matches at best, among all the neighboring curves. To quantify the notion of “matching” between two curves, we introduce the notion of score. Let \( a, b \) be the extremities of a curve \( C_1 \), and let \( c, d \) be the extremities of a curve \( C_2 \). Let \( \delta_a, \delta_b \) be the respective distances of \( a, b \) to \( C_2 \) and \( p_a, p_b \) their projected points. Let \( \delta_c, \delta_d \) be the respective distances of \( c, d \) to \( C_1 \) and \( p_c, p_d \) their projected points.

**Definition.** The score between curves \( C_1 \) and \( C_2 \), denoted by \( S(C_1, C_2) \) is defined as follows:

\[
\text{if } \min(\delta_a, \delta_b, \delta_c, \delta_d) = \begin{cases} 
\delta_a & \text{then } S(C_1, C_2) = \delta_a + \min(\delta_b, A) \\
\delta_b & \text{then } S(C_1, C_2) = \delta_b + \min(\delta_a, B) \\
\delta_c & \text{then } S(C_1, C_2) = \delta_c + \min(\delta_d, C) \\
\delta_d & \text{then } S(C_1, C_2) = \delta_d + \min(\delta_c, D)
\end{cases}
\]

(2)

where the \( A, B, C \) and \( D \) values are:

\[
A = \begin{cases} 
\delta_b & \text{if } p_a = d \text{ or } p_d = a \\
\delta_d & \text{if } p_a = c \text{ or } p_c = a \\
\min(\delta_c, \delta_d) & \text{otherwise}
\end{cases}
\]

(3)

\[
B = \begin{cases} 
\delta_c & \text{if } p_b = d \text{ or } p_d = b \\
\delta_d & \text{if } p_b = c \text{ or } p_c = b \\
\min(\delta_b, \delta_d) & \text{otherwise}
\end{cases}
\]

(4)

\[
C = \begin{cases} 
\delta_a & \text{if } p_c = b \text{ or } p_b = c \\
\delta_b & \text{if } p_c = a \text{ or } p_a = c \\
\min(\delta_a, \delta_b) & \text{otherwise}
\end{cases}
\]

(5)

\[
D = \begin{cases} 
\delta_b & \text{if } p_d = b \text{ or } p_b = d \\
\delta_c & \text{if } p_d = a \text{ or } p_a = d \\
\min(\delta_a, \delta_b) & \text{otherwise}
\end{cases}
\]

(6)

**Remarks.** If the two curves share the same extremities then \( S(C_1, C_2) = 0 \). Values \( A, B, C \) and \( D \) are made up to guarantee the good orientation of \( C_1 \) and \( C_2 \) while computing the score for each configuration. Before computing the score between \( C_1 \) and \( C_2 \), we eliminate all the cases which are not acceptable, i.e. the \( p_a = p_b \) and \( p_c = p_d \) cases.

So, it is possible to rank the curve pairs according to their scores. A similar approach using a score function has been proposed in [4]. The score indicates how the curve extremities topologically match. If this score
is minimal, their extremities are likely to be associated. Pairs whose curves jointly make the same minimal score are called “candidate pairs”.

### 3.4 Ranking the curve pairs

A list of candidate pairs can be obtained by sweeping the neighborhood of each curve using the *curve balls*. After this process, some curves may not be included in a pair. This is the case when the ball of each extremity of the curve is empty. Such curves generally belong to the surface boundary, or else it means that $\varepsilon$ should be increased (see section 4). There are two main configurations for a curve belonging to a pair, double case (the curve belongs to both balls of the extremities of the other curve) and simple case (the curve belongs to only one ball). The candidate pairs obtained are divided into two distinct groups according to the configuration of the curves. The first group gathers all the pairs in which at least one curve is in double configuration, called “double pairs”. The remaining pairs, in which at least one curve is in simple configuration, form a second group of “simple pairs”. No other configuration is possible.

### 3.5 Association algorithm

To ensure the conformity of the topological skeleton, new vertices may be inserted on the surface, thus creating new curves. Each new curve can form a new candidate pair. In order to organize the processing of curve pairs, the vertices association is decomposed into two embedded loops (see algorithm 1). Double pairs, which represent the most convenient configuration, have priority over simple pairs. The algorithm of vertex association follows this priority rule to avoid problems that would occur if the pairs were associated randomly “on the fly”.

### 3.6 Validity of merging curves

In algorithm 1 described below, we associate the vertices belonging to a pair of curves. Sometimes, when the extremities of a pair of curves are likely to be associated, it is possible that the curves do not have to be merged. In fact, we have configurations where the pair of curves defines the boundary of a hole in the surface. To control those cases we check the validity of the merging curves operation before associating vertices. We introduce a new tolerance constant $\Delta$. This constant sets the distance limit below which two curves, whose vertices could be associated two by two, must be merged. A representative sample of distances between these curves is easily computed. If all the distances are less than $\Delta$ then the two curves are likely to be merged and its vertices are allowed to be associated.
Algorithm 1: Associate the vertices.

