Automatic Non-manifold Topology Recovery and Geometry Noise Removal

Aurélien Alleaume

Distene SAS, Pôle Teratec-Bard 1, Domaine du Grand rué 91680
Bruyères-le-Chatel, France
aurelien.alleaume@distene.com

Abstract. This paper presents an effective method for automatic topology recovery of non-manifold geometries. Mixing topology recovery and geometry noise detection/removal allowed us to achieve effectively the automation and robustness required by such methods. We developped our method on CAD boundary representation (BREP) geometries in the context of surface mesh generation, but it could also be directly applied to discrete STL or mesh geometries. Its reliability and efficiency has been validated on a variety of complex manifold and non-manifold CAD geometries.

1 Introduction

Even today, legacy or imported CAD models often lack a large amount of topological information (patch connectivity) or contain some geometrical noise entities (like micro-curves) which are often due to translations involved when exchanging data between CAD systems. These can prevent a conforming boundary representation of the surface assembly from being built straightaway.

This especially becomes a problem for modern parametric meshing methods like [5] when it comes to generate a conforming surface mesh.

Even though great improvements were allowed by modern file formats such as STEP, these issues still did not totally disappear especially on large geometries made of complex assemblies.

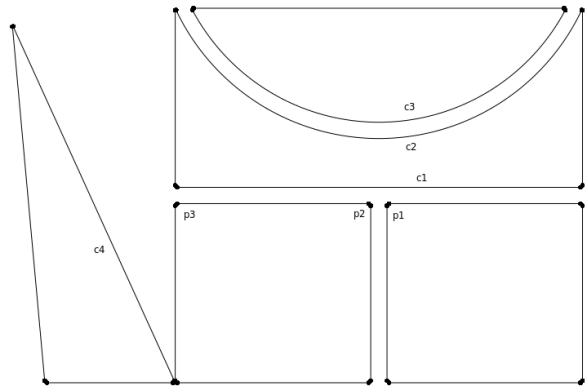
Let us recall that boundary representation models are composed of two parts:

1. Geometry: surfaces, curves and points.
2. Topology: faces, coedges, edges and vertices. A face being a connex component of a surface bounded by coedges; a coedge a bounded interval of a curve. Edges are bounded by vertices and associate coedges accross patches to define how faces are connected. Vertices associate points to define how edges are connected. Higher level entities such as lumps and shells are often defined but are beyond the scope of the method we propose.

The problem of topology recovery is to compute coherent topological edges and vertices data from :

1. the given geometry,
2. the topology of faces and coedges.

In other words, points and curves must be associated to one another to define how faces are linked together. This might also require some curves to be split as it will be illustrated later on figure 1.



Let us consider that this whole figure is smaller than the tolerance ε : p_1 and p_2 will be identified. p_1 will also be projected on c_1. p_1 will not be projected on c_2 because $face(c_2) \in F(p_1)$. Last, neither p_1 will be projected on c_3 nor p_3 on c_4 because the reverse projection condition contained in $C_{point}()$ is not satisfied.

Fig. 1. Vertex projection and association

Methods for topology recovery have been proposed since the early 90's. Tolerance driven methods which associate all geometrical entities within a given threshold quickly show their limits as the CAD geometries are often multiscale, making the global tolerance adjustment complex, case dependent and even sometimes impossible.

In [2], a method to recover the topology of a mesh based CAD is proposed. It uses heuristics to determine wrong associations of edges, thus removing the burden of tolerance tuning. The method proposed by [3] to repair a discrete polyhedral CAD geometry is to first compute a shared-vertex polygonal representation and then merge each polygon edge with the most appropriate edge. While providing more automation, these methods only apply to discrete mesh based CAD geometries and do not remove the need for a human intervention on complex models.

The approach recently described in [1] based on scoring function to pair curves together extends these methods to continuous CAD models, providing more automation and robustness. Unfortunately, it applies only to manifold geometries and it is still not adapted to models containing geometry noise such as micro-curves.

Our experience with CAD models and topology reconstruction is that, without effective geometry noise detection and removal, a reconstruction method cannot be effective on complex or dirty models. This led us to another approach whose basic keys are:

1. to **mix topology recovery with geometry noise detection and removal**. Since the geometrical noise detection is a multiscale problem, it is mainly made through topological considerations without requiring another extraneous tolerance threshold. This allows us to detect automatically noise, without removing small features.
2. to **associate geometrical entities (points then curves) as long as it makes sense from a geometrical/topological non-manifold point of view** and the optionally specified tolerance is not reached.