Input: list of curve pairs \( [C] \)

repeat
  insertion2 ← false;
  repeat
    foreach curve pair \((C_1, C_2) \in [C]\) do /* double pairs */
      if \((C_1, C_2)\) is a double pair then
        if insertion is required then
          insertion2 ← true;
          subdivide the curves;
        end
        associate the matching vertices;
        delete \((C_1, C_2)\) from \([C]\);
      end
    end
  until insertion2 = false;
  insertion1 ← false;
  foreach curve pair \((C_1, C_2) \in [C]\) do /* simple pairs */
    if \((C_1, C_2)\) is a simple pair then
      insertion1 ← true;
      subdivide the curves;
    end
    associate the matching vertices;
    delete \((C_1, C_2)\) from \([C]\);
  end
  until insertion1 = false;

4. Topology reconstruction

The vertex association procedure we have detailed before constitutes the core of the topology reconstruction process. To improve the robustness and the adaptivity of our method, this procedure is included into an iterative loop (see algorithm 2).

4.1 Computation of \( \varepsilon \)

The computation of \( \varepsilon \) rules the detection of pairs of curves. In order to be consistent, we suggest to initialize \( \varepsilon \) (or to fix \( \varepsilon_0 \)) by considering two dimensions of the surface, a global dimension depending on the diagonal length of the surface bounding box \((L_D)\) (thanks to \(L_D\), we define a tolerance domain where \( \varepsilon_D \) must be defined, this domain being bounded with a lower and upper thresholds \((\alpha_{\text{inf}} L_D \text{ and } \alpha_{\text{sup}} L_D)\)) and a local dimension depending on the diagonal length of the bounding box of a significant curve \((L_d)\) (in practice, we compute the diagonal lengths for all the curves and we choose the minimum; in general \(L_d\) (weighted by a coefficient \(a\)) is
Algorithm 2: reconstruction loop.

| Input: list of curves $\mathcal{C}$, number of connected components of the surface boundary $N$ |
| while true do |
| let $i$ be the current iteration step; |
| compute $\varepsilon_i$; |
| compute balls $(B_i(\varepsilon_i))$ for $\mathcal{C}_i$; |
| identify candidate pairs $(\mathcal{C}_i \rightarrow \{\mathcal{C}_i\})$; |
| associate vertices for $\{\mathcal{C}_i\}$; |
| if $\mathcal{C}_i = \mathcal{C}_{i-1}$ then |
| build the connected components $(N_i)$; |
| if $N_i = N$ then |
| exit from reconstruction loop; |
| end |
| end |

a relevant initialization). Finally, the initialization of $\varepsilon$ is given by:

$$
\begin{cases}
\text{if } \alpha L_d \begin{cases}
< \alpha_{\text{inf}} L_D & \text{then } \varepsilon_0 = \alpha_{\text{inf}} L_D \\
> \alpha_{\text{sup}} L_D & \text{then } \varepsilon_0 = \alpha_{\text{sup}} L_D
\end{cases} & \text{else } \varepsilon_0 = \alpha L_d
\end{cases}
$$  (7)

We define a simple increasing $\varepsilon_{i+1} = \beta \varepsilon_i$ (where $\beta$ is a constant factor) to fix the evolution of $\varepsilon$ among the iterative steps.

4.2 Connected components

In the reconstruction process, the ending of the loop (which stops the increasing of $\varepsilon$) is controlled using conditions (1) and (2) (see algorithm 2). Condition (1) checks if there were no association made between the current iteration and the previous one. If this is true, some cases of pair of corresponding curves can also remain. This situation appears when the $\varepsilon$ value is still too small to detect those isolated pairs. To enable this detection, we add condition (2) to continue the increasing of $\varepsilon$ if necessary. This condition is based on a connected components reconstruction. Indeed, the boundary curves of a surface can be gathered into one or several connected components (see figure 1). Connected components can also be formed with the remaining isolated pairs of curves. Each time this step is reached, the idea is to rebuild all the connected components with the remaining curves. While we find more connected components that the boundary contains, we continue to increase $\varepsilon$. The topology reconstruction ends when the number of connected components of the boundary is retrieved.
Figure 1. Example of a surface whose boundary curves make three connected components.

5. Surface meshing

An indirect method for meshing general parametric surfaces conforming to a pre-specified size map is used (for more details, see [8] and [9]). In the following, we briefly recall the general scheme of this meshing approach. Let $\Sigma$ be such a surface parameterized by $\sigma : \Omega \rightarrow \Sigma$, $(u,v) \mapsto \sigma(u,v)$, where $\Omega$ denotes the parametric domain. First, from this size specification, a Riemannian metric $M_3 = \frac{1}{h^2} I_3$ ($h$ being the specified size function, supposed here to be a scalar value, and $I_3$ the identity matrix) is defined so that the desired mesh has unit length edges with respect to the related Riemannian space (such meshes being referred to as “unit” meshes). Then, based on the intrinsic properties of the surface, namely the first fundamental form:

$$M_\sigma = \begin{pmatrix} \sigma_u^T \sigma_u & \sigma_u^T \sigma_v \\ \sigma_v^T \sigma_u & \sigma_v^T \sigma_v \end{pmatrix},$$

the Riemannian structure $M_3$ is induced into the parametric space which yields $\tilde{M}_2 = \frac{1}{h^2} M_\sigma$. The initial size specification is isotropic while the induced metric in parametric space is in general anisotropic, due to the variation of the tangent plane along the surface. Finally, a unit mesh is generated completely inside the parametric space such that it conforms to the induced metric $\tilde{M}_2$. This mesh is constructed using a combined advancing-front – Delaunay approach applied within a Riemannian context: the field points are defined after an advancing front method and are connected using a generalized Delaunay type method [10].

One can control explicitly the accuracy of a generated element with respect to the geometry of the surface if careful attention is paid. Indeed, a mesh of a parametric patch whose element vertices belong to the surface is “geometrically” suitable if the two following properties hold: all mesh
elements are close to the surface and every mesh element is close to the tangent planes related to its vertices. A mesh satisfying these properties is called a geometric mesh. In reference [9], we show that a mesh whose element size is locally proportional to the minimal radius of curvature is a geometric mesh. Note that if a given size map is specified, the two above properties can be locally violated. In fact, it is more useful to find a compromise between the geometric approximation of the surface and the size map conformity.

The indirect method which has just been described for one parametric surface can be easily extended to composite parametric surfaces meshing. The main difference lies in the discretization of interface curves which represent the common boundary of several patches. In fact, these interfaces must be discretized directly in $\mathbb{R}^3$ in contrast to the case of one unique patch. Indeed, it provides a unique discretization of contours in related parametric spaces so as to ensure the conformity of the global surface mesh. Therefore, to discretize the interface curves we should know the inverse of the mapping functions defining the surface. To avoid this, we can use an approximation of interface curves in $\mathbb{R}^3$ and also in parametric spaces by polyline segments.

### 6. Numerical results

This method has been implemented in a software program which displays several results in tabular form. For each surface, the following table contains the number of connected components of the boundary ($n_b$), the initial number of patches and curves ($n_p$ and $n_c$), the number of created curves ($n_i$) and its percentage compared to the initial number of curves (%) and the computation time (in seconds) using a Pentium M at 1.7GHz.

<table>
<thead>
<tr>
<th>Test surface</th>
<th>$n_b$</th>
<th>$n_p$</th>
<th>$n_c$</th>
<th>$n_i$</th>
<th>%</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangle</td>
<td>1</td>
<td>38</td>
<td>263</td>
<td>50</td>
<td>19</td>
<td>0.21</td>
</tr>
<tr>
<td>Aircraft</td>
<td>0</td>
<td>83</td>
<td>739</td>
<td>29</td>
<td>4</td>
<td>0.40</td>
</tr>
<tr>
<td>Pillar</td>
<td>1</td>
<td>902</td>
<td>4670</td>
<td>47</td>
<td>1</td>
<td>1.35</td>
</tr>
<tr>
<td>Fender</td>
<td>1</td>
<td>1106</td>
<td>5392</td>
<td>862</td>
<td>16</td>
<td>4.09</td>
</tr>
<tr>
<td>Decklid</td>
<td>1</td>
<td>964</td>
<td>10868</td>
<td>5216</td>
<td>48</td>
<td>8.41</td>
</tr>
<tr>
<td>Motor</td>
<td>0</td>
<td>2381</td>
<td>12300</td>
<td>492</td>
<td>4</td>
<td>9.78</td>
</tr>
</tbody>
</table>

Several examples of surface meshes generated thanks to their respective topological skeletons are briefly presented hereafter (figures 2 and 3).
7. Conclusion

Using the approach presented in this paper, the topology of a CAD surface can be automatically reconstructed. The extremities of curves located at the interface between two patches are associated: the efficiency of the method lies on the association of these vertices by using the correspondence between adjacent curves. These curves are finally merged and the connected components of the surface boundary are identified. Finally, the topological skeleton provides a complete description of the adjacencies between patches for the surface mesher.

References


Figure 2. Left: skeleton of an aircraft ©EADS and corresponding uniform and geometric meshes. Right: skeleton and meshes of a pillar ©Audi.
Figure 3. Left: skeleton of a decklid © GM and corresponding uniform and geometric meshes. Right: skeleton and meshes of a motor © Altair.