One of the major strengths of our method is its generality. Unlike [2] it is not based on several specific heuristics but only on a couple of criteria: one for vertex association and one for edge association. Moreover it provides greater automaticity without restricting the field of application to manifold geometries like [1]. Last but not least, non-manifold geometry features do not require any special treatment but are treated as the general case.

2 Algorithm Description

2.1 Pre-requisites

We suppose that the intra-face topology is known or has already been reconstructed. In other words, vertex association is already achieved inside each face of the input geometry. We concentrate here on the reconstruction of the inter-face topology which is the most complex part of the problem.

To simplify notations, we will use the term *curve* instead of *coedge* from now on.

Definition 1.

For a curve c, we will note $\hat{c}$ the edge corresponding to c, i.e. the set of curves associated to c.

For a point p, we will note $\hat{p}$ the vertex corresponding to p, i.e. the set of points associated to p.

We will also define $order(p)$ as the number of points in $\hat{p}$ and $order(c)$ as the number of curves in $\hat{c}$.

In the illustrations of this section, points associated together as a vertex are surrounded by a dashed line ellipsis.

2.2 The Optional ε Tolerance Parameter

The only parameter the user can optionally tune in the algorithm is the tolerance ε. It defines the size of the smallest feature or gap the user wants to preserve. Even if some modelling artifacts or features are smaller than this tolerance, it will not be a problem for our reconstruction algorithm.

Moreover, in our method, the effects of a given ε are overriden by the following rules:

- a geometric detail smaller than ε will be kept unless it prevents conformity achievement;
- no vertex/edge association above ε will be made
- a vertex/edge association below ε will not be made if it is in contradiction with the local geometrical/topological configuration.

In practice, this ε parameter is only optional and practically never needs to be tuned because our method is mostly driven by the local geometry and topology.

2.3 Pre-processing

Since a great part of the algorithm is based on the notion of proximity between geometrical entities, curves must be discretized and accelerating structures need to be built in order to achieve a good implementation of this method. The algorithm will mainly need:

- to localize and enumerate close curves in the space,
- to enumerate the edges connected to each vertex.

2.4 Vertex Association and Projection

This step is the heart of the method: it detects which vertex associations or projections make sense or not, given the local geometry/topology. It also removes a part of the geometry noise using a collapsing technique.

Definition 2.
For a vertex v, let us define :

- $\mathcal{C}(v)$ *the set of curves c such that v is an extremity of c*
- $\mathcal{F}(v)$ *the set of faces f such that $v \in f$*

Definition 3.
For a point p and a curve c we define:

- $face(c)$ *as being the face c belongs to.*
- *the distance between p and c as $d(p,c) = min_{a \in c} d(p,a)$ where $a \in c$ means that a is a point of c.*
- $P_c(p) \in c$ *the projection of p on c, i.e. a point such that $d(p,c) = d(p, P_c(p))$*

Let us remark that in this definition:

- $P_c(p)$ is not necessarily an orthogonal projection;
- $P_c(p)$ always exists (we consider a finite space) but is not necessarily unique. In this latter case, any choice of minimum can be made.

Definition 4. *For a curve c, we will say that 2 points $p_1 \in c$ and $p_2 \in c$ are compatible if $d_c(p_1, p_2)$, the distance between p_1 and p_2 along c, verifies: $d_c(p_1, p_2) \leq min\left\{\varepsilon, \frac{1}{10} \times length(c)\right\}$.*

Let us remark that in this definition the choice of the constant $\frac{1}{10}$ might seem arbitrary, but this choice does not actually have a strong impact on the algorithm behavior.

Definition 5. *For a set of faces F, we note $P_F(p)$ the projection (in the sense of definition 3) of the point p on all the curves of the faces in F.*

We can now introduce the criterion upon which the vertex association is decided:

Definition 6. *Let p be a point and c a curve. We define the proposition $C_{point}(p, c)$ as: $d(p, c) \leq \varepsilon$ and $P_{\mathcal{F}(\hat{p})}(P_c(p))$ is compatible with p.*

The last condition of this definition can be seen as a form of reflexivity condition and can be interpreted as "the reverse projection of the projection of p is topologically equivalent to p".

We will use the condition C_{point} to decide whether merging two points or projecting a point on a curve makes sense or not from a geometrical/topological point of view. The distance condition will prevent us from merging entities beyond the tolerance while the reverse projection condition will prevent us from creating faces overlaps or partially collapsing faces.

The algorithm 1 to project and associate vertices is illustrated on some common cases on figure 1. The reader should note that geometry noise removal happens when we remove (if it exists) the smallest edge between the two points we are about to identify.

2.5 Edge Association

The edge association step heavily depends on the results of the previous vertex association step: for all curves sharing the same vertex extremities, it determines whether they should be associated as a single edge or not.

Definition 7.

For an edge e, let $\mathcal{F}(e)$ be the set of faces f such that $e \in f$.

For 2 curves c_1 and c_2 we define the distance between c_1 and c_2 as:

$$d(c_1, c_2) = max\left\{max_{a \in c_1} d(a, c_2), max_{b \in c_2} d(b, c_1)\right\}$$

```
foreach point p in the model do
    list = ∅;
    foreach curve c such that d(p, c) ≤ ε do
        list += c;
    end
    sort list in the ascending order relatively to the function d(p, .);
    foreach curve c in list (in order) do
        if face(c) ∉ F(p̂) then
            if C_point(p, c) then
                p_1 = the first extremity of c ;
                p_2 = the second extremity of c ;
                if p and p_1 are compatible then
                    remove (if exists) the smallest edge between p̂ and p̂_1;
                    identify p and p_1 ;
                else if p and p_2 are compatible then
                    remove (if exists) the smallest edge between p̂ and p̂_2;
                    identify p and p_2 ;
                else
                    split the curve c at P_c(p);
                    identify P_c(p) and p;
                end
            else
                // merging further does not make topological sense
                end of loop on curve c;
            end
        end
    end
end
```

Algorithm 1. The vertex projection and association algorithm

Definition 8. *For 2 curves c_1 and c_2 we define the proposition $C_{curve}(c_1, c_2)$ as: $d(c_1, c_2) \leq \varepsilon$ and $\mathcal{F}(\hat{c_1}) \cap \mathcal{F}(\hat{c_2}) = \emptyset$.*

The condition C_{curve} is the criterion upon which the edge association is decided. The distance condition will prevent us from merging curves beyond the tolerance while the empty intersection condition will prevent us from creating faces overlaps and from partially collapsing faces.

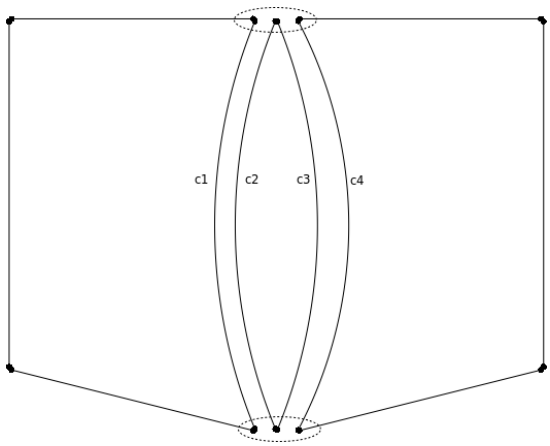
This edge association step is described in algorithm 2 and illustrated on a simple case on figure 2. For clarity concerns, we did not detail the simple extra treatment which can be added to deal correctly with some face overlaps.

2.6 Topology Post-processing

While constructing vertices association, we may have removed already some geometry noise, but there might still be some micro-curves preventingthe

```
foreach curve c1 in the model do
    list = ∅;
    foreach curve c2 such that c1 and c2 share the same extremities do
        list += c2;
    end
    sort list in the ascending order relatively to the function d(., c1);
    foreach curve c2 in list (in order) do
        if C_curve(c1, c2) then
            identify curves c1 and c2;
        else
            // merging further for c1 would not make topological sense
            end of loop on c2;
        end
    end
end
```

Algorithm 2. The edge association algorithm

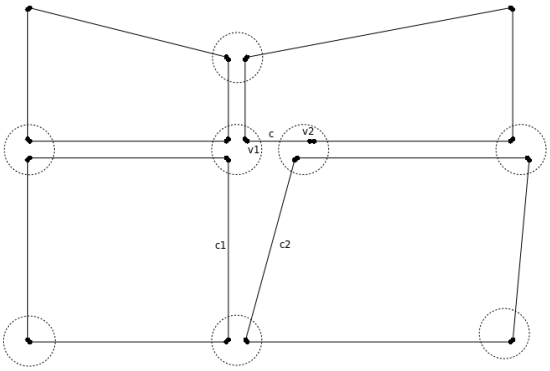


Let us consider that this whole pattern is smaller than the tolerance ε: curves c_1 and c_2 will be merged, as well as c_3 and c_4 but c_1 will not be identified to c_3 because $\mathcal{F}(\hat{c_1}) \cap \mathcal{F}(\hat{c_3}) \neq \emptyset$ (c_1 and c_2 have been identified before).

Fig. 2. Edge association

global conformity to be achieved, as illustrated on figure 3. The goal of the post-processing which is described in algorithm 3 is to identify and collapse them.

The reader can remark that at the end of the post-processing, any face whose curves have all been removed is removed.



Let us consider that c is a small curve preventing conformity. It will be removed, vertices v_1 and v_2 will be merged, as well as curves c_1 and c_2

Fig. 3. Noise post-processing

```
list = ∅;
foreach curve c in the model do
    if order(c) == 1 and length(c) ≤ ε then
        list += e;
    end
end
sort list in the ascending order relatively to the length of the curves;
foreach curve c in list (in order) do
    let v1 and v2 be the vertex extremities of c;
    if order(c) == 1 then
        foreach curve c1 ∈ C(v1) such that c1 ≠ c do
            foreach curve c2 ∈ C(v2) such that c2 ≠ c do
                if C_curve(c1, c2) then
                    identify vertices v1 and v2;
                    remove curve c;
                    if c1 and c2 share the same extremities then
                        identify curves c1 and c2;
                    end
                    continue with the next iteration on c;
                end
            end
        end
    end
end
remove any face whose curves have all been removed;
```

Algorithm 3. The topology post-processing algorithm

3 Applications

We implemented our algorithm in PreCAD, a module aimed at preparing CAD data for surface meshing. The CAD is first imported from a CAD engine, such as for example Opencascade [9] or ACIS [10]; the topology of each face is then reconstructed separately. Our algorithm is then applied to recover the global topology.

After this processing, the CAD is meshed with BLSURF [5]. To test our method deeper, all the input topology information for vertices and edges was discarded from the input CAD. Last but not least, the capacity to change the optional tolerance parameter was not used for any of these cases. Our goal was indeed to assess a fully automatic approach.

Figures 4, 5, 6, 7 and 8 show typical examples of geometries we are able to reconstruct automatically with our method.

We tested our method on a large test base of 800 more or less complex IGES, STEP and SAT cases with successful results. Table 1 gives some statistics on a representative set of industrial geometries. It presents:

- Input faces/curves: the number of faces and curves the input CAD geometries is made of.
- Removed faces/curves: the number of faces and curves that our algorithm decided to collapse because they were considered as noise, preventing conformity achievement.
- Free edges: the number of edges that remain unassociated after the process. One can expect this count to be null for a closed manifold surface.
- Non-manifold edges: the number of topological edges that have a class with at least 3 geometrical curves.

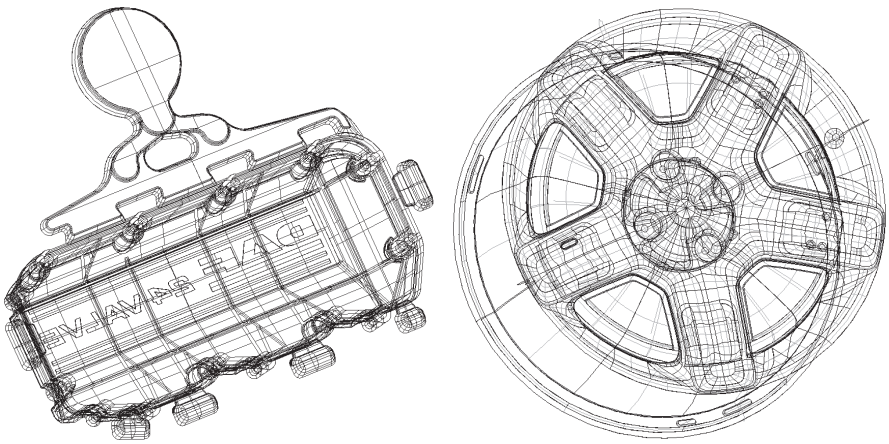


Fig. 4. Fad and Wheel geometries made of respectively 1,328 and 1,447 faces (courtesy of Fathi El-Yafi)

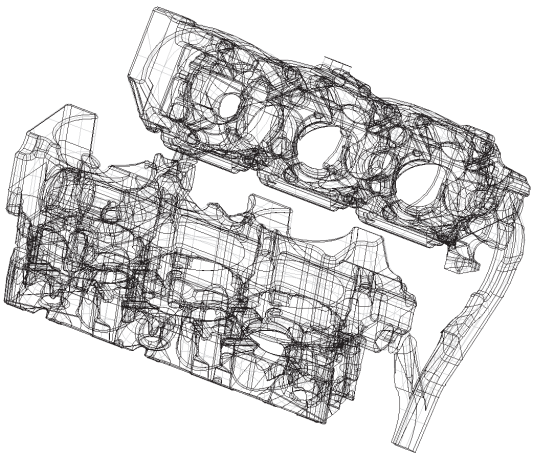


Fig. 5. V6 engine part made of 7,268 faces (courtesy of Fathi El-Yafi)

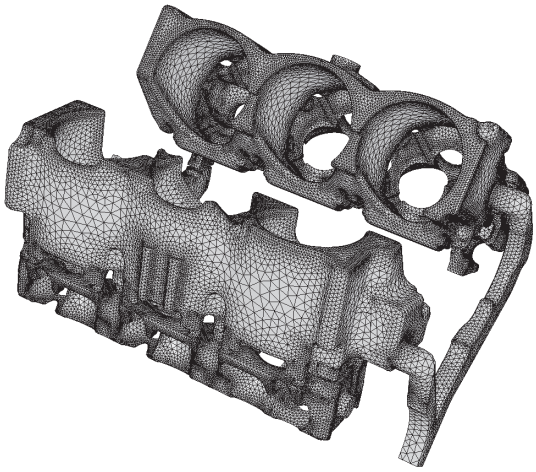


Fig. 6. V6 engine patch-independent surface mesh generated by BLSURF

- Time: CPU time required by the computation of the topology on an AMD Opteron 2.4 GHz.
- Valid volume: "yes" means that a volume mesh can be generated from the derived surface mesh with the surface constrained Delaunay mesher Tetmesh-GHS3D (see [6]). In this case, the surface mesh defines a volume and does not self-intersect. We can thus consider with a high probability the reconstruction as a success.
- Reconstruction: tells whether the reconstruction can be considered as successful or not. This is mainly determined by examining free and non-manifold (higher order) edges and visually checking in the generated surface

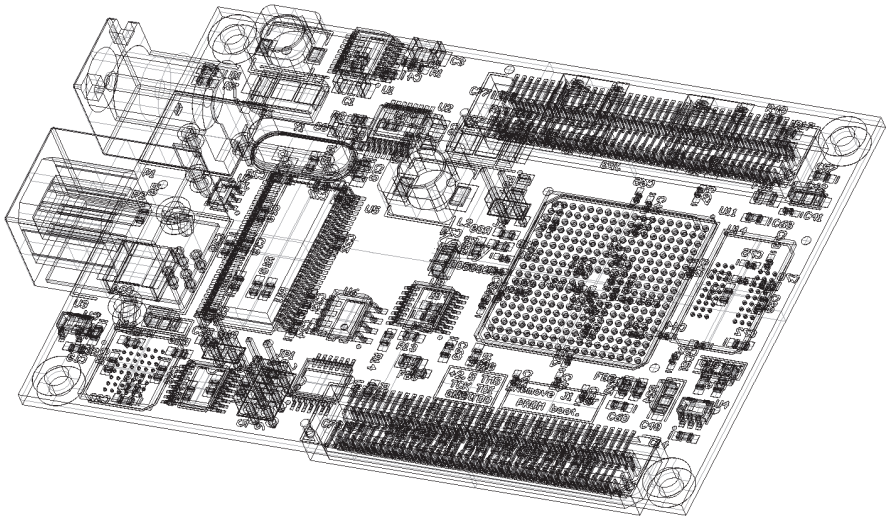


Fig. 7. Circuit board non-manifold multi-scale geometry made of 31,705 faces

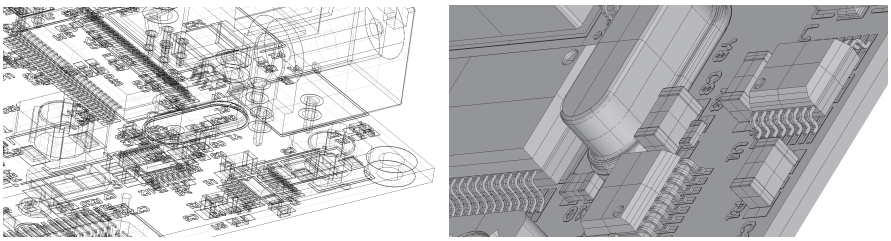


Fig. 8. Circuit board details

Table 1. Results

CAD Geometry	Mold	EBox	Fad	Wheel	V6	VX	Circuit
Input faces	381	2,325	1,328	1,447	7,268	18,700	31,705
Input curves	2,201	11,134	15,985	10,874	40,015	100,068	158,460
Removed faces	0	0	0	0	16	4	0
Removed curves	9	0	16	126	1,717	89	14
Free egdes	4	0	0	0	0	0	0
Non-manifold edges	0	162	0	0	0	3,620	524
Time	0.5s	1.25s	2.01 s	1.74s	7.72s	35.52s	28.01s
Valid volume	no	no	yes	yes	no	no	no
Reconstruction	failure	success	success	success	success	success	success

mesh that the computed topology is correct and conforms to expectations in these area.

These results lead to several remarks:

1. Even on large, complex and noisy geometries, the method is fast and really reliable, without any parameter tuning.
2. As expected geometry noise is removed in varying quantities, (from 0 to 4.2 percent) depending on the quality of the input case (see removed curves/faces rows).
3. Non-manifold (higher order) edges are correctly reconstructed in the non-manifold geometries (EBox, VX and Circuit). As expected, non-manifold edges are not created in manifold geometries.
4. The "Mold" case is considered as a failure because two curve associations were beyond the default tolerance ε and thus were rejected, leaving four free edges. In other words, this case could be successfully processed when changing the ε parameter. This shows that even though tolerance tuning issues have been drastically reduced, the human intervention cannot be totally eliminated from the reconstruction process when the geometry is too far from the intended topology.
5. Even when the reconstruction is successful, the derived surface mesh is not always directly suitable for volume mesh generation. This is often due to a wrong CAD design or to missing CAD boolean operations. To correct this kind of default, a face imprinting step like described in [8] should be carried out after topology recovery.

4 Conclusion

We have presented an effective automatic method to recover topology information while removing geometry noise from non-manifold geometries. Its robustness and efficiency has been shown on a wide range of complex CAD geometries.

Even though we are now able to reconstruct successfully the topology from complex geometries, we still cannot guarantee that the derived surface mesh defines a volume, especially on large geometries made of complex assemblies. That is why our next concern will be to apply our approach to automatic face imprinting techniques.

Acknowledgement. This work was partially funded by the Pôle System@tic's EHPOC project.

References

1. Renaut, E., Borouchaki, H., Laug, P.: Extraction of surface topological skeletons. In: Proceedings of the Numerical geometry, grid generation and high performance computing Conference(NUMGRID 2008), pp. 256–263 (2008)

2. Weihe, K., Willhalm, T.: Reconstructing the Topology of a CAD Model: A Discrete Approach. In: Burkard, R.E., Woeginger, G.J. (eds.) ESA 1997. LNCS, vol. 1284, pp. 500–513. Springer, Heidelberg (1997)
3. Barequet, G., Kumar, S.: Repairing CAD models. In: Proceedings of the 8th conference on Visualization (VIS 1997), pp. 363–370 (1997)
4. Mezentsev, A., Woehler, T.: Methods and algorithms of automated CAD repair for incremental surface meshing. In: Proceedings of the 8th International Meshing Roundtable, pp. 299–309 (1999)
5. Laug, P., Borouchaki, H.: BLSURF Mesh Generator for Composite Parametric Surfaces. INRIA Technical Report RT-0235 (1999)
6. George, P.L., Hecht, F., Saltel, E.: Automatic mesh generator with specified boundary. In: Computer Methods in Applied Mechanics and Engineering, pp. 269–288 (1991)
7. Merazzi, S., Gerteisen, E.A., Mezentsev, A.: A generic CAD-mesh interface. In: Proceedings of the 9th International Meshing Roundtables (2000)
8. Clark, B.W., Hanks, B.W., Ernst, C.D.: Conformal Assembly Meshing with Tolerant Imprinting. In: Proceedings of the 17th International Meshing Roundtable, pp. 267–280 (2008)
9. Open CASCADE S.A.S., Open CASCADE Technology, 3D modeling & numerical simulation, `http://www.opencascade.org`
10. Spatial Corporation, 3D ACIS Modeling, `http://www.spatial.com